`rdlocrand`: Local Randomization Methods for RD Designs*

Matias D. Cattaneo† Rocio Titiunik‡ Gonzalo Vazquez-Bare§

May 22, 2025

Abstract

The regression discontinuity (RD) design is a popular quasi-experimental design for causal inference and policy evaluation. Under the local randomization approach, RD designs can be interpreted as randomized experiments inside a window around the cutoff. The `rdlocrand` package provides tools to analyze RD designs under local randomization: `rdrandinf` to perform hypothesis testing using randomization inference, `rdwinselect` to select a window around the cutoff in which randomization is likely to hold, `rdsensitivity` to assess the sensitivity of the results to different window lengths and null hypotheses and `rdrbounds` to construct Rosenbaum bounds for sensitivity to unobserved confounders. We illustrate the implementation of these four functions, which have the same syntax and capabilities of the `Stata` commands described in Cattaneo, Titiunik, and Vazquez-Bare [2016]. For more details, and related Stata and R packages useful for analysis of RD designs, visit `https://rdpackages.github.io/`.

Keywords: regression discontinuity designs, quasi-experimental techniques, causal inference, randomization inference, finite-sample methods, Fisher's exact p-values, Neyman's repeated sampling approach.

*We thank Xinwei Ma for helpful comments. Financial support from the National Science Foundation (SES 1357561) is gratefully acknowledged.

†Department of Economics and Department of Statistics, University of Michigan.

‡Department of Political Science, University of Michigan.

§Department of Economics, UC Santa Barbara.

Contents

1 Introduction

This article illustrates the R package `rdlocrand`, which provides tools to analyze RD designs under a local randomization approach. The functions included in this package have the same syntax, and offer the same functionalities, as our companion `Stata` commands described in Cattaneo, Titiunik, and Vazquez-Bare [2016].

For brevity, we focus exclusively on software implementation issues. Extensive discussion and details on methodological and practical aspects can be found in Cattaneo, Frandsen, and Titiunik [2015], Cattaneo, Titiunik, and Vazquez-Bare [2016] and Cattaneo, Titiunik, and Vazquez-Bare [2017]. For help on the functions' syntax and related issues, please refer to reference manual.

For related Stata and R packages useful for analysis of RD designs, visit:

`https://rdpackages.github.io/`

2 Illustration of Methods

We illustrate how to implement the four functions described above using the dataset from Cattaneo et al. [2015]. This section replicates, as close as possible, section 7 of Cattaneo et al. [2016].

First, to install the `rdlocrand` package, type:

```
install.packages("rdlocrand")
```

Next, load the data:

```
> data <- read.csv("rdlocrand_senate.csv")
> dim(data)
[1] 1390   14
> names(data)
 [1] "state"             "year"              "dopen"             "population"        "presdemvoteshlag1" "demmv"
[10] "demvoteshfor2"     "demwinprv1"        "demwinprv2"        "dmidterm"          "dpresdem"
>
> # Select predetermined covariates to be used for window selector
>
> X  <-  cbind(data$presdemvoteshlag1,
+             data$population/1000000,
+             data$demvoteshlag1,
+             data$demvoteshlag2,
+             data$demwinprv1,
+             data$demwinprv2,
+             data$dopen,
+             data$dmidterm,
+             data$dpresdem)
>
> # Assign names to the covariates
>
```

```
> colnames(X) <-  c("DemPres Vote",
+                   "Population",
+                   "DemSen Vote t-1",
+                   "DemSen Vote t-2",
+                   "DemSen Win t-1",
+                   "DemSen Win t-2",
+                   "Open", "Midterm",
+                   "DemPres")
>
> # Running variable and outcome variable
>
> R <- data$demmv
> Y <- data$demvoteshfor2
> D <- as.numeric(R>=0)
```

