Using PanelMatch

In Song Kim, Adam Rauh, Erik Wang, Kosuke Imai

2022-06-21

Overview

The goal of this vignette is to provide a quick overview of using the `PanelMatch` package, highlight important features and functions, and help users get the most out of their experience with the package. It assumes that you have been able to install the package successfully and are already familiar with the basics of R.

We will be working with the `dem` data set throughout these examples, which comes included with the package. This is a subset of the data used in Acemoglu et. al's "Democracy Does Cause Growth" (2019).

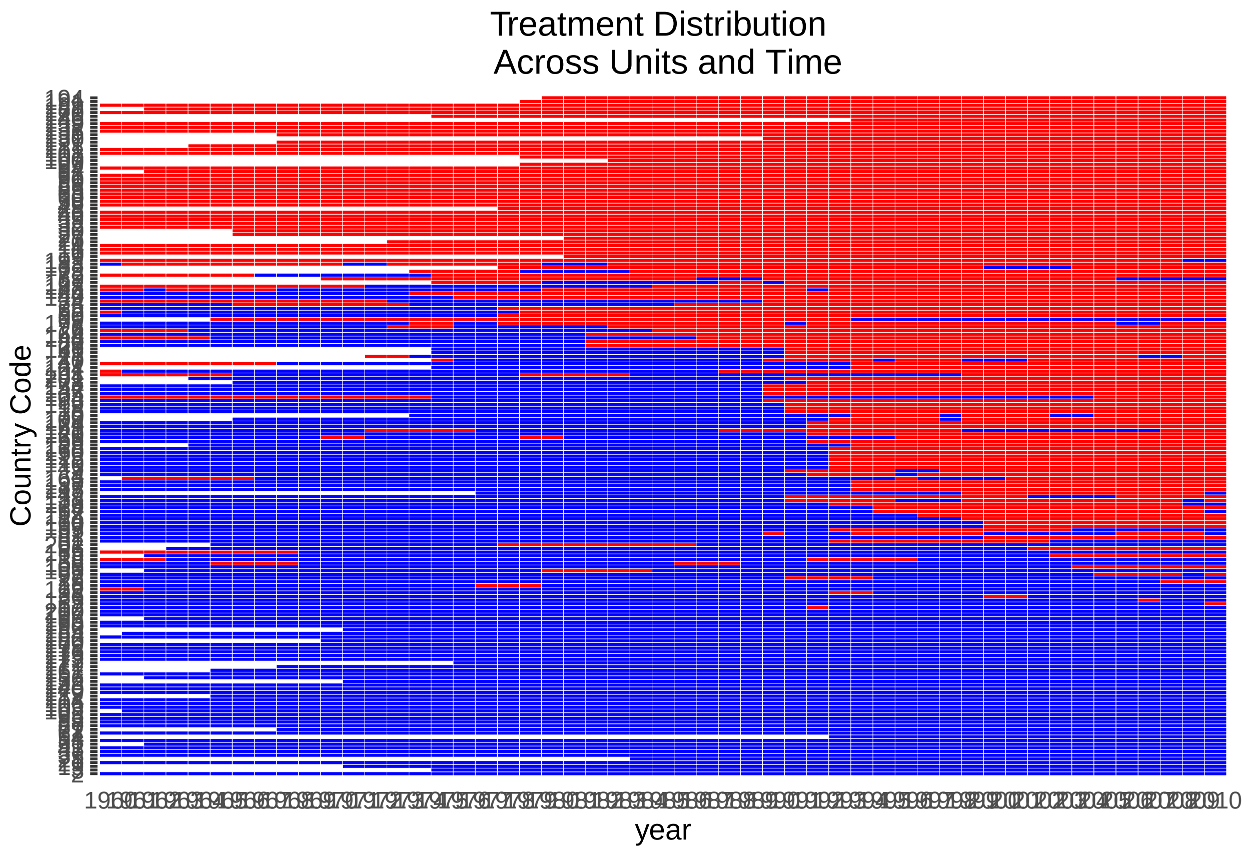
Assumptions and Requirements

The package is designed to implement a set of methodological tools that enable researchers to apply matching methods to time-series cross-sectional data, so it is assumed that the data set being used for analysis meets this general structure. Additionally, the treatment variable must be binary, where 0 means "assignment" to control, and 1 to treatment. We also impose the requirements that the variable identifying units in the data are numeric or integer, and that the variable identifying time periods are consecutive numeric/integer data. The data must also be provided in the form of a standard `data.frame` object (as opposed to a `data.table`, `tibble` or other such object). The package will signal an error if these assumptions are violated.

DisplayTreatment

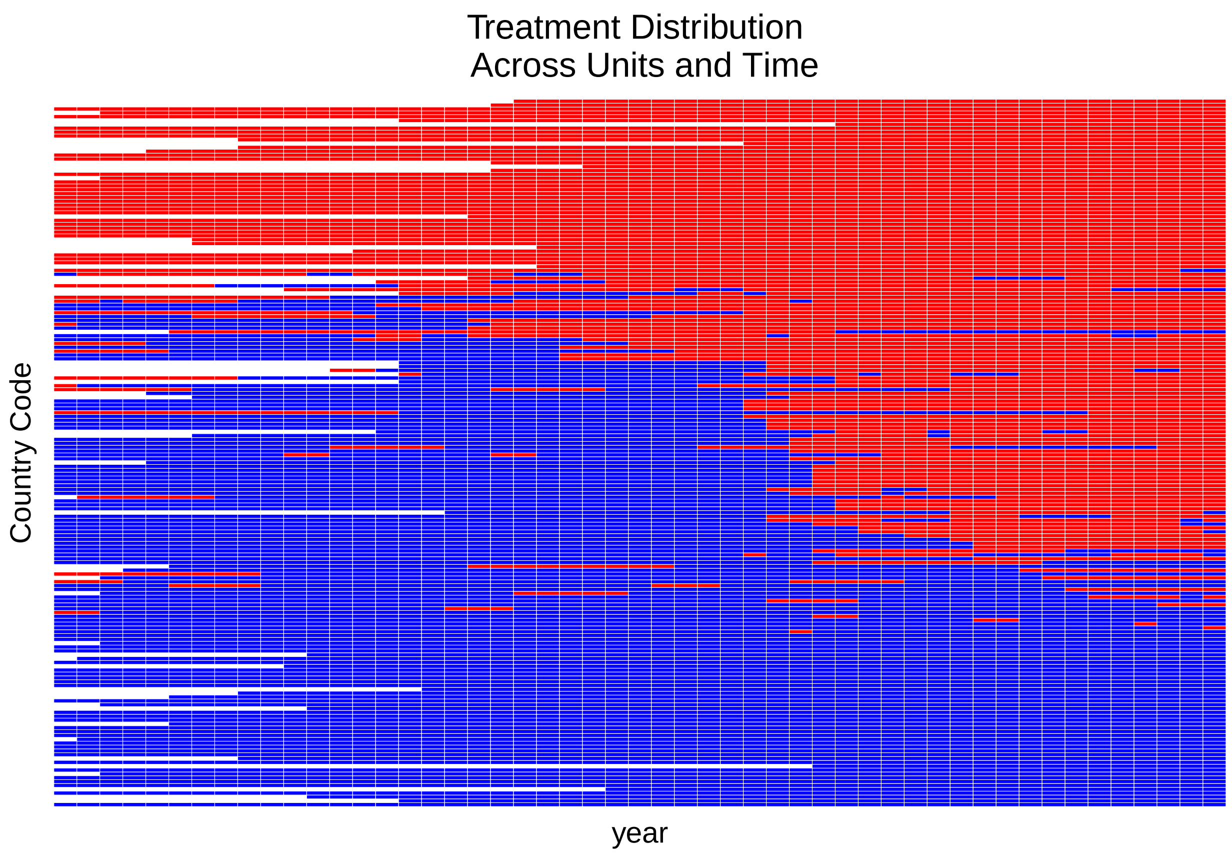
Before doing any matching, estimation or further analysis, it is helpful to understand the distribution of the treatment variable within your data set. The package hopes to facilitate this with the `DisplayTreatment` function.

```r
library(PanelMatch)
DisplayTreatment(unit.id = "wbcode2",
                 time.id = "year", legend.position = "none",
                 xlab = "year", ylab = "Country Code",
                 treatment = "dem", data = dem)
```

