Package ‘PanelMatch’

March 3, 2025

Type Package

Title Matching Methods for Causal Inference with Time-Series Cross-Sectional Data

Version 3.0.0

Date 2025-02-26

Description Implements a set of methodological tools that enable researchers to apply matching methods to time-series cross-sectional data. Imai, Kim, and Wang (2023) <http://web.mit.edu/insong/www/pdf/tscs.pdf> proposes a nonparametric generalization of the difference-in-differences estimator, which does not rely on the linearity assumption as often done in practice. Researchers first select a method of matching each treated observation for a given unit in a particular time period with control observations from other units in the same time period that have a similar treatment and covariate history. These methods include standard matching methods based on propensity score and Mahalanobis distance, as well as weighting methods. Once matching and refinement is done, treatment effects can be estimated with standard errors. The package also offers diagnostics for researchers to assess the quality of their results.

License GPL (>= 3)

Imports Rcpp (>= 0.12.5), data.table, ggplot2, CBPS, stats, graphics, MASS, Matrix, doParallel, foreach, methods

Depends R (>= 2.14.0)

LinkingTo RcppArmadillo, Rcpp, RcppEigen

Encoding UTF-8

LazyData true

BugReports https://github.com/insongkim/PanelMatch/issues

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

NeedsCompilation yes

Author In Song Kim [aut, cre],
Adam Rauh [aut],
Erik Wang [aut],
Kosuke Imai [aut]

Maintainer In Song Kim <insong@mit.edu>

Repository CRAN

Date/Publication 2025-03-03 09:50:08 UTC

Contents

dem — *Country-year level democratization data*

Description

A dataset containing the democracy indicator for 184 countries from 1960 to 2010

Format

A data.frame containing 9384 rows and 3 variables

Details

- wbcode2. World Bank country ID. Integer.
- year. year (1960–2010). Integer.
- dem. binary indicator of democracy as defined in Acemoglu et al (2019).
- y log of GDP per capita in 2000 constant dollars (multiplied by 100). Numeric.
- tradewb Exports plus imports as a share of GDP from World Bank. Numeric.

Source

Acemoglu, Daron, Suresh Naidu, Pascual Restrepo, and James A Robinson. "Democracy does cause growth." Journal of Political Economy.

DisplayTreatment — *Visualize the treatment distribution across units and time in a panel data set*

Description

Visualize the treatment distribution across units and time in a panel data set

Usage

```
DisplayTreatment(
  panel.data,
  color.of.treated = "red",
  color.of.untreated = "blue",
  title = "Treatment Distribution \n Across Units and Time",
  xlab = "Time",
  ylab = "Unit",
  x.size = NULL,
  y.size = NULL,
  legend.position = "none",
  x.angle = NULL,
  y.angle = NULL,
  legend.labels = c("not treated", "treated"),
  decreasing = FALSE,
  matched.set = NULL,
  show.set.only = FALSE,
  hide.x.tick.label = FALSE,
  hide.y.tick.label = FALSE,
  gradient.weights = FALSE,
  dense.plot = FALSE
)
```

Arguments

`panel.data`	`PanelData` object
`color.of.treated`	Color of the treated observations provided as a character string (this includes hex values). Default is red.
`color.of.untreated`	Color of the untreated observations provided as a character string (this includes hex values). Default is blue.
`title`	Title of the plot provided as character string
`xlab`	Character label of the x-axis
`ylab`	Character label of the y-axis
`x.size`	Numeric size of the text for xlab or x axis tick labels. Assign x.size = NULL to use built in ggplot2 method of determining label size. When the length of the time period is long, consider setting to NULL and adjusting size and ratio of the plot.
`y.size`	Numeric size of the text for ylab or y axis tick labels. Assign y.size = NULL to use built in ggplot2 method of determining label size. When the number of units is large, consider setting to NULL and adjusting size and ratio of the plot.
`legend.position`	Position of the legend. Provide this according to ggplot2 standards.
`x.angle`	Angle (in degrees) of the tick labels for x-axis
`y.angle`	Angle (in degrees) of the tick labels for y-axis
`legend.labels`	Character vector of length two describing the labels of the legend to be shown in the plot. ggplot2 standards are used.
`decreasing`	Logical. Determines if display order should be increasing or decreasing by the amount of treatment received. Default is `decreasing` = FALSE.
`matched.set`	(optional) a `matched.set` object containing a single treated unit and a set of matched controls. If provided, this set will be highlighted on the resulting plot.
`show.set.only`	(optional) logical. If TRUE, only the treated unit and control units contained in the provided `matched.set` object will be shown on the plot. Default is FALSE. If no `matched.set` is provided, then this argument will have no effect.