The most basic syntax for `rdwinselect` is the following:

```
> tmp <- rdwinselect(R,X)
Mass points detected in running variable
You may use wmasspoints option for constructing windows at each mass point


Window selection for RD under local randomization

Number of obs     =          1390
Order of poly     =             0
Kernel type       =       uniform
Reps              =          1000
Testing method    =     rdrandinf
Balance test      =     diffmeans

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         640         750
     1st percentile           7           7
     5th percentile          32          37
    10th percentile          64          75
    20th percentile         127         149

================================================================================
      Window          p-value        Var. name    Bin.test      Obs<c     Obs>=c
================================================================================
  -0.5287   0.5287      0.206 DemSen.Vote.t.2       0.327         10         16
  -0.7305   0.7305      0.339  DemSen.Win.t.1       0.256         15         23
  -1.0800   1.0800      0.034            Open       0.119         19         31
  -1.3068   1.3068      0.107            Open       0.245         25         35
  -1.5386   1.5386      0.106         Midterm       0.728         35         39
  -1.6758   1.6758      0.074         Midterm       0.581         38         44
  -1.9042   1.9042      0.040         Midterm       0.602         43         49
  -2.1853   2.1853      0.071         Midterm       0.621         48         54
```

```
  -2.3665    2.3665      0.136            Open       0.637         53         59
  -2.6439    2.6439      0.164            Open       0.929         62         64
================================================================================
Recommended window is [-0.7305;0.7305] with 38 observations (15 below, 23 above).
```

Because in this particular application the cutoff is zero, which is the default value, the `cutoff` option can be omitted. For this reason, this and all the remaining examples will not specify this option. In practice, when the cutoff is not zero, the user can simply specify `cutoff = c`. Alternatively, it may be easier to simply redefine the running variable by recentering it at the cutoff. By default, `rdwinselect` uses the difference-in-means statistic to perform hypothesis tests—but this can be changed with the `statistic` option.

The output of `rdwinselect` is divided in three panels. The upper panel indicates the total sample size, the degree of the polynomial used by `rdrandinf`, the type of kernel used for the weighting scheme (`uniform`, `triangular` or `epan`), the number of replications in the permutation test (whenever this test is performed), the method used to perform the covariate balance tests (`approximate` or `rdrandinf`), the test statistic used (`test`, `ksmirnov` or `ranksum`).

The middle panel provides information on sample sizes. The first row gives the total number of observations to the left and to the right of the cutoff, and also the total sample size. The following four rows provide the same information but around small neighborhoods around the cutoffs defined by the first, fifth, tenth and twentieth percentile of the running variable.

Finally, the main panel gives the result of the two balance tests performed at each of the windows considered. The first column indicates the window. The second column, labeled "`p-value`", provides the minimum p-value of the difference-in-means test, and the name of the corresponding variable associated to this p-value is given in column 3, "`Var.  name`". The p-value is obtained by either permutation testing or a t-test, depending on the option specified. The fourth column gives the p-value from a Binomial probability test of the hypothesis that the probability of treatment is 0.5 using the `binom.test` command. Columns 5 and 6 give the number of observations to the left and right of the cutoff inside each window.

With the default options, the command recommends the window $[-0.7305; 0.7305]$, the largest window for which the minimum p-value of the balance tests is above 0.15.

The `wasymmetric` option can be used to allow for asymmetric windows. With this option, the command adds an exact number of observations at each side, without symmetrizing the window.

```
Mass points detected in running variable
You may use wmasspoints option for constructing windows at each mass point


Window selection for RD under local randomization

Number of obs     =          1390
Order of poly     =             0
Kernel type       =       uniform
Reps              =          1000
```

```
Testing method    =     rdrandinf
Balance test      =     diffmeans

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         640         750
     1st percentile           7           7
     5th percentile          32          37
    10th percentile          64          75
    20th percentile         127         149

================================================================================
      Window          p-value        Var. name    Bin.test      Obs<c     Obs>=c
================================================================================
  -0.5287   0.3525      0.078 DemSen.Vote.t.1       1.000         10         10
  -0.7305   0.4393      0.403 DemSen.Vote.t.2       1.000         15         15
  -1.0800   0.6378      0.174            Open       1.000         19         20
  -1.2911   0.7597      0.145    DemPres.Vote       1.000         24         25
  -1.3465   1.0478      0.057            Open       1.000         29         30
  -1.4936   1.2960      0.046         Midterm       1.000         34         35
  -1.6298   1.4641      0.125         Midterm       0.909         37         39
  -1.7033   1.6371      0.166         Midterm       0.826         40         43
  -1.9687   1.8617      0.061         Midterm       0.755         44         48
  -2.1933   2.1460      0.080         Midterm       0.767         49         53
================================================================================
Smallest window does not pass covariate test.
Decrease smallest window or reduce level.
```

By default and with the `wasymmetric` option, the command will start with an initial window including exactly 10 observations at each side, and add 5 observations at each step. When covariates have missing values, however, the resulting number of observations may be lower. This can be avoided using the `dropmissing` option.