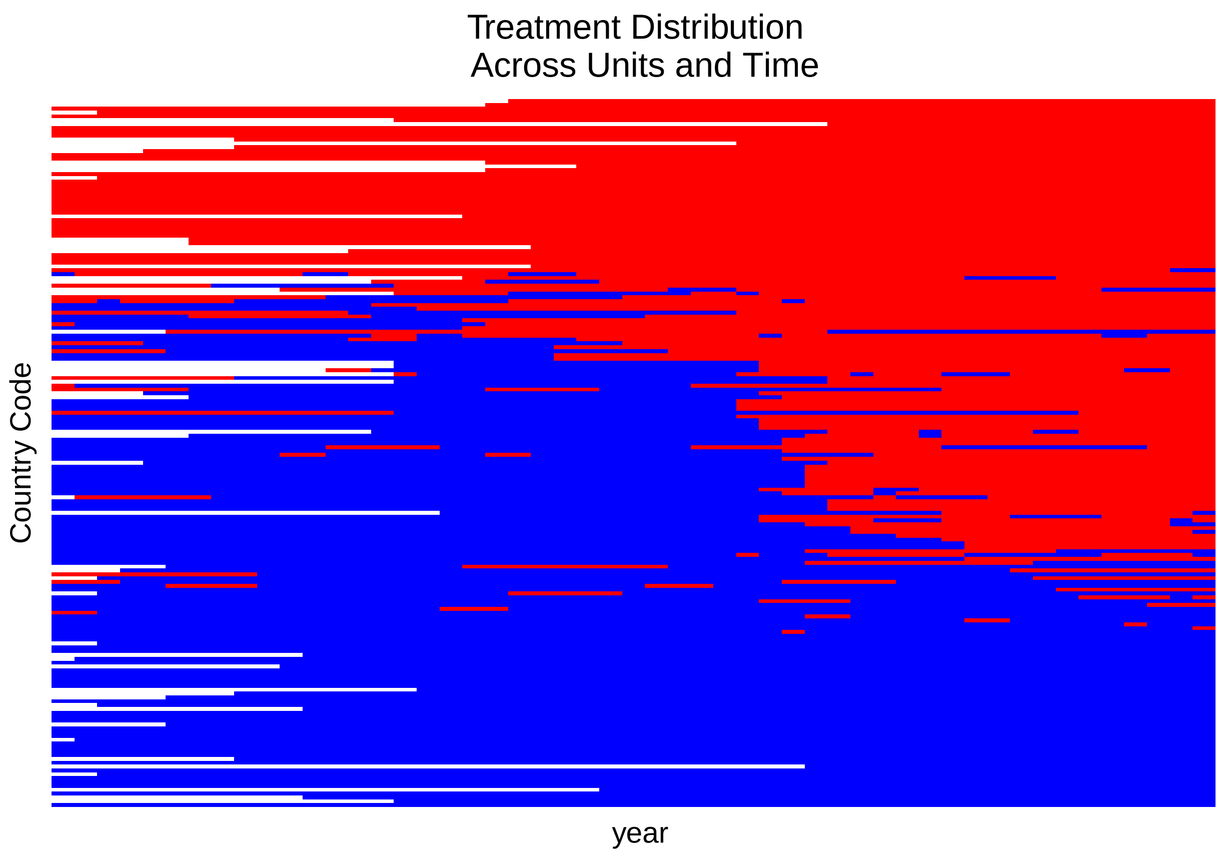


In the plot, the x axis represents time, and the y axis displays the different units in your data set. Red tiles indicate periods where "treatment" is applied to a given unit and blue tiles indicate "control" periods. White spaces indicate missing data. In the above plot, we have a large number of unique units and time periods, so the axes become hard to read. The `DisplayTreatment` function uses `ggplot2` to create this plot, so any custom styling can be applied easily using `ggplot2` conventions, as the function returns a `ggplot2` object. However, the function also has some built in options to make cleaning up the plot a little easier. For instance, when the data set is particularly large, it can help to use the `dense.plot` option. There are many ways to customize these plots. Consult the documentation for the full list and descriptions of the arguments.

```
DisplayTreatment(unit.id = "wbcode2",
                 time.id = "year", legend.position = "none",
                 xlab = "year", ylab = "Country Code",
                 treatment = "dem", data = dem,
                 hide.x.tick.label = TRUE, hide.y.tick.label = TRUE) # axis label options
```



```r
DisplayTreatment(unit.id = "wbcode2",
                 time.id = "year", legend.position = "none",
                 xlab = "year", ylab = "Country Code",
                 treatment = "dem", data = dem,
                 hide.x.tick.label = TRUE, hide.y.tick.label = TRUE,
                 dense.plot = TRUE) # setting dense.plot to TRUE
```



PanelMatch

Next, we will move to the `PanelMatch` function. The primary purposes of this function are to 1) use the treatment histories of units to create sets matching treated units to control units and 2) refine the matched sets by determining weights for each control unit in a given matched set. These weights are then used in the estimation stage in an intuitive way: units with higher weights factor more heavily into the estimations. There are a number of parameters to the `PanelMatch` function, some of which will not be discussed in this vignette. Please consult the function documentation for a complete list.

Matching on Treatment History

1 is achieved by matching units that receive treatment after previously being untreated (ie. units that move from `control`, or 0 in the data, to `treatment`, or 1 in the data, at a certain time) to control units that have matching treatment histories in a specified time window, while also remaining untreated during the same period that the treated unit receives treatment. For example, if unit 4 is a control unit until the year 1992, when it then receives treatment, then, for a specified lag window of four time periods (specified as `lag = 4`), it will be matched with control units that share an identical treatment history with unit 4 from 1988-1991, while also *remaining control units* in 1992.

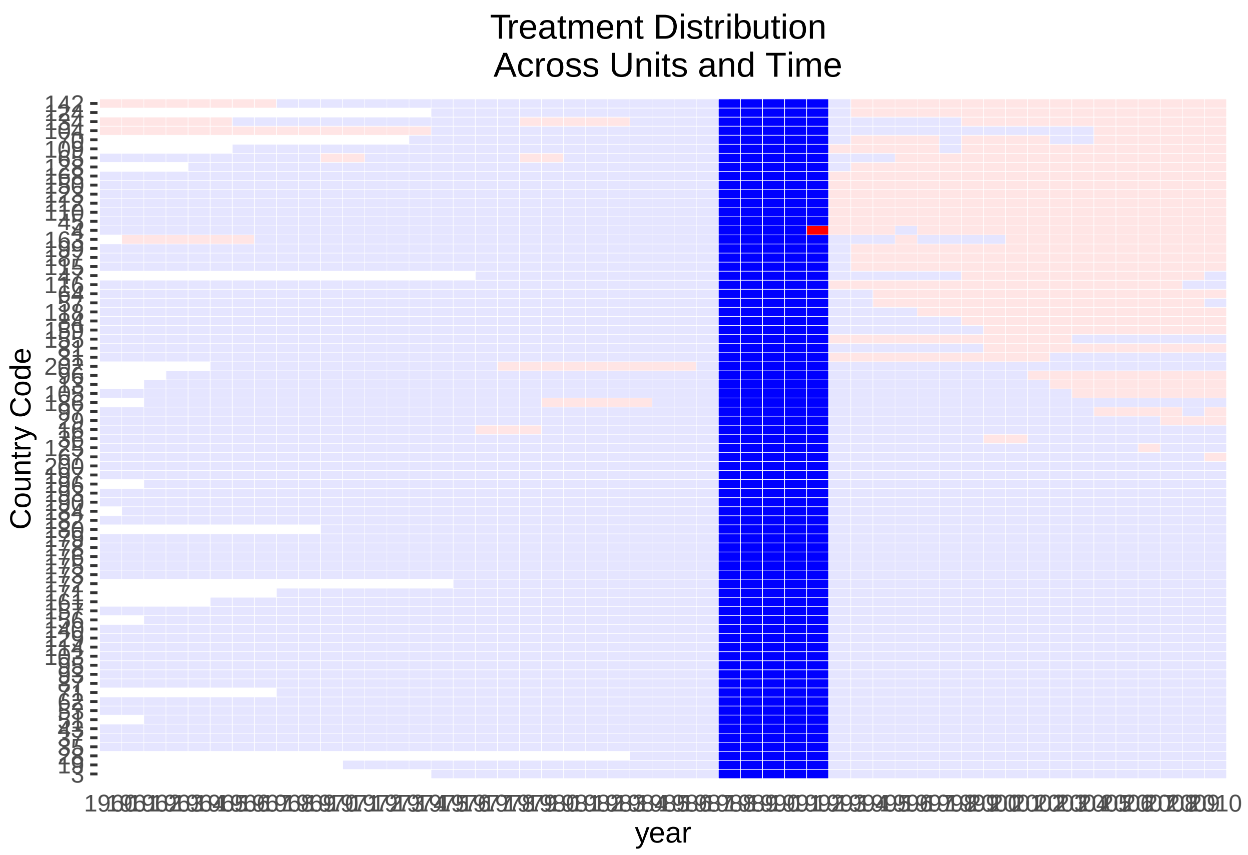
Setting the Quantity of Interest This process of matching on treatment history is also affected by the specified quantity of interest, which is specified using the `qoi` parameter, set to either "att", "atc", "art", or "ate". See Imai, Kim, and Wang (2021) http://web.mit.edu/insong/www/pdf/tscs.pdf for full descriptions. When the `qoi` is set to "att", control units have `0` in the treatment variable column, and treated units have `1`. The process is flipped when the `qoi` is set to `atc` and `art`. When `ate` is specified as the `qoi`, sets are computed for both the `att` and `atc` setups.