`hide.x.tick.label`	logical. If TRUE, x axis tick labels are not shown. Default is FALSE.
`hide.y.tick.label`	logical. If TRUE, y axis tick labels are not shown. Default is FALSE.
`gradient.weights`	(optional) logical. If TRUE, the "darkness"/shade of units in the provided `matched.set` object will be displayed according to their weight. Control units with higher weights will appear darker on the resulting plot. Control units with lower weights will appear lighter. This argument has no effect unless a `matched.set` is provided.
`dense.plot`	logical. if TRUE, lines between tiles are removed on resulting plot. This is useful for producing more readable plots in situations where the number of units and/or time periods is very high.

Value

`DisplayTreatment` returns a treatment variation plot (generated via ggplot2 geom_tile() or geom_raster()), which visualizes the variation of treatment across units and time. The results can be customized using ggplot2 syntax.

Author(s)

In Song Kim <insong@mit.edu>, Erik Wang <haixiao@Princeton.edu>, Adam Rauh <amrauh@umich.edu>, and Kosuke Imai <imai@harvard.edu>

Examples

```
dem.panel <- PanelData(panel.data = dem,
              unit.id = "wbcode2",
              time.id = "year",
              treatment = "dem",
              outcome = "y")
DisplayTreatment(panel.data = dem.panel,
                 legend.position = "none",
                 xlab = "year", ylab = "Country Code")
```

`distances` *Get distances See distances.matched.set method*

Description

Get distances See distances.matched.set method

Usage

```
distances(object)
```

Arguments

`object`	`matched.set` object

`distances.matched.set` *Extract the distances of matched control units*

Description

Extract the distances of matched control units

Usage

```
## S3 method for class 'matched.set'
distances(object)
```

Arguments

`object`	a matched.set object

Value

A named list of named vectors. Each element corresponds to a matched set and will be a named vector, where the names of each element will identify a matched control unit and its distance from the treated observation within a particular matched set. These correspond to the "distances" attribute, which are calculated and included when the `verbose` option is set to TRUE in `PanelMatch`.

Examples

```
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.panel, lag = 4,
                         refinement.method = "mahalanobis",
                         verbose = TRUE,
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
r1 <- extract(PM.results, qoi = "att")
lt <- distances(r1)
```

`extract` *Extract matched.set objects from PanelMatch results*

Description

Extract matched.set objects from PanelMatch results

Usage

```
extract(pm.object, qoi)
```

Arguments

`pm.object`	`PanelMatch` object
`qoi`	character, specifying the qoi. Valid inputs include "att", "atc", "art", and NULL. If NULL, function extracts att, art, or atc results if possible. Otherwise, throws an error if ate is specified.

`extract.PanelMatch` *Extract matched.set objects from PanelMatch results*

Description

Extract matched.set objects from PanelMatch results

Usage

```
## S3 method for class 'PanelMatch'
extract(pm.object, qoi = NULL)
```

Arguments

`pm.object`	`PanelMatch` obect
`qoi`	character, specifying the qoi. Valid inputs include "att", "atc", "art", and NULL. If NULL, function extracts att, art, or atc results if possible. Otherwise, throws an error if ate is specified.