Notice that `rdwinselect` gives the following message in the first line of the output:

```
Mass points detected in running variable
```

This message indicates that the running variable has repeated values, that is, multiple observations sharing the same value of the running variable. The presence of mass points in the running variable requires careful examination; see Cattaneo, Titiunik, and Vazquez-Bare [2017], Cattaneo, Idrobo, and Titiunik [2020] for details. In this application, the only mass points occur at the minimum and maximum of the running variable (which correspond to uncontested elections). Since these observations are not used for estimation and inference, this warning can be ignored in this case.

By default, `rdwinselect` starts with a window that contains at least 10 observations at each side of the cutoff, and increases the length ensuring that at least two observations are added in each successive window. The user can choose these two values using the `obsmin` and `obsstep` options, respectively, or can define the windows in terms of their length instead of the number of

observations. For instance, Cattaneo et al. [2015] start from the window $[-0.5\,;\,0.5]$ and increase the width by 0.125 using 10,000 replications in the permutation test. To replicate their results, we can type:

```
> tmp <- rdwinselect(R,X,wmin=.5,wstep=.125,reps=10000)
Mass points detected in running variable


Window selection for RD under local randomization

Number of obs      =          1390
Order of poly      =             0
Kernel type        =       uniform
Reps               =         10000
Testing method     =     rdrandinf
Balance test       =     diffmeans

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         640         750
     1st percentile           7           7
     5th percentile          32          37
    10th percentile          64          75
    20th percentile         127         149

================================================================================
  Window length / 2   p-value        Var. name    Bin.test      Obs<c     Obs>=c
================================================================================
             0.5000     0.265 DemSen.Vote.t.2       0.230          9         16
             0.6250     0.427             Open      0.377         13         19
             0.7500     0.257             Open      0.200         15         24
             0.8750     0.150             Open      0.211         16         25
             1.0000     0.074             Open      0.135         17         28
             1.1250     0.038             Open      0.119         19         31
             1.2500     0.059             Open      0.105         21         34
             1.3750     0.137          Midterm      0.539         30         36
             1.5000     0.090          Midterm      0.640         34         39
             1.6250     0.112          Midterm      0.734         37         41
================================================================================
Recommended window is [-0.75;0.75] with 39 observations (15 below, 24 above).
```

We see that the command selects the window $[-0.75; 0.75]$, as in Cattaneo et al. [2015]. However, it is important to note that these results can vary slightly because of the randomization process behind the selection procedure. Additionally, observe that the minimum p-value is not necessarily monotonic on the length of the window. The `plot` option allows the user to depict graphically how these values change for different lengths. We will set the number of windows to 80 to have more observations in the plot, and we will specify the `approx` option to speed up the calculations. By specifying this option, the command uses the large-sample approximation instead of randomization

inference. It is useful for illustration purposes as it is much faster, but it can be misleading since the approximation may be poor when the sample is small. The output from `rdwinselect` with 80 windows is a long table and will be omitted. The resulting graph is shown in Figure 1.

```
> tmp <- rdwinselect(R,X,wmin=.5,wstep=.125,approx=TRUE,nwin=80,quietly=TRUE,plot=TRUE)
Mass points detected in running variable
```