Visualize Treatment History Matching

We can use the `DisplayTreatment` function to help better understand this definition and the example given above. In the code below, we will create matched sets as described in **1**, extract one of these sets, and then show the treated unit and matched control units from this set on a plot. We will cover the `PanelMatch` function in more detail later, so do not worry about understanding all of the arguments yet. For now, know that the lag window is specified using the `lag` variable and that we are not refining the matched sets in any other way, aside from the description in **1** (`refinement.method` = "none").

The visualization lines up with our expectations. We see that the control units and the treated unit have identical treatment histories over the lag window (1988-1991). Unit 4 then receives treatment in 1992 and the other units shown do not.

```r
# Create the matched sets
PM.results.none <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "none",
                         data = dem, match.missing = TRUE,
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)
# Extract the first matched set
mset <- PM.results.none$att[1]

# Use the DisplayTreatment function to visualize the
# treated unit and matched controls.
DisplayTreatment(unit.id = "wbcode2",
                 time.id = "year", legend.position = "none",
                 xlab = "year", ylab = "Country Code",
                 treatment = "dem", data = dem,
                 matched.set = mset, # this way we highlight the particular set
                 show.set.only = TRUE)
```



Refining Matched Sets

In order to refine the matched sets further and assign weights to control units, the user must specify 1) a method for calculating similarity/distance between units and 2) the variables to be used in similarity/distance calculations to compare units. Other parameters do affect the refinement process, and the user should consult the documentation for a more complete list and set of descriptions.

Select a Refinement Method Users must choose a refinement method by setting the `refinement.method` argument to one of “mahalanobis”, “ps.match”, “CBPS.match”, “ps.weight”, “CBPS.weight”, “ps.msm.weight”, “CBPS.msm.weight”, or “none”. The “matching” (any method that has “match” in the name) and mahalanobis refinement methods will assign equal weights to the `size.match` most similar control units in a matched set. The “weighting” methods (methods with “weight” in the name) will generate weights in such a way that control units more similar to treated units will be assigned higher weights. Please consult the documentation and/or Imai, Kim, and Wang (2021) http://web.mit.edu/insong/www/pdf/tscs.pdf for descriptions of each refinement method.

Variable Selection Users must also define which covariates should be used in this process for defining similarity between units. This is set using the `covs.formula` argument, which takes the form of a one sided formula object. The variables defined on the right hand side of the formula are the variables used in these calculations. Users can included “lagged” versions of variables using `I(lag(name.of.var, 0:n))`.

We will now turn to understanding how to work with the `PanelMatch` function and the objects it returns to execute and tune these processes.

Understanding `PanelMatch` and `matched.set` objects

The `PanelMatch` function returns a `PanelMatch` object. These objects contain a number of elements, the most important of which is the `matched.set` object. Within the `PanelMatch` object, the attached `matched.set` object is always named either `att`, `art`, or `atc`. It should be noted that when `qoi = ate`, then there are two `matched.set` objects included in the results of the `PanelMatch` call. Specifically, there will be two matched sets, named `att` and `atc`, respectively.

In implementation, the `matched.set` is just a named list with some added attributes (lag, names of treatment, unit, and time variables) and a structured name scheme. Each entry in the list corresponds to a matched set of treated and control units. Specifically, the names of the elements correspond to the unit and time identifiers of the treated units. This naming scheme is structured, using the pattern `[id varable].[time variable]`. Each element in the list is a vector indicating the control units (as a vector of the unit ids) that are matched with the treated unit specified in the name of that element.

Recall earlier we used a very basic `PanelMatch` call that did no unit refinement other than matching units on treatment history. We create an additional `PanelMatch` call, specifying that we want to use Mahalanobis distance and lagged versions of the `tradewb` and `y` variables to calculate the distances between units. Because this is a matching method, we will only give weights to the `size.match` most similar control units to each treated unit as determined by the distance calculations.

```r
#Call PanelMatch without any refinement
PM.results.none <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "none",
                         data = dem, match.missing = TRUE,
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)
#Extract the matched.set object
msets.none <- PM.results.none$att

#Call PanelMatch with refinement
PM.results.maha <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "mahalanobis",
                         # use Mahalanobis distance
                         data = dem, match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att" , outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)

msets.maha <- PM.results.maha$att
```

Examining `matched.set` objects

`print`, `summary`, and `plot`

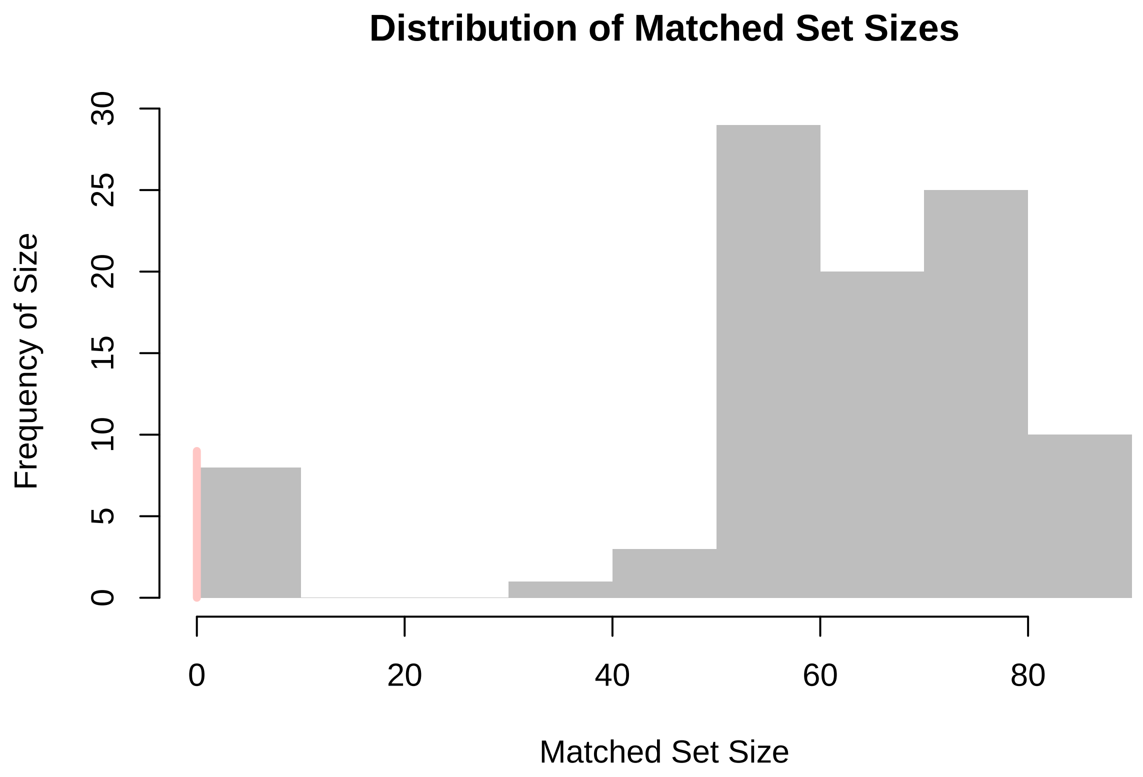
Methods for printing, summarizing, and plotting `matched.set` objects are implemented. We can print basic information about the matched sets using the `print` method. The output will show information about the treated units and the number of control units in each matched set. *Note that this does not consider the weights of control units within each set. The matched set sizes shown in the output of `print.matched.set` will show the number of controls matched to a treated unit based on treatment history only.* Matched set objects are implemented as lists, but the default printing behavior resembles that of a data frame. One can toggle a `verbose` option on the `print` method to print as a list and also display a less summarized version of the matched set data.