Value

a `matched.set` object

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel,
                         lag = 4,
                         refinement.method = "mahalanobis",
                         match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4, forbid.treatment.reversal = FALSE)
extract(PM.results, qoi = "att")
extract(PM.results) # valid since att is specified
```

---

get_covariate_balance *Calculate covariate balance measures for refined and unrefined matched sets*

Description

Calculate covariate balance for user specified covariates across matched sets. Balance is assessed by taking the average of the difference between the values of the specified covariates for the treated unit(s) and the weighted average of the control units across all matched sets. Results are standardized and are expressed in standard deviations. Balance is calculated for each period in the specified lag window.

Usage

```
get_covariate_balance(..., panel.data, covariates, include.unrefined = TRUE)
```

Arguments

`...`	one or more PanelMatch objects
`panel.data`	`PanelData` object
`covariates`	a character vector, specifying the names of the covariates for which the user is interested in calculating balance.
`include.unrefined`	logical. Indicates whether or not covariate balance measures for unrefined matched sets should be included. If TRUE, the function will return covariate balance results for the PanelMatch configurations provided, as well as a set of balance results that assume all matched controls have equal weight (i.e., the matched sets are unrefined). These results are included in addition to whatever PanelMatch configurations are specified to the function. Note that if you provide a PanelMatch object where no refinement is applied (that is, where `refinement.method = "none"`) and set this option to TRUE, then both sets of covariate balance results will be identical. If FALSE, then only balance calculations for the provided PanelMatch specifications are performed and returned.

Value

A list of matrices, or a list of lists (if the QOI is ATE). The matrices contain the calculated covariate balance levels for each specified covariate for each period. Each element in the list (whether that be a matrix or a sublist) corresponds to a `PanelMatch` configuration specified to the function. Results are returned in the order they were provided. Unrefined results are stored as a parallel list object in an attribute called "unrefined.balance.results".

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
# create subset of data for simplicity
#add some additional data to data set for demonstration purposes
dem.sub$rdata <- runif(runif(nrow(dem.sub)))
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb + rdata,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
get_covariate_balance(PM.results, panel.data = dem.sub.panel, covariates = c("tradewb", "rdata"))
```

---

get_set_treatment_effects

Calculate matched set level treatment effects

Description

Calculate the size of treatment effects for each matched set.

Usage

```
get_set_treatment_effects(pm.obj, panel.data, lead)
```

Arguments

`pm.obj`	an object of class `PanelMatch`
`panel.data`	`PanelData` object with the time series cross sectional data used for matching, refinement, and estimation
`lead`	integer (or integer vector) indicating the time period(s) in the future for which the treatment effect size will be calculated. Calculations will be made for the period t + lead, where t is the time of treatment. If more than one lead value is provided, then calculations will be performed for each value.

Value

a list equal in length to the number of lead periods specified to the `lead` argument. Each element in the list is a vector of the matched set level effect estimates.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
set.effects <- get_set_treatment_effects(pm.obj = PM.results,
                panel.data = dem.sub.panel, lead = 0)
```

`get_unrefined_balance` *Extract just the unrefined covariate balance results, if they exist*

Description

Extract just the unrefined covariate balance results, if they exist

Usage

```
get_unrefined_balance(pb.object)
```

Arguments

`pb.object`	`PanelBalance` object

`get_unrefined_balance.PanelBalance` *Extract unrefined covariate balance results, if they exist*

Description

Extract unrefined covariate balance results, if they exist

Usage

```
## S3 method for class 'PanelBalance'
get_unrefined_balance(pb.object)
```

Arguments

`pb.object`	`PanelBalance` object

Value

A `PanelBalance` object, with just the unrefined balance results

Examples

```
dem$rdata <- runif(runif(nrow(dem)))
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
pm.obj <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "mahalanobis",
                     panel.data = dem.panel, match.missing = TRUE,
                 covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                     size.match = 5, qoi = "att")