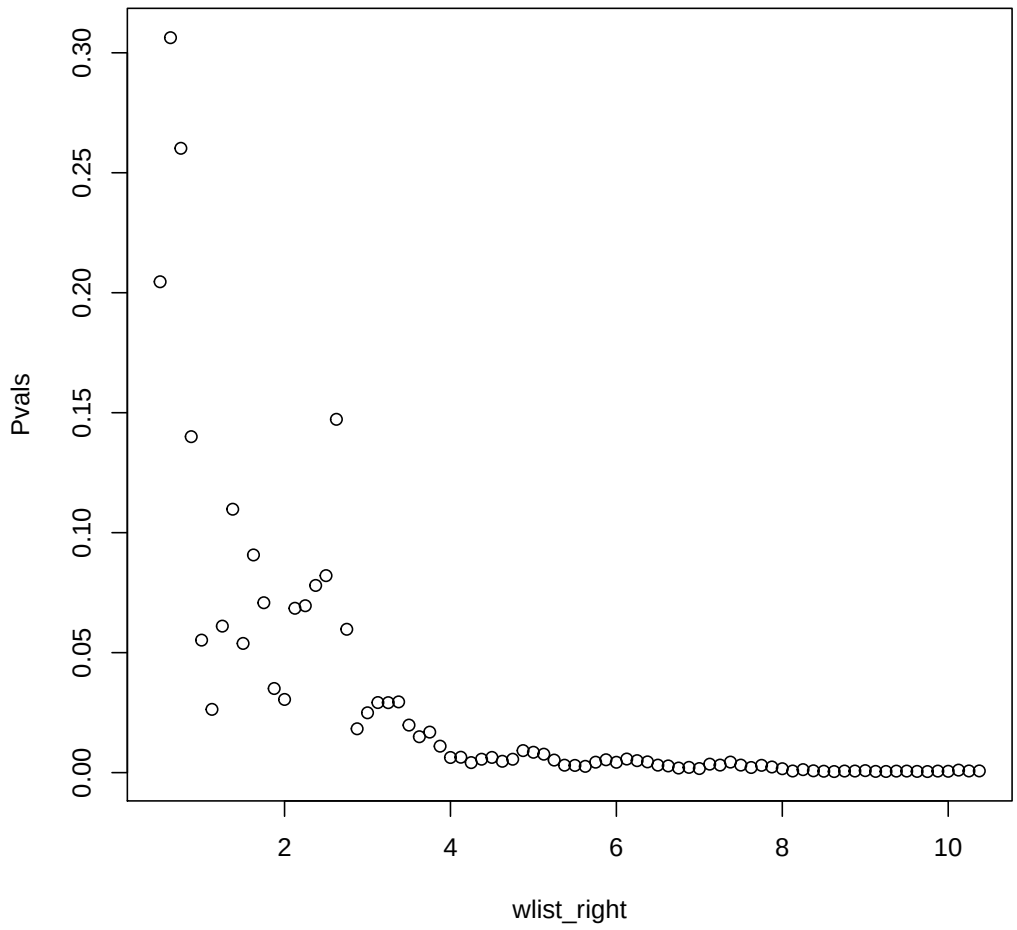

Figure 1. Plot of p-values.

The figure shows that the p-values vary widely for very short windows, but the sequence stabilizes once the window length is large enough (around the value 3 in this case).

Once the window has been selected, randomization inference to test the sharp null hypothesis of no treatment effect can be performed using `rdrandinf`. For example, take the window $[-0.75\,;\,0.75]$, which is the one selected by Cattaneo et al. [2015] and replicated above. The basic syntax for `rdrandinf` is:

```
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75)

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs      =          1297
Order of poly      =             0
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =             0
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                Finite sample            Large sample
                              ------------------  ----------------------------
          Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.300
================================================================================
```

Like `rdwinselect`, the output of `rdrandinf` is divided in three panels. The upper panel gives the total sample size, the order of the polynomial, the type of kernel used for the weighting scheme (`uniform`, `triangular` or `epan`), the number of replications in the randomization test, and whether the window was specified by the user by setting `wl` and `wr` or calculated using `rdwinselect` as will be illustrated shortly. The middle panel provides the number of observations at each side of the cutoff, sample size below and above the cutoff inside the specified window, some descriptive statistics for the outcome inside the window, and the selected window. Note that the first line in this panel displays the number of observations with non-missing values of the outcome and running variable, so the sample sizes shown can differ from the total sample size.

Finally, the main panel gives the results from the randomization test. The first column, labeled "Statistic", indicates the statistic used in the randomization test. The second column gives the observed value of the selected statistic and the third column shows its finite-sample p-value obtained from the randomization test. The fourth column gives the asymptotic p-value, that is, the p-value obtained from the corresponding asymptotic distribution of the chosen statistic. Finally the fifth

column gives the asymptotic power against an alternative value that can be specified using the options `d()` or `dscale()`. The default is `dscale(.5)`, that is, an effect size equal to half the standard deviation of the outcome for the control group inside the window (the critical value for the power calculation is set to 1.96).

As mentioned above, `rdrandinf` uses the difference in means as the default statistic, but it can also use the Kolmogorov-Smirnov and the rank sum statistics. By adding `statistic(all)` as an option we can obtain the result for all three statistics. The output is:

```
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75,statistic='all')

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs      =          1297
Order of poly      =             0
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =             0
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                Finite sample            Large sample
                              ------------------  ----------------------------
         Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.300
 Kolmogorov-Smirnov      0.552        0.008        0.005                      NA
    Rank sum z-stat     -3.217        0.000        0.001                   0.209
================================================================================
```

We can see that the three statistics provide basically the same result in terms of inference; the randomization test rejects the sharp null of no treatment effect at one percent significance level in all three cases. Also note that the `rdrandinf` command does not provide the asymptotic power for the Kolmogorov-Smirnov statistic.