Despite the refinement applied to the `msets.maha` object, the matched set sizes appear identical because the weights of control units are not considered in this view – just treatment history. Users are able to view more information about these objects with the `summary` function.

```
print(msets.none)
#>     wbcode2 year matched.set.size
#> 1         4 1992               74
#> 2         4 1997                2
#> 3         6 1973               63
#> 4         6 1983               73
#> 5         7 1991               81
#> 6         7 1998                1
#> 7        12 1992               74
#> 8        13 2003               58
#> 9        15 1991               81
#> 10       16 1977               63
#> 11       17 1991               81
#> 12       18 1991               81
#> 13       22 1991               81
#> 14       25 1982               72
#> 15       26 1985               76
#> 16       31 1993               65
#> 17       34 1990               79
#> 18       36 2000               57
#> 19       38 1992               74
#> 20       40 1990               79
#> 21       40 1996                1
#> 22       40 2002                1
#> 23       41 1991               81
#> 24       45 1993               65
#> 25       47 1999               59
#> 26       50 1978               63
#> 27       52 1979               60
#> 28       55 1978               63
#> 29       56 1992               74
#> 30       57 1995               57
#> 31       59 1990                1
#> 32       64 1995               57
#> 33       65 1970               58
#> 34       65 1979               60
#> 35       65 1996               57
#> 36       70 1994               58
#> 37       70 2005                1
#> 38       72 1975               61
#> 39       73 1984               74
#> 40       75 1966               46
#> 41       75 1986               77
#> 42       78 1992               74
#> 43       80 1982               72
#> 44       81 2000               57
#> 45       82 2006               49
#> 46       83 1990               79
#> 47       84 1999               59
#> 48       96 2002               58
#> 49       97 2005               54
#> 50      101 1988                80
#> 51      104 2005                54
#> 52      105 2004                57
#> 53      109 1993                65
#> 54      110 1993                65
#> 55      112 1993                65
#> 56      115 1994                58
#> 57      116 1993                65
#> 58      118 1997                58
#> 59      119 1991                81
#> 60      120 1992                74
#> 61      123 1993                65
#> 62      124 1994                58
#> 63      128 1994                58
#> 64      133 1991                81
#> 65      134 1979                60
#> 66      134 1999                59
#> 67      135 1990                79
#> 68      138 1991                81
#> 69      138 2006                49
#> 70      141 1972                63
#> 71      141 1988                80
#> 72      142 1994                58
#> 73      143 1980                61
#> 74      144 1987                78
#> 75      149 1976                63
#> 76      150 1993                65
#> 77      154 1990                79
#> 78      155 1993                65
#> 79      158 1965                40
#> 80      158 1986                77
#> 81      159 2000                57
#> 82      162 2004                57
#> 83      163 1996                57
#> 84      163 2001                59
#> 85      164 1982                72
#> 86      167 1988                80
#> 87      168 1993                65
#> 88      169 1992                74
#> 89      177 1974                62
#> 90      177 1978                 1
#> 91      183 1983                 2
#> 92      187 1994                58
#> 93      188 1985                76
#> 94      199 1994                58
#> 95      201 1991                81
#> 96      202 1978                63
#> 97       17 2009                 0
#> 98       70 1999                 0
#> 99       82 1994                 0
#> 100     109 1999                 0
#> 101     133 1999                 0
#> 102     143 1993                 0
#> 103     167 1991                0
#> 104     177 1992                0
#> 105     183 1973                0
print(msets.maha)
#>     wbcode2 year matched.set.size
#> 1         4 1992               74
#> 2         4 1997                2
#> 3         6 1973               63
#> 4         6 1983               73
#> 5         7 1991               81
#> 6         7 1998                1
#> 7        12 1992               74
#> 8        13 2003               58
#> 9        15 1991               81
#> 10       16 1977               63
#> 11       17 1991               81
#> 12       18 1991               81
#> 13       22 1991               81
#> 14       25 1982               72
#> 15       26 1985               76
#> 16       31 1993               65
#> 17       34 1990               79
#> 18       36 2000               57
#> 19       38 1992               74
#> 20       40 1990               79
#> 21       40 1996                1
#> 22       40 2002                1
#> 23       41 1991               81
#> 24       45 1993               65
#> 25       47 1999               59
#> 26       50 1978               63
#> 27       52 1979               60
#> 28       55 1978               63
#> 29       56 1992               74
#> 30       57 1995               57
#> 31       59 1990                1
#> 32       64 1995               57
#> 33       65 1970               58
#> 34       65 1979               60
#> 35       65 1996               57
#> 36       70 1994               58
#> 37       70 2005                1
#> 38       72 1975               61
#> 39       73 1984               74
#> 40       75 1966               46
#> 41       75 1986               77
#> 42       78 1992               74
#> 43       80 1982               72
#> 44       81 2000               57
#> 45       82 2006               49
#> 46       83 1990               79
#> 47       84 1999               59
#> 48       96 2002               58
#> 49        97 2005                  54
#> 50       101 1988                  80
#> 51       104 2005                  54
#> 52       105 2004                  57
#> 53       109 1993                  65
#> 54       110 1993                  65
#> 55       112 1993                  65
#> 56       115 1994                  58
#> 57       116 1993                  65
#> 58       118 1997                  58
#> 59       119 1991                  81
#> 60       120 1992                  74
#> 61       123 1993                  65
#> 62       124 1994                  58
#> 63       128 1994                  58
#> 64       133 1991                  81
#> 65       134 1979                  60
#> 66       134 1999                  59
#> 67       135 1990                  79
#> 68       138 1991                  81
#> 69       138 2006                  49
#> 70       141 1972                  63
#> 71       141 1988                  80
#> 72       142 1994                  58
#> 73       143 1980                  61
#> 74       144 1987                  78
#> 75       149 1976                  63
#> 76       150 1993                  65
#> 77       154 1990                  79
#> 78       155 1993                  65
#> 79       158 1965                  40
#> 80       158 1986                  77
#> 81       159 2000                  57
#> 82       162 2004                  57
#> 83       163 1996                  57
#> 84       163 2001                  59
#> 85       164 1982                  72
#> 86       167 1988                  80
#> 87       168 1993                  65
#> 88       169 1992                  74
#> 89       177 1974                  62
#> 90       177 1978                   1
#> 91       183 1983                   2
#> 92       187 1994                  58
#> 93       188 1985                  76
#> 94       199 1994                  58
#> 95       201 1991                  81
#> 96       202 1978                  63
#> 97        17 2009                   0
#> 98        70 1999                   0
#> 99        82 1994                   0
#> 100      109 1999                   0
#> 101      133 1999                   0
#> 102     143 1993                0
#> 103     167 1991                0
#> 104     177 1992                0
#> 105     183 1973                0

summary(msets.none)
#> $overview
#>     wbcode2 year matched.set.size
#> 1         4 1992               74
#> 2         4 1997                2
#> 3         6 1973               63
#> 4         6 1983               73
#> 5         7 1991               81
#> 6         7 1998                1
#> 7        12 1992               74
#> 8        13 2003               58
#> 9        15 1991               81
#> 10       16 1977               63
#> 11       17 1991               81
#> 12       18 1991               81
#> 13       22 1991               81
#> 14       25 1982               72
#> 15       26 1985               76
#> 16       31 1993               65
#> 17       34 1990               79
#> 18       36 2000               57
#> 19       38 1992               74
#> 20       40 1990               79
#> 21       40 1996                1
#> 22       40 2002                1
#> 23       41 1991               81
#> 24       45 1993               65
#> 25       47 1999               59
#> 26       50 1978               63
#> 27       52 1979               60
#> 28       55 1978               63
#> 29       56 1992               74
#> 30       57 1995               57
#> 31       59 1990                1
#> 32       64 1995               57
#> 33       65 1970               58
#> 34       65 1979               60
#> 35       65 1996               57
#> 36       70 1994               58
#> 37       70 2005                1
#> 38       72 1975               61
#> 39       73 1984               74
#> 40       75 1966               46
#> 41       75 1986               77
#> 42       78 1992               74
#> 43       80 1982               72
#> 44       81 2000               57
#> 45       82 2006               49
#> 46        83  1990                     79
#> 47        84  1999                     59
#> 48        96  2002                     58
#> 49        97  2005                     54
#> 50       101  1988                     80
#> 51       104  2005                     54
#> 52       105  2004                     57
#> 53       109  1993                     65
#> 54       110  1993                     65
#> 55       112  1993                     65
#> 56       115  1994                     58
#> 57       116  1993                     65
#> 58       118  1997                     58
#> 59       119  1991                     81
#> 60       120  1992                     74
#> 61       123  1993                     65
#> 62       124  1994                     58
#> 63       128  1994                     58
#> 64       133  1991                     81
#> 65       134  1979                     60
#> 66       134  1999                     59
#> 67       135  1990                     79
#> 68       138  1991                     81
#> 69       138  2006                     49
#> 70       141  1972                     63
#> 71       141  1988                     80
#> 72       142  1994                     58
#> 73       143  1980                     61
#> 74       144  1987                     78
#> 75       149  1976                     63
#> 76       150  1993                     65
#> 77       154  1990                     79
#> 78       155  1993                     65
#> 79       158  1965                     40
#> 80       158  1986                     77
#> 81       159  2000                     57
#> 82       162  2004                     57
#> 83       163  1996                     57
#> 84       163  2001                     59
#> 85       164  1982                     72
#> 86       167  1988                     80
#> 87       168  1993                     65
#> 88       169  1992                     74
#> 89       177  1974                     62
#> 90       177  1978                      1
#> 91       183  1983                      2
#> 92       187  1994                     58
#> 93       188  1985                     76
#> 94       199  1994                     58
#> 95       201  1991                     81
#> 96       202  1978                     63
#> 97        17  2009                      0
#> 98        70  1999                      0
#> 99       82 1994                0
#> 100     109 1999                0
#> 101     133 1999                0
#> 102     143 1993                0
#> 103     167 1991                0
#> 104     177 1992                0
#> 105     183 1973                0
#>
#> $set.size.summary
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   57.00   62.00   55.75   74.00   81.00
#>
#> $number.of.treated.units
#> [1] 105
#>
#> $num.units.empty.set
#> [1] 9
#>
#> $lag
#> [1] 4
summary(msets.maha)
#> $overview
#>     wbcode2 year matched.set.size
#> 1         4 1992               74
#> 2         4 1997                2
#> 3         6 1973               63
#> 4         6 1983               73
#> 5         7 1991               81
#> 6         7 1998                1
#> 7        12 1992               74
#> 8        13 2003               58
#> 9        15 1991               81
#> 10       16 1977               63
#> 11       17 1991               81
#> 12       18 1991               81
#> 13       22 1991               81
#> 14       25 1982               72
#> 15       26 1985               76
#> 16       31 1993               65
#> 17       34 1990               79
#> 18       36 2000               57
#> 19       38 1992               74
#> 20       40 1990               79
#> 21       40 1996                1
#> 22       40 2002                1
#> 23       41 1991               81
#> 24       45 1993               65
#> 25       47 1999               59
#> 26       50 1978               63
#> 27       52 1979               60
#> 28       55 1978               63
#> 29       56 1992               74
#> 30       57 1995               57
#> 31         59  1990                    1
#> 32         64  1995                   57
#> 33         65  1970                   58
#> 34         65  1979                   60
#> 35         65  1996                   57
#> 36         70  1994                   58
#> 37         70  2005                    1
#> 38         72  1975                   61
#> 39         73  1984                   74
#> 40         75  1966                   46
#> 41         75  1986                   77
#> 42         78  1992                   74
#> 43         80  1982                   72
#> 44         81  2000                   57
#> 45         82  2006                   49
#> 46         83  1990                   79
#> 47         84  1999                   59
#> 48         96  2002                   58
#> 49         97  2005                   54
#> 50        101  1988                   80
#> 51        104  2005                   54
#> 52        105  2004                   57
#> 53        109  1993                   65
#> 54        110  1993                   65
#> 55        112  1993                   65
#> 56        115  1994                   58
#> 57        116  1993                   65
#> 58        118  1997                   58
#> 59        119  1991                   81
#> 60        120  1992                   74
#> 61        123  1993                   65
#> 62        124  1994                   58
#> 63        128  1994                   58
#> 64        133  1991                   81
#> 65        134  1979                   60
#> 66        134  1999                   59
#> 67        135  1990                   79
#> 68        138  1991                   81
#> 69        138  2006                   49
#> 70        141  1972                   63
#> 71        141  1988                   80
#> 72        142  1994                   58
#> 73        143  1980                   61
#> 74        144  1987                   78
#> 75        149  1976                   63
#> 76        150  1993                   65
#> 77        154  1990                   79
#> 78        155  1993                   65
#> 79        158  1965                   40
#> 80        158  1986                   77
#> 81        159  2000                   57
#> 82        162  2004                   57
#> 83        163  1996                   57
#> 84      163 2001                59
#> 85      164 1982                72
#> 86      167 1988                80
#> 87      168 1993                65
#> 88      169 1992                74
#> 89      177 1974                62
#> 90      177 1978                 1
#> 91      183 1983                 2
#> 92      187 1994                58
#> 93      188 1985                76
#> 94      199 1994                58
#> 95      201 1991                81
#> 96      202 1978                63
#> 97       17 2009                 0
#> 98       70 1999                 0
#> 99       82 1994                 0
#> 100     109 1999                 0
#> 101     133 1999                 0
#> 102     143 1993                 0
#> 103     167 1991                 0
#> 104     177 1992                 0
#> 105     183 1973                 0
#>
#> $set.size.summary
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    0.00   57.00   62.00   55.75   74.00   81.00
#>
#> $number.of.treated.units
#> [1] 105
#>
#> $num.units.empty.set
#> [1] 9
#>
#> $lag
#> [1] 4
```