# create multiple configurations to compare
pm2 <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "ps.match",
                  panel.data = dem.panel, match.missing = TRUE,
                covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                  size.match = 5, qoi = "att")

pb <- get_covariate_balance(pm.obj, pm2,
                            include.unrefined = TRUE,
                            panel.data = dem.panel,
                            covariates = c("tradewb", "rdata"))
get_unrefined_balance(pb)
```

`matched_set` *A constructor for the matched.set class.*

Description

Users should never need to use this function by itself. See below for more about `matched.set` objects.

Usage

```
matched_set(matchedsets, id, t, L, t.var, id.var, treatment.var)
```

Arguments

`matchedsets`	a list of treated units and matched control units. Each element in the list should be a vector of control unit ids.
`id`	A vector containing the ids of treated units
`t`	A vector containing the times of treatment for treated units.
`L`	integer specifying the length of the lag window used in matching
`t.var`	string specifying the time variable
`id.var`	string specifying the unit id variable
`treatment.var`	string specifying the treatment variable.

The constructor function returns a `matched.set` object. `matched.set` objects are a modified list. Each element in the list is a vector of ids corresponding to the control unit ids in a matched set. Additionally, these vectors might have additional attributes – "weights". These correspond to the weights assigned to each control unit, as determined by the specified refinement method. Each element in the list also has a name, which corresponds to the unit id of the treated unit and time of treatment, concatenated together and separated by a period. `matched.set` objects also have a number of methods defined: `summary`, `plot`, and `` `[` ``. `matched.set` objects can be modified manually as long as these conventions (and conventions about other attributes) are maintained. It is important to note that `matched.set` objects are distinct from `PanelMatch` objects. `matched.set` objects are often contained within `PanelMatch` objects.

Value

`matched.set` objects have additional attributes. These reflect the specified parameters when using the `PanelMatch` function:

`lag`	an integer value indicating the length of treatment history to be used for matching. Treated and control units are matched based on whether or not they have exactly matching treatment histories in the lag window.
`t.var`	time variable name, represented as a character/string
`id.var`	unit id variable name, represented as a character/string
`treatment.var`	treatment variable name, represented as a character/string
`class`	class of the object: should always be "matched.set"
`refinement.method`	method used to refine and/or weight the control units in each set.
`covs.formula`	One sided formula indicating which variables should be used for matching and refinement
`match.missing`	Logical variable indicating whether or not units should be matched on the patterns of missingness in their treatment histories
`max.match.size`	Maximum size of the matched sets after refinement. This argument only affects results when using a matching method

Author(s)

Adam Rauh <amrauh@umich..edu>, In Song Kim <insong@mit.edu>, Erik Wang <haixiao@Princeton.edu>, and Kosuke Imai <imai@harvard.edu>

`PanelData` *Pre-process and balance panel data*

Description

Pre-process and balance panel data

Usage

```
PanelData(panel.data, unit.id, time.id, treatment, outcome)
```

Arguments

`panel.data`	A `data.frame` object containing time series cross sectional data. Time data should be sequential integers that increase by 1. Unit identifiers must be integers. Treatment data must be binary. If time data is non-integer, the package will attempt to sensibly convert it by converting the data to factor, then to integer. If a conversion is performed, a mapping will be returned as an attribute called "time.data.map"
`unit.id`	A character string indicating the name of unit identifier in the data. This data must be integer.
`time.id`	A character string indicating the name of the time variable in the data.
`treatment`	A character string indicating the name of the treatment variable. The treatment must be a binary indicator variable (integer with 0 for the control group and 1 for the treatment group).
`outcome`	A character string identifying the outcome variable

Value

`PanelData()` returns an object of class `PanelData`. This takes the form of a `data.frame` object with the following properties and attributes. First, the data has been balanced and sorted. These properties are noted in the "is.balanced" and "is.sorted" attributes, respectively. So, each unit appears the same number of times in the resulting `PanelData` object, with NAs filling out missing data. Second, the data has been sorted to appear in order for each unit. Next, the `PanelData` object has the following attributes: "unit.id", "time.id", "treatment", and "outcome" reflecting the variables provided in the specification. If the function attempts to automatically convert time data to be consecutive integers, the mapping between the original time data and the "new" converted time data is provided as a `data.frame` object and stored as the "time.data.map" attribute.

Examples