The window in which to perform the randomization-based tests can be set manually using `wl` and `wr`. This options specify the lower and upper limits of the chosen window. Importantly, these are window limits and not lengths, so for instance, if the cutoff is 100 and the user wants a window of ± 5, the correct syntax is `wl` $= 95$ and `wr` $= 105$. We advise the user to always normalize the cutoff to zero by centering the running variable to avoid confusion.

Alternatively, the user can specify the list of covariates to have `rdrandinf` select the window automatically using `rdwinselect`. All the options allowed in `rdwinselect` can be passed through `rdrandinf`. For example:

```
> tmp <- rdrandinf(Y,R,statistic='all',covariates=X,wmin=.5,wstep=.125,rdwreps=10000)

Running rdwinselect...
Mass points detected in running variable

rdwinselect complete.

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs     =          1297
Order of poly     =             0
Kernel type       =       uniform
Reps              =          1000
Window            =   rdwinselect
H0:          tau  =             0
Randomization     = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                               Finite sample            Large sample
                             ------------------  ----------------------------
         Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.300
 Kolmogorov-Smirnov      0.552        0.006        0.005                      NA
    Rank sum z-stat     -3.217        0.000        0.001                   0.209
================================================================================
```

The `rdrandinf` command allows the user to specify a polynomial transformation model for the outcomes using the option `p`. By default, the command sets `p=0`, which means no transformation. When the `p` is set to an integer larger than zero, the slopes (and possibly higher order terms) are subtracted from the outcomes, leaving a residualized version of the outcome that only differs above and below the cutoff in the intercept. For instance, to perform a linear transformation, the syntax is:

```
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75,statistic='all',p=1)

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs      =          1297
Order of poly      =             1
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =             0
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                Finite sample            Large sample
                               ------------------  ------------------------------
          Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means     15.297        0.000        0.066                   0.071
 Kolmogorov-Smirnov      0.797        0.000        NA                         NA
    Rank sum z-stat     -4.455        0.000        NA                         NA
================================================================================
```

When a model for the outcomes is specified—that is, when `p` is set to a number greater than zero—with the option `statistic="ttest"` fits a regression of the outcome on the treatment dummy interacted with a polynomial of the running variable, and uses the difference in intercepts as the test-statistic. The other test-statistics use as outcomes the residuals described above. Note that the command does not provide the asymptotic p-value nor the asymptotic power of the Kolmogorov-Smirnov and rank sum statistics, as the asymptotic distribution does not account for the model

transformation and hence can be misleading.

In the presence of arbitrary interference, a confidence interval for a particular measure of the effects of the program can be obtained with the `interfci` option. For example, to obtain a 95 percent confidence interval, we type:

```
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75,interfci=.05)

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs      =          1297
Order of poly      =             0
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =             0
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                Finite sample            Large sample
                               -----------------  ----------------------------
          Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.300
================================================================================

95% confidence interval under interference: [3.795;15.824]
```

In terms of interpretation, it is important to keep in mind that the confidence interval under interference is not a confidence interval for the point estimate (and in fact, it may even not contain the point estimate). The interference confidence interval is constructed based on the difference between the observed statistic and the statistic that would be observed if the treatment was withheld from all units. In our application, allowing for arbitrary interference we can say with 95 percent confidence that the "excess" benefit of the treated group compared to the control group is between 3.795 and 15.824. Again, in this particular example the point estimate under SUTVA happens to

be contained in the confidence interval under interference, but this need not be the case and has no clear interpretation.