This information can also be shown graphically. Using the `plot` method, users can visualize the distribution of the size of matched sets. By default, a red line will indicate the number of matched sets where treated units were unable to be matched with any control units (eg. the number of empty matched sets). These plots can be adjusted using the arguments described in the documentation, or any other arguments normally passed to `graphics::plot`.

```
plot(msets.none)
```



Subsetting `matched.set` objects Since `matched.set` objects are just lists with attributes, you can expect the `[` and `[[` functions to work similarly to how they would with a list. So, for instance, users can extract information about matched sets using numerical indices or by taking advantage of the naming scheme.

```
msets.maha[1] #prints like a matched.set object
#>   wbcode2 year matched.set.size
#> 1       4 1992               74
msets.maha[[1]] #prints the matched control unit ids.
#>  [1]   3  13  16  19  28  29  31  35  36  37  43  45  47  51  53  57  62  64  65
#> [20]  67  70  71  81  84  87  93  95  96  97 103 104 105 109 110 112 114 115 116
#> [39] 118 123 124 125 128 129 134 140 142 150 155 156 157 159 161 163 168 171 172
#> [58] 173 175 176 178 179 180 182 184 186 187 190 193 196 197 199 200 202
#> attr(,"treatment.change")
#> [1] 1
#> attr(,"control.change")
#>  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#> [39] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#> attr(,"weights")
#>   3  13  16  19  28  29  31  35  36  37  43  45  47  51  53  57  62  64  65  67
#> 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#>  70  71  81  84  87  93  95  96  97 103 104 105 109 110 112 114 115 116 118 123
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 124 125 128 129 134 140 142 150 155 156 157 159 161 163 168 171 172 173 175 176
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 178 179 180 182 184 186 187 190 193 196 197 199 200 202
#> 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.0 0.0
msets.maha[["4.1992"]] # same behavior as above
#>  [1]   3  13  16  19  28  29  31  35  36  37  43  45  47  51  53  57  62  64  65
#> [20]  67  70  71  81  84  87  93  95  96  97 103 104 105 109 110 112 114 115 116
#> [39] 118 123 124 125 128 129 134 140 142 150 155 156 157 159 161 163 168 171 172
#> [58] 173 175 176 178 179 180 182 184 186 187 190 193 196 197 199 200 202
#> attr(,"treatment.change")
#> [1] 1
#> attr(,"control.change")
#>  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#> [39] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
#> attr(,"weights")
#>   3  13  16  19  28  29  31  35  36  37  43  45  47  51  53  57  62  64  65  67
#> 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#>  70  71  81  84  87  93  95  96  97 103 104 105 109 110 112 114 115 116 118 123
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 124 125 128 129 134 140 142 150 155 156 157 159 161 163 168 171 172 173 175 176
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 178 179 180 182 184 186 187 190 193 196 197 199 200 202
#> 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.0 0.0
```