```
d <- PanelData(panel.data = dem,
               unit.id = "wbcode2",
               time.id = "year",
               treatment = "dem",
               outcome = "y")
```

PanelEstimate *Estimate a causal quantity of interest*

Description

Estimate a causal quantity of interest, including the average treatment effect for treated or control units (att and atc, respectively), the average effect of treatment reversal on reversed units (art), or average treatment effect (ate), as specified in `PanelMatch()`. This is done by estimating the counterfactual outcomes for each treated unit using matched sets. Users will provide matched sets that were obtained by the `PanelMatch` function and obtain point estimates and standard errors.

Usage

```
PanelEstimate(
  sets,
  panel.data,
  number.iterations = 1000,
  df.adjustment = FALSE,
  confidence.level = 0.95,
  moderator = NULL,
  se.method = "bootstrap",
  pooled = FALSE,
  include.placebo.test = FALSE,
  parallel = FALSE,
  num.cores = 1
)
```

Arguments

`sets`	A `PanelMatch` object attained via the `PanelMatch()` function.
`panel.data`	The same time series cross sectional data set provided to the `PanelMatch()` function used to produce the matched sets. This should be a `PanelData` object.
`number.iterations`	If using bootstrapping for calculating standard errors, this is the number of bootstrap iterations. Provide as integer. If `se.method` is not equal to "bootstrap", this argument has no effect.
`df.adjustment`	A logical value indicating whether or not a degree-of-freedom adjustment should be performed for the standard error calculation. The default is `FALSE`. This parameter is only available for the bootstrap method of standard error calculation.

`confidence.level`	A numerical value specifying the confidence level and range of interval estimates for statistical inference. The default is .95.
`moderator`	The name of a moderating variable, provided as a character string. If a moderating variable is provided,the returned object will be a list of `PanelEstimate` objects. The names of the list will reflect the different values of the moderating variable. More specifically, the moderating variable values will be converted to syntactically proper names using `make.names()`.
`se.method`	Method used for calculating standard errors, provided as a character string. Users must choose between "bootstrap", "conditional", and "unconditional" methods. Default is "bootstrap". "bootstrap" uses a block bootstrapping procedure to calculate standard errors. The conditional method calculates the variance of the estimator, assuming independence across units but not across time. The unconditional method also calculates the variance of the estimator analytically, but makes no such assumptions about independence across units. When the quantity of interest is "att", "atc", or "art", all methods are available. Only "bootstrap" is available for the ate. If `pooled` argument is TRUE, then only bootstrap is available.
`pooled`	Logical. If TRUE, estimates and standard errors are returned for treatment effects pooled across the entire lead window. Only available for `se.method = ``bootstrap''`
`include.placebo.test`	Logical. If TRUE, a placebo test is run and returned in the results. The placebo test uses the same specifications for calculating standard errors as the main results. That is, standard errors are calculated according to the user provided `se.method` and `confidence.level` arguments (and, if applicable, parallelization specifications).
`parallel`	Logical. If TRUE and `se.method = ``bootstrap''`, bootstrap procedure will be parallelized. Default is FALSE. If `se.method` is not set to `bootstrap`, this option does nothing.
`num.cores`	Integer. Specifies the number of cores to use for parallelization. If `se.method = ``bootstrap''` and `parallel = TRUE`, then this option will take effect. Otherwise, it will do nothing.

Value

`PanelEstimate` returns a list of class `PanelEstimate` containing the following components:

`estimates`	the point estimates of the quantity of interest for the lead periods specified
`se.method`	The method used to calculate standard errors. This is the same as the argument provided to the function.
`bootstrapped.estimates`	the bootstrapped point estimate values, when applicable
`bootstrap.iterations`	the number of iterations used in bootstrapping, when applicable
`method`	refinement method used to create the matched sets from which the estimates were calculated

`lag`	See PanelMatch() argument `lag` for more information.
`lead`	The lead window sequence for which `PanelEstimate()` is producing point estimates and standard errors.
`confidence.level`	the confidence level
`qoi`	the quantity of interest
`matched.sets`	the refined matched sets used to produce the estimations
`standard.error`	the standard error(s) of the point estimates
`pooled`	Logical indicating whether or not estimates were calculated for individual lead periods or pooled.
`placebo.test`	if `include.placebo.test` = `TRUE`, a placebo test is conducted using `placebo_test()` and returned as a list. See documentation for `placebo_test()` for more about each individual item.

Author(s)

In Song Kim <insong@mit.edu>, Erik Wang <haixiao@Princeton.edu>, Adam Rauh <amrauh@umich.edu>, and Kosuke Imai <imai@harvard.edu>

References

Imai, Kosuke, In Song Kim, and Erik Wang (2023)

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
PE.results <- PanelEstimate(sets = PM.results,
               panel.data = dem.sub.panel,
               se.method = "unconditional")
```