The `rdlocrand` package provides two types of sensitivity analyses to assess how p-values change with window length. The first one, `rdsensitivity`, calculates and plots a matrix of p-values over a range of values for the treatment effect under the null hypothesis (rows) and window lengths (columns). For instance, we can see how the p-values change by starting from the selected window, increasing the window length by 0.25 and over a range of treatment effects that is roughly the point estimate plus and minus 10:

```
> tmp <- rdsensitivity(Y,R,wlist=seq(.75,10,by=.25),tlist=seq(0,20,by=1))

Running sensitivity analysis...

Sensitivity analysis complete.
```

Note that the `rdsensitivity` command does not display any output, but its results can be called using `tmp$results`. In addition to the p-values, the `rdsensitivity` command returns the plot shown in Figure 2. The plot depicts the grid of window lengths in the horizontal axis and the grid of treatment effects under the null. The color represents the p-value for each pair of window length and treatment effect, where white corresponds to zero and black corresponds to one. This is simply a graphical display of the results given by `rdsensitivity`. The plot can be replicated (or modified) with the following code:

```
> xaxis <- tmp$wlist
> yaxis <- tmp$tlist
> zvalues <- tmp$results
> filled.contour(xaxis,yaxis,t(zvalues),
+                xlab='window',ylab='treatment effect',
+                key.title=title(main = 'p-value',cex.main=.8),
+                levels=seq(0,1,by=.01),col=gray.colors(100,1,0))
```

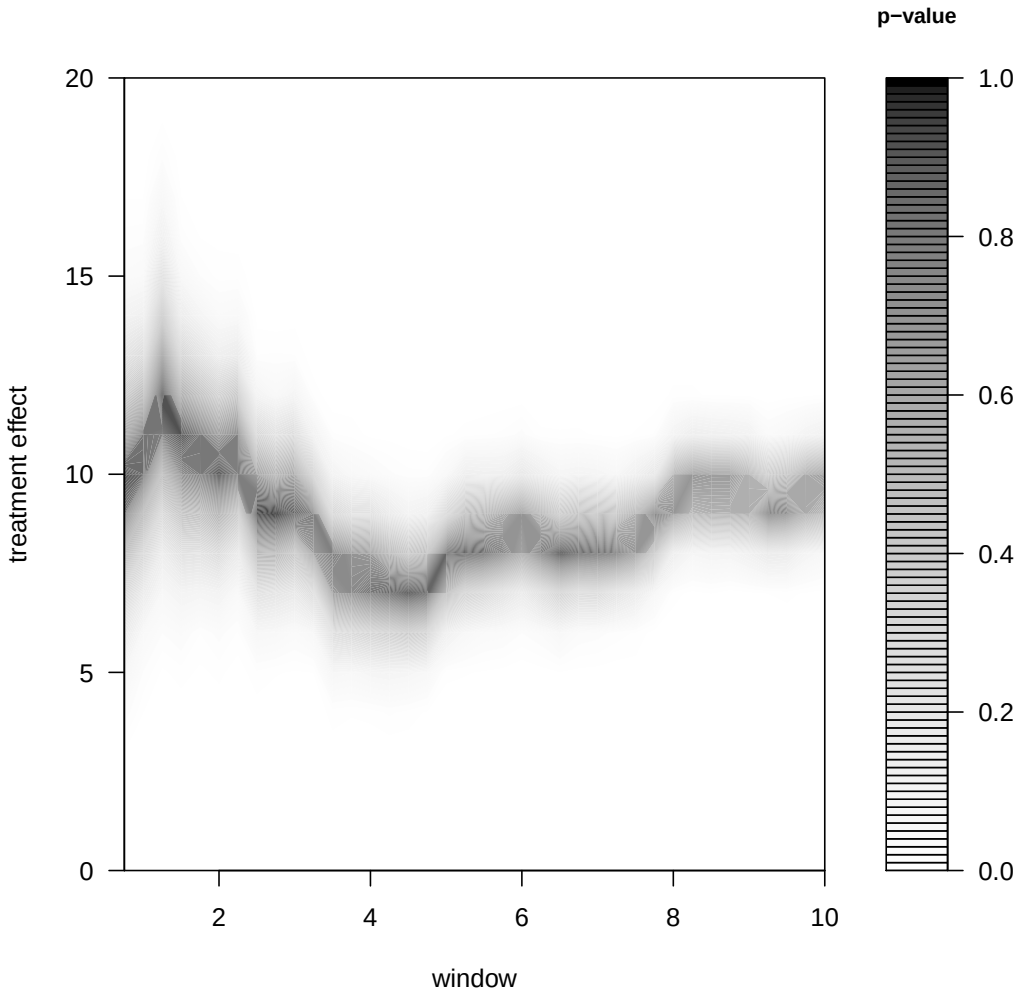

Figure 2. Sensitivity analysis.