Control Unit Weights

Each matched set will have an attribute named "weights." This vector indicates the weights calculated for each control unit in the refinement process. Each of the possible refinement methods behave differently. We continue with our running examples to observe this. When the `refinement.method` is set to `none`, all control units will receive equal weights. Using the Mahalanobis distance refinement method will identify the `size.match` most similar units to each treated unit. These units then receive a weight, while all other control units receive a weight of zero. We can use R's standard interface for working with object attributes to examine the weights. The non-zero weights assigned to control units will always sum to one in a given matched set.

```
# Examine the weights assinged to control units in first matched set
attr(msets.none[[1]], "weights")
#>          3         13         16         19         28         29         31
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>         35         36         37         43         45         47         51
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>         53         57         62         64         65         67         70
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>         71         81         84         87         93         95         96
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>         97        103        104        105        109        110        112
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        114        115        116        118        123        124        125
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        128        129        134        140        142        150        155
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        156        157        159        161        163        168        171
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        172        173        175        176        178        179        180
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        182        184        186        187        190        193        196
#> 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351 0.01351351
#>        197        199        200        202
#> 0.01351351 0.01351351 0.01351351 0.01351351
attr(msets.maha[[1]], "weights")
#>   3  13  16  19  28  29  31  35  36  37  43  45  47  51  53  57  62  64  65  67
#> 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#>  70  71  81  84  87  93  95  96  97 103 104 105 109 110 112 114 115 116 118 123
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 124 125 128 129 134 140 142 150 155 156 157 159 161 163 168 171 172 173 175 176
#> 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
#> 178 179 180 182 184 186 187 190 193 196 197 199 200 202
#> 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.0 0.0
```

Comparing Methods of Refinement

While we have mostly looked at a simple case of refinement using Mahalanobis distance so far, there are a number of ways to tune the refinement process. There are no hard and fast rules for determining the best configuration. Instead, users should use their substantive knowledge to experiment with and evaluate a number of different setups. In general, users should consider 1) the number of matched sets 2) the number of controls matched to each treated unit and 3) covariate balance as they configure `PanelMatch`. Having a large number of small matched sets will create larger standard errors in the estimation stage. Poorly balanced covariates suggest undesirable comparisons between treated and control units. Users ought to consider the method of refinement, the variables used for calculating weights, the size of the lag window, the desired procedures for handling missing data (See the `match.missing` and `listwise.delete` arguments in the `PanelMatch` function documentation), and the maximum size of matched sets (when using a matching method). The package aims to provide a number of featres to help users with this process. First, the `print`, `plot`, and `summary` methods for `matched.set` objects will help users study **1** and **2**. To help evaluate covariate balance, users can take advantage of the `get_covariate_balance` function.

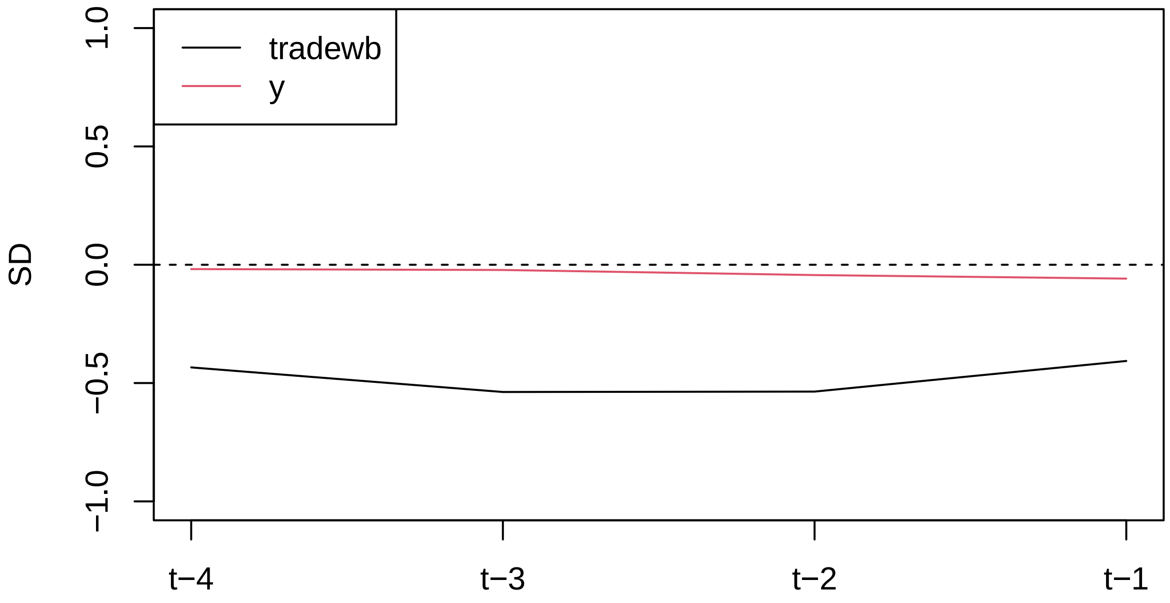
Using `get_covariate_balance` The following code runs a number of different `PanelMatch` refinement configurations and calculates the covariate balance for each setup. Lower values in the covariate balance calculations are desirable. See the documentation of `get_covariate_balance` for more detailed information.

```r
PM.results.none <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "none",
                         data = dem, match.missing = TRUE,
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)


PM.results.maha <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "mahalanobis",
                         data = dem, match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)

# listwise deletion used for missing data
PM.results.listwise <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "mahalanobis",
                         data = dem, match.missing = FALSE, listwise.delete = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)

# propensity score based weighting method
```

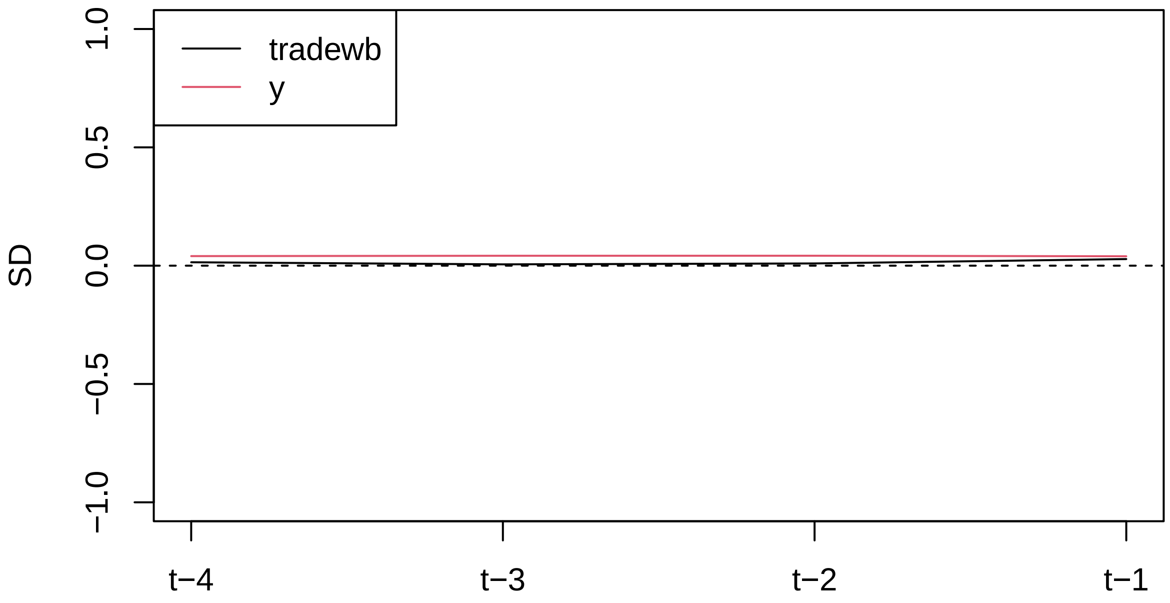
```
PM.results.ps.weight <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "ps.weight",
                         data = dem, match.missing = FALSE, listwise.delete = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE)

get_covariate_balance(PM.results.none$att,
                      data = dem,
                      covariates = c("tradewb", "y"),
                      plot = FALSE)
#>         tradewb            y
#> t_4 -0.07245466  0.291871990
#> t_3 -0.20930129  0.208654876
#> t_2 -0.24425207  0.107736647
#> t_1 -0.10806125 -0.004950238

get_covariate_balance(PM.results.maha$att,
                      data = dem,
                      covariates = c("tradewb", "y"),
                      plot = FALSE)
#>         tradewb          y
#> t_4  0.04558637 0.09701606
#> t_3 -0.03312750 0.10844046
#> t_2 -0.01396793 0.08890753
#> t_1  0.10474894 0.06618865

get_covariate_balance(PM.results.listwise$att,
                      data = dem,
                      covariates = c("tradewb", "y"),
                      plot = FALSE)
#>         tradewb          y
#> t_4  0.05634922 0.05223623
#> t_3 -0.01104797 0.05217896
#> t_2  0.01411473 0.03094133
#> t_1  0.06850180 0.02092209

get_covariate_balance(PM.results.ps.weight$att,
                      data = dem,
                      covariates = c("tradewb", "y"),
                      plot = FALSE)
#>         tradewb          y
#> t_4 0.014362590 0.04035905
#> t_3 0.005529734 0.04188731
#> t_2 0.009410044 0.04195008
#> t_1 0.027907540 0.03975173
```