`PanelMatch` *Create and refine sets of matched treated and control observations*

Description

`PanelMatch` identifies treated observations and a matched set for each treated observation. Specifically, for a given treated unit, the matched set consists of control observations that have an identical treatment history up to a number of `lag` time periods. A further refinement of the matched set using matching or weighting techniques, described below.

Usage

```
PanelMatch(
  panel.data,
  lag,
  refinement.method,
  qoi,
  size.match = 10,
  match.missing = TRUE,
  covs.formula = NULL,
  lead = 0,
  verbose = FALSE,
  exact.match.variables = NULL,
  forbid.treatment.reversal = FALSE,
  matching = TRUE,
  listwise.delete = FALSE,
  use.diagonal.variance.matrix = FALSE,
  restrict.control.period = NULL,
  placebo.test = FALSE
)
```

Arguments

`panel.data`	A `PanelData` object containing time series cross sectional data. Time data must be sequential integers that increase by 1. Unit identifiers must be integers. Treatment data must be binary.
`lag`	An integer value indicating the length of treatment history periods to be matched on
`refinement.method`	A character string specifying the matching or weighting method to be used for refining the matched sets. The user can choose "mahalanobis", "ps.match", "CBPS.match", "ps.weight", "CBPS.weight", "ps.msm.weight", "CBPS.msm.weight", or "none". The first three methods will use the `size.match` argument to create sets of at most `size.match` closest control units. Choosing "none" will assign equal weights to all control units in each matched set. The MSM methods refer to marginal structural models. See Imai, Kim, and Wang (2023) for a more in-depth discussion of MSMs.
`qoi`	quantity of interest, provided as a string: `att` (average treatment effect on treated units), `atc` (average treatment effect of treatment on the control units) `art` (average effect of treatment reversal for units that experience treatment reversal), or `ate` (average treatment effect).
`size.match`	An integer dictating the number of permitted closest control units in a matched set after refinement. This argument only affects results when using a matching method ("mahalanobis" or any of the refinement methods that end in ".match"). This argument is not needed and will have no impact if included when a weighting method is specified (any `refinement.method` that includes "weight" in the name).