One way to interpret these results is to see the range of values for which the p-value is above, say, .05, as a 95 percent confidence interval for the point estimate (assuming a constant additive treatment effect). Thus, the above table shows how the confidence interval for the treatment effect changes as the window length increases. For instance, the 95 percent confidence interval for the window $[-.75\,;\,.75]$ is roughly $[5\,;\,14]$, whereas for the window $[-2\,;\,2]$ it becomes $[7\,;\,14]$. In this case, the point estimate seems to be relatively stable over the range of windows considered. The confidence interval for the window $[-.75\,;\,.75]$ can be obtained using the `ci` option:

```
> tmp <- rdsensitivity(Y,R,wlist=seq(.75,2,by=.25),tlist=seq(0,20,by=1),ci=c(-0.75,0.75))

Running sensitivity analysis...
Sensitivity analysis complete.
> tmp$ci
     [,1] [,2]
```

```
[1,]    5   14
```

Additionally, `rdsensitivity` can be called from within `rdrandinf` to obtain confidence intervals for the point estimates obtained using the `ci` option. The syntax is the following:

```
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75,ci=c(.05,seq(3,20,by=1)))

Selected window = [-0.75;0.75]

Running randomization-based test...
Randomization-based test complete.

Running sensitivity analysis...
Sensitivity analysis complete.


Number of obs      =          1297
Order of poly      =             0
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =         0.000
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                Finite sample            Large sample
                               ------------------  ------------------------------
          Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.300
================================================================================

95% confidence interval: [5,14]
```

The second type of sensitivity analysis is performed with `rdrbounds`, which calculates Rosenbaum bounds [Rosenbaum, 2002]. As explained above, this command calculates upper and lower bounds for the randomization p-value under Bernoulli trials for a range of values of a parameter $\Gamma \equiv \exp(\gamma)$ that captures the strength with which an unobservable binary variable U_i affects the probability of selection into treatment. The basic syntax is:

```
> tmp <- rdrbounds(Y,R,expgamma=c(1.5,2,3),wlist=c(.5,.75,1),reps=1000)
```

```
Calculating randomization p-value...

Bernoulli p-value (w = 0.5) = 0.01
Bernoulli p-value (w = 0.75) = 0
Bernoulli p-value (w = 1) = 0

Running sensitivity analysis...

Sensitivity analysis complete.
> tmp$lower.bound
            [,1]  [,2] [,3]
[1,] 0.006000000 0.000    0
[2,] 0.002000000 0.001    0
[3,] 0.002004008 0.000    0
> tmp$upper.bound
      [,1]  [,2]  [,3]
[1,] 0.027 0.013 0.003
[2,] 0.088 0.058 0.016
[3,] 0.262 0.280 0.162
```

The output from `rdrbounds` is divided in several parts. The first one shows the randomization p-value based on Bernoulli trials for each window. The matrices `$lower.bound` and `$upper.bound` present the lower and upper bounds for the p-values for different values of γ and windows. The wider the distance between the lower and upper bounds, the more sensitive the inference to deviations from a randomized experiment. The remaining elements give the list of values used to calculate the bounds.

The `fmpval` option adds the fixed margins randomization p-value to the first panel of the output. This allows the user to compare the p-values obtained using each method.

```
> tmp <- rdrbounds(Y,R,expgamma=c(1.5,2,3),wlist=c(.5,.75,1),reps=1000,fmpval=TRUE)

Calculating randomization p-value...

Bernoulli p-value (w = 0.5) = 0.01
Fixed margins p-value (w = 0.5) = 0.004
Bernoulli p-value (w = 0.75) = 0
Fixed margins p-value (w = 0.75) = 0
Bernoulli p-value (w = 1) = 0
Fixed margins p-value (w = 1) = 0.001

Running sensitivity analysis...

Sensitivity analysis complete.
> tmp$lower.bound
            [,1]  [,2] [,3]
[1,] 0.006000000 0.000    0
```

```
[2,] 0.002000000 0.001    0
[3,] 0.002004008 0.000    0
> tmp$upper.bound
      [,1]  [,2]  [,3]
[1,] 0.027 0.013 0.003
[2,] 0.088 0.058 0.016
[3,] 0.262 0.280 0.162
```

We can see that the p-values obtained by both methods are very similar, which we found to be usually true in applications as long as the number of replications is large enough.