The `get_covariate_balance` function has a few useful options. Users can generate plots showing the covariate balance by setting `plot = TRUE`. These plots can be customized using the same arguments one would normally pass to the base R `plot` method. Users can also set `use.equal.weights = TRUE` to easily get the balance of unrefined sets. This makes it easy to understand the impact of refinement.

```
# Use equal weights
get_covariate_balance(PM.results.ps.weight$att,
                      data = dem,
                      use.equal.weights = TRUE,
                      covariates = c("tradewb", "y"),
                      plot = TRUE,
                      # visualize by setting plot to TRUE
                      ylim = c(-1, 1))
```

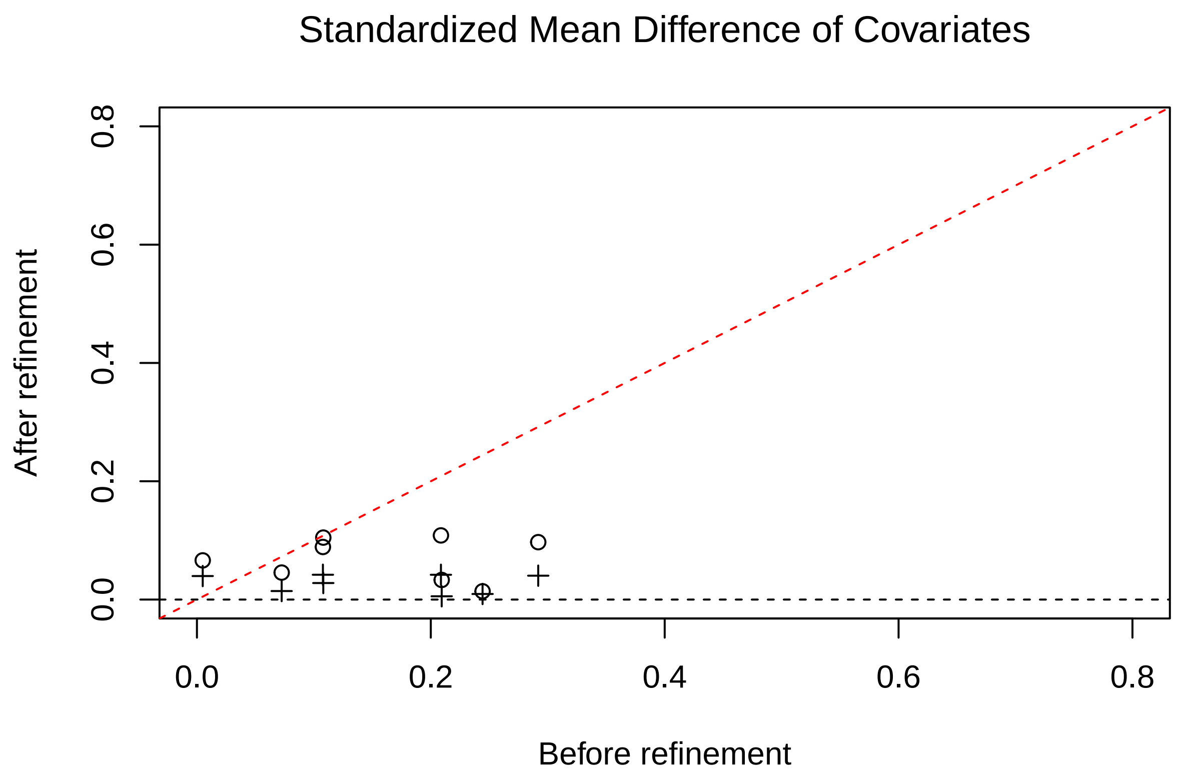


```
# Compare covariate balance to refined sets
# See large improvement in balance
get_covariate_balance(PM.results.ps.weight$att,
                    data = dem,
                    covariates = c("tradewb", "y"),
                    plot = TRUE,
                    # visualize by setting plot to TRUE
                    ylim = c(-1, 1))
```



We can also evaluate our results using the `balance_scatter` function:

```
balance_scatter(matched_set_list =
                  list(PM.results.maha$att,
                       PM.results.ps.weight$att),
                data = dem,
                covariates = c("y", "tradewb"))
```



PanelEstimate

We now move to the other major part of the package: obtaining point estimates and standard errors using `PanelEstimate`. There are a variety of methods for calculating standard errors: “bootstrap”, “conditional”, and “unconditional”. By default, the package uses a bootstrap approach for calculating standard errors (1000 bootstrap iterations are used by default). When the `qoi` is set to “att”, “art”, or “atc”, you may also use analytical methods for calculating standard errors: “conditional” or “unconditional”, with the former assuming independence across units (but not time) and the latter not making such assumptions. Consult Imai, Kim, and Wang (2021) for descriptions about these methods. .95 is the default confidence level.

The function returns a `PanelEstimate` object, which behaves like a list, much like the other objects in the package. As such, you can access the various elements just as you would in a list.

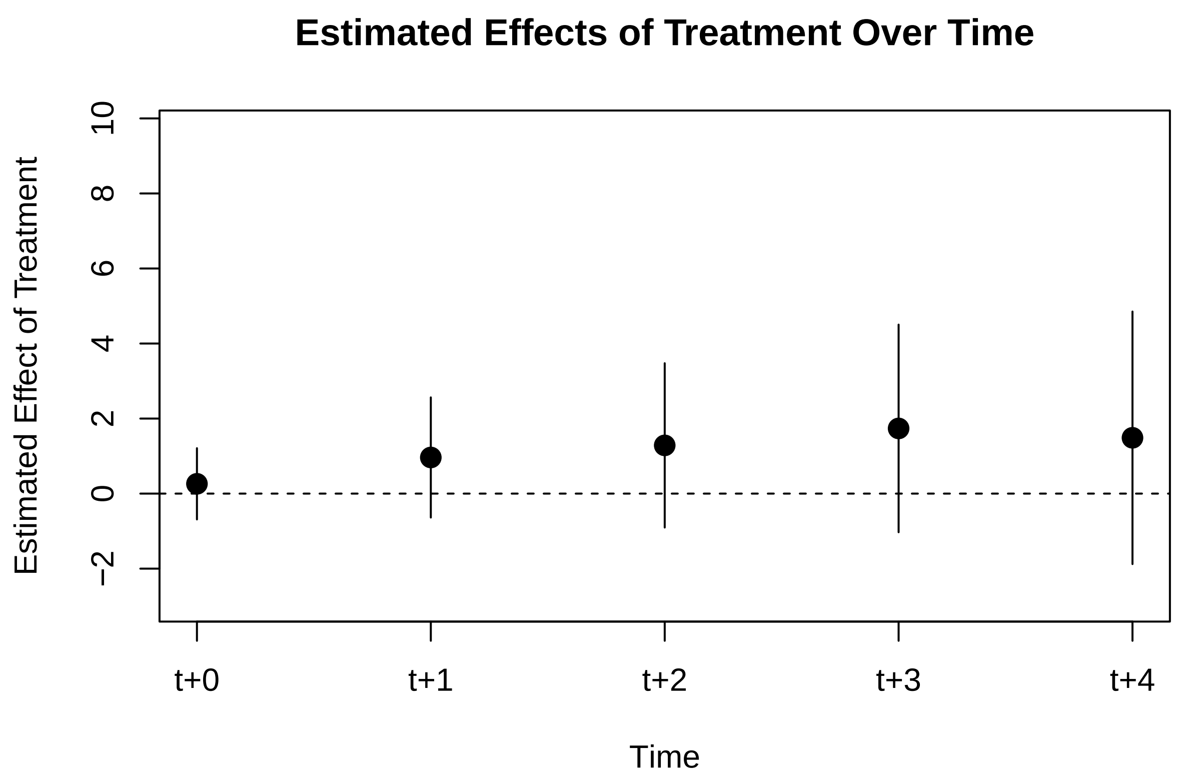
```
PE.results <- PanelEstimate(sets = PM.results.ps.weight, data = dem,
                            se.method = "bootstrap",
                            number.iterations = 1000,
                            confidence.level = .95)

# View the point estimates
PE.results[["estimates"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.2609565 0.9630847 1.2851017 1.7370930 1.4871846
```

```
# View standard errors
PE.results[["standard.error"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.6464514 1.0026846 1.3735504 1.7472245 2.1334570

# use conditional method
PE.results <- PanelEstimate(sets = PM.results.ps.weight, data = dem,
                            se.method = "conditional",
                            confidence.level = .95)

# View the point estimates
PE.results[["estimates"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.2609565 0.9630847 1.2851017 1.7370930 1.4871846
# View standard errors
PE.results[["standard.error"]]
#>       t+0       t+1       t+2       t+3       t+4
#> 0.4844805 0.8170604 1.1171942 1.4116879 1.7172143
```