Finally, when using outcome transformation, the `rdrandinf` command allows the user to choose in which point to evaluate the transformed outcomes. By default, the evaluation point is the cutoff, which emulates the idea used in the local polynomial approach of estimating the effect at the cutoff. However, whenever the local randomization assumption is plausible, the cutoff need not be the point of interest. For instance, to set the evaluation points at the means of the running variable inside the window below and above the cutoff, we can type:

```
> ii <- (R>=-.75) & (R<=.75) & !is.na(Y) & !is.na(R)
> m0 <- mean(R[ii & D==0],na.rm=TRUE)
> m1 <- mean(R[ii & D==1],na.rm=TRUE)
> tmp <- rdrandinf(Y,R,wl=-.75,wr=.75,p=1,evall=m0,evalr=m1)

Selected window = [-0.75;0.75]

Running randomization-based test...

Randomization-based test complete.


Number of obs      =          1297
Order of poly      =             1
Kernel type        =       uniform
Reps               =          1000
Window             =   set by user
H0:          tau   =             0
Randomization      = fixed margins

Cutoff c =    0.000   Left of c  Right of c
      Number of obs         595         702
 Eff. number of obs          15          22
    Mean of outcome      42.808      52.497
    S.d. of outcome       7.042       7.742
             Window      -0.750       0.750

================================================================================
                                 Finite sample           Large sample
                               ------------------  ----------------------------
```

```
         Statistic          T        P>|T|        P>|T|    Power vs d =   3.521
================================================================================
     Diff. in means      9.689        0.000        0.000                   0.283
================================================================================
```

The user can verify that the point estimate in this case is the same as when no transformation is used, which is due to the fact that the transformation comes from a linear regression which by construction passes through the means. The p-values, however, can differ. Incidentally, note that the means are taken over the sample inside the window with non-missing values for the outcome and the running variable. The reason is that `rdrandinf` drops the observations inside the window with missing outcomes of running variable. Similarly, `rdwinselect` drops, at each evaluated window, the observations with missing values of the covariates and running variable.

3 Auxiliary Functions

This section introduces the auxiliary functions used by the four functions described above. All the auxiliary functions are included in the file `rdlocrand_fun`. We do not recommend to use these functions directly, as they typically do not contain error handling and incorrectly using them may lead to unexpected mistakes.

3.1 Statistics for randomization inference

```
rdrandinf.model(Y,D,statistic,pvalue=FALSE,kweights,endogtr,delta="")
```

This function uses the outcome and treatment variable $D = \mathbf{1}(R \geq c)$ to calculate the observed statistics, asymptotic p-values and asymptotic power for all the statistics used by `rdrandinf`, namely, `"ttest"`, `"ksmirnov"`, `"ranksum"`, `"all"`, `"ar"` and `"wald"`.

3.2 Hotelling's T^2 statistic

```
hotelT2(X,D)
```

This function computes Hotelling's T^2 statistic and its asymptotic p-value for the matrix of covariates `X` for `rdwinselect`.

3.3 List of windows

```
findowbs(wobs,nwin,posl,posr,R,dups)
```

This function finds the windows containing `wobs` observations at each side of the cutoff. It is used to find the default list of windows in `rdwinselect`.

3.4 List of symmetric windows

`findowbs_sym(wobs,nwin,posl,posr,R,dups)`

This function is similar to `findowbs()` but symmetrizes the window around the cutoff.

3.5 Randomization inference confidence interval

`findCI(pvals,alpha,tlist)`

This function returns the randomization inference confidence interval using `rdsensitivity`.

References

Matias D. Cattaneo, Brigham Frandsen, and Rocio Titiunik. Randomization inference in the regression discontinuity design: An application to party advantages in the u.s. senate. *Journal of Causal Inference*, 3(1):1–24, 2015.

Matias D. Cattaneo, Rocio Titiunik, and Gonzalo Vazquez-Bare. Inference in regression discontinuity designs under local randomization. *Stata Journal*, 16(2):331–367, 2016.

Matias D. Cattaneo, Rocio Titiunik, and Gonzalo Vazquez-Bare. Comparing inference approaches for rd designs: A reexamination of the effect of head start on child mortality. *Journal of Policy Analysis and Management*, 36(3):643–681, 2017.

Matias D. Cattaneo, Nicolás Idrobo, and Rocío Titiunik. *A Practical Introduction to Regression Discontinuity Designs: Extensions*. Cambridge Elements: Quantitative and Computational Methods for Social Science, Cambridge University Press (to appear), 2020.

Paul R. Rosenbaum. *Observational Studies*. Springer, New York, 2002.