PanelEstimate Methods

`PanelEsimate` objects have custom `summary` and `plot` methods defined. The `plot` method can be customized with all of the same arguments/operations as the regular `plot` function in base R. This method shows the point estimates for the specified lead window periods, along with the standard errors.

```
summary(PE.results)
#> Weighted Difference-in-Differences with Propensity Score
#> Matches created with 4 lags
#>
#> Standard errors computed with conditional  method
#>
#> Estimate of Average Treatment Effect on the Treated (ATT) by Period:
#> $summary
#>      estimate std.error       2.5%    97.5%
#> t+0 0.2609565 0.4844805 -0.6886078 1.210521
#> t+1 0.9630847 0.8170604 -0.6383243 2.564494
#> t+2 1.2851017 1.1171942 -0.9045586 3.474762
#> t+3 1.7370930 1.4116879 -1.0297644 4.503950
#> t+4 1.4871846 1.7172143 -1.8784937 4.852863
#>
#> $lag
#> [1] 4
#>
#> $qoi
#> [1] "att"

plot(PE.results)
```



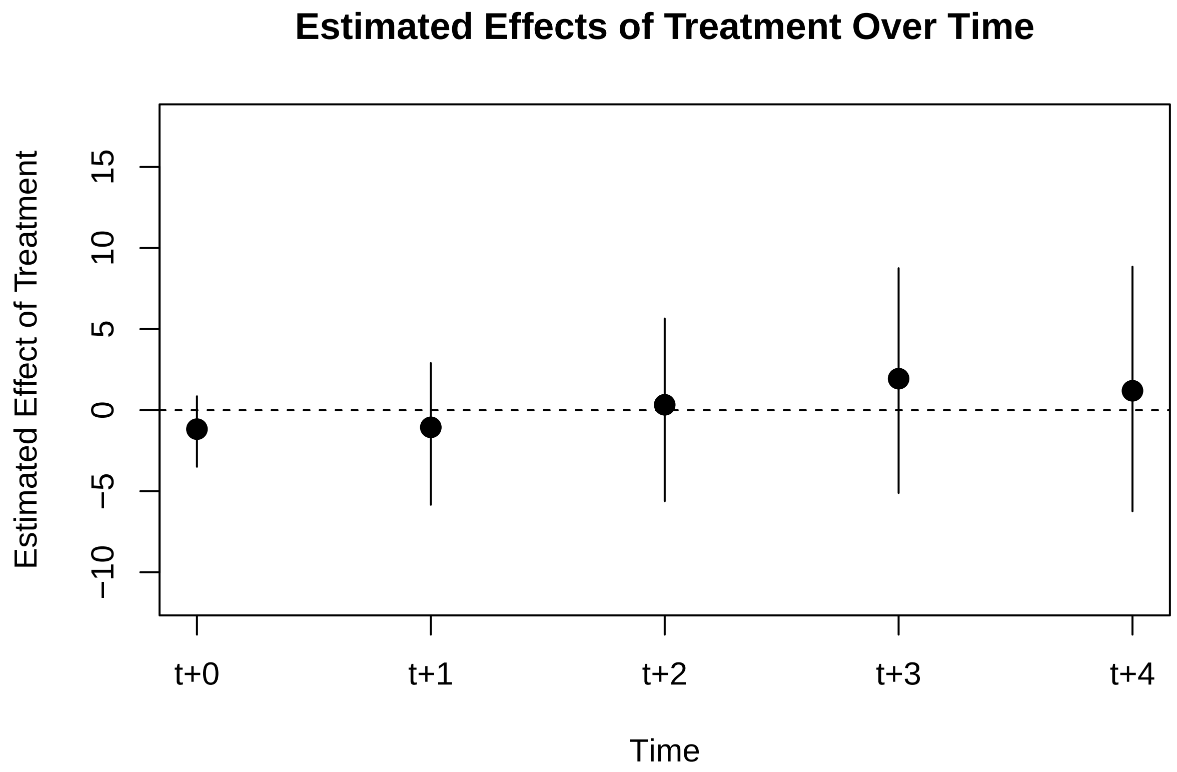
This last plot makes it clear that the effect of treatment on treated units (att) in this configuration is statistically insignificant.

Moderating Variables

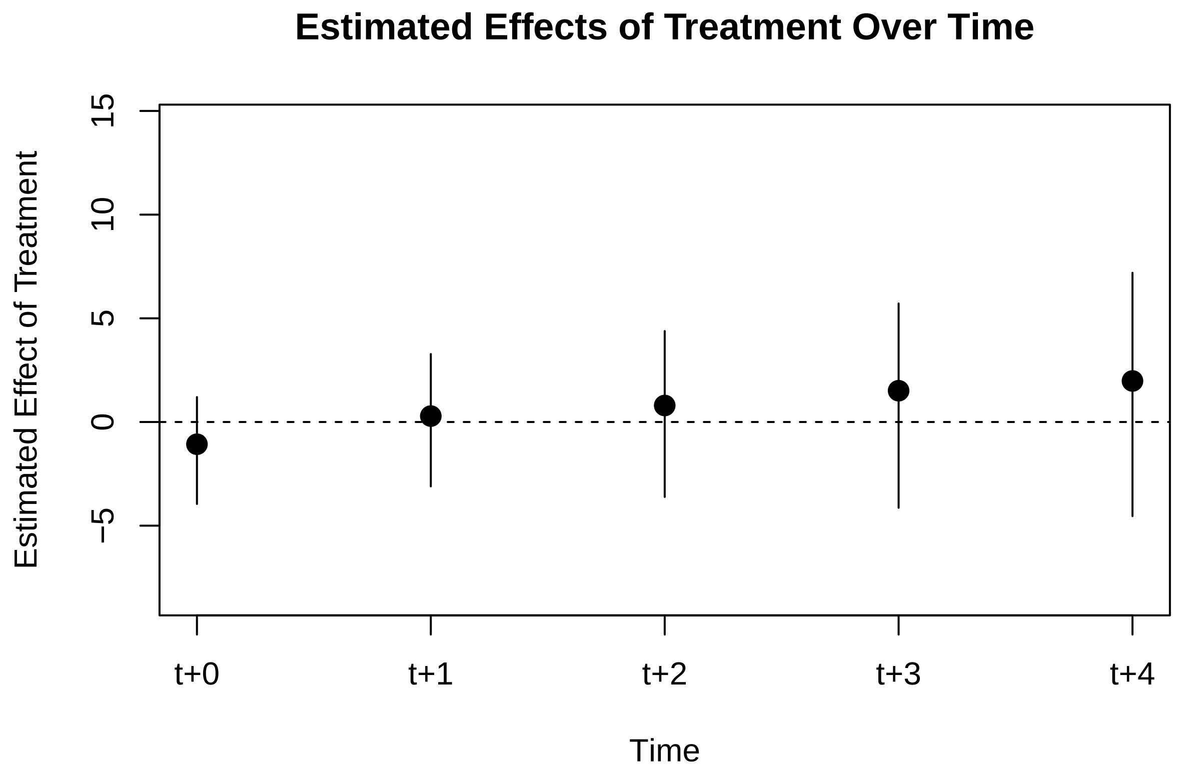
The `PanelEstimate` function can handle moderating variables. When a moderating variable is specified in the `PanelEstimate` function, a list of `PanelEstimate` objects are returned, each with the structure of a standard `PanelEstimate` object. There will be a separate `PanelEstimate` object in the list for each value of the moderating variable, and the name of each element in the returned list will reflect this value. The moderating variable is assumed to be categorical.

```
# add simple moderating variable
dem$moderator <- 0
dem$moderator <- ifelse(dem$wbcode2 > 100, 1, 2)


PM.results <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                         treatment = "dem", refinement.method = "mahalanobis",
                         data = dem, match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)), # lags
                         size.match = 5, qoi = "att", outcome.var = "y",
                         lead = 0:4, forbid.treatment.reversal = FALSE,
                         use.diagonal.variance.matrix = TRUE)

PE.results <- PanelEstimate(sets = PM.results, data = dem, moderator = "moderator")

# Names correspond to
# moderator values
names(PE.results)
#> [1] "2" "1"
```

```r
# Extract each result
plot(PE.results[[1]])
```



```r
plot(PE.results[[2]])
```


Estimated Effects of Treatment Over Time
Estimated Effect of Treatment
t+0
t+1
t+2
t+3
t+4
Time