`match.missing`	Logical variable indicating whether or not units should be matched on the patterns of missingness in their treatment histories. Default is TRUE. When FALSE, neither treated nor control units are allowed to have missing treatment data in the lag window.
`covs.formula`	One sided formula object indicating which variables should be used for matching and refinement. Argument is not needed if `refinement.method` is set to "none" If the user wants to include lagged variables, this can be done using a function, "lag()", which takes two, unnamed, positional arguments. The first is the name of the variable which you wish to lag. The second is the lag window, specified as an integer sequence in increasing order. For instance, I(lag(x, 1:4)) will then add new columns to the data for variable "x" for time t-1, t-2, t-3, and t-4 internally and use them for defining/measuring similarity between units. Other transformations using the I() function, such as I(x^2) are also permitted. The variables specified in this formula are used to define the similarity/distances between units.
`lead`	integer sequence specifying the lead window, for which qoi point estimates (and standard errors) will ultimately be produced. Default is 0 (which corresponds to contemporaneous treatment effect).
`verbose`	option to include more information about the `matched.set` object calculations, like the distances used to create the refined sets and weights.
`exact.match.variables`	character vector giving the names of variables to be exactly matched on. These should be time invariant variables. Exact matching for time varying covariates is not currently supported.
`forbid.treatment.reversal`	Logical. For the ATT, it indicates whether or not it is permissible for treatment to reverse in the specified lead window. This is defined analogously for the ART. It is not valid for the ATC or ATE. When set to TRUE, only matched sets for treated units where treatment is applied continuously in the lead window are included in the results. Default is FALSE.
`matching`	logical indicating whether or not any matching on treatment history should be performed. This is primarily used for diagnostic purposes, and most users will never need to set this to FALSE. Default is TRUE.
`listwise.delete`	TRUE/FALSE indicating whether or not missing data should be handled using listwise deletion or the package's default missing data handling procedures. Default is FALSE.
`use.diagonal.variance.matrix`	TRUE/FALSE indicating whether or not a regular covariance matrix should be used in mahalanobis distance calculations during refinement, or if a diagonal matrix with only covariate variances should be used instead. In many cases, setting this to TRUE can lead to better covariate balance, especially when there is high correlation between variables. Default is FALSE. This argument is only necessary when `refinement.method = mahalanobis` and will have no impact otherwise.
`restrict.control.period`	(optional) integer specifying the number of pre-treatment periods that treated units and potentially matched control units should be non-NULL and in the

control state. For instance, specifying 4 would mean that the treatment history cannot contain any missing data or treatment from t-4 to t.

`placebo.test` logical TRUE/FALSE. indicates whether or not you want to be able to run a placebo test. This will add additional requirements on the data – specifically, it requires that no unit included in the matching/refinement process can having missing outcome data over the lag window. Additionally, you should not use the outcome variable in refinement when `placebo.test` = `TRUE`.

Value

`PanelMatch()` returns an object of class `PanelMatch`. This is a list that contains a few specific elements: First, a `matched.set` object(s) that has the same name as the provided qoi if the qoi is "att", "art", or "atc". If qoi = "ate" then two `matched.set` objects will be attached, named "att" and "atc." Please consult the documentation for `matched_set()` to read more about the structure and usage of `matched.set` objects. The `PanelMatch` object also has some additional attributes that track metadata about the specification, like the names of the unit and time identifier variables.

Author(s)

Adam Rauh <amrauh@umich.edu>, In Song Kim <insong@mit.edu>, Erik Wang <haixiao@Princeton.edu>, and Kosuke Imai <imai@harvard.edu>

References

Imai, Kosuke, In Song Kim, and Erik Wang (2023)

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
# include lagged variables
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.weight",
                         match.missing = TRUE,
                        covs.formula = ~ tradewb + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
```

placebo_test *Conduct a placebo test*

Description

Calculate the results of a placebo test, looking at the change in outcome at time = t-1, compared to other pre-treatment periods in the lag window.

Usage

```
placebo_test(
  pm.obj,
  panel.data,
  lag.in = NULL,
  number.iterations = 1000,
  confidence.level = 0.95,
  plot = FALSE,
  se.method = "bootstrap",
  parallel = FALSE,
  num.cores = 1,
  ...
)
```

Arguments

`pm.obj`	an object of class `PanelMatch`
`panel.data`	`PanelData` object
`lag.in`	integer indicating earliest the time period(s) in the future for which the placebo test change in outcome will be calculated. Calculations will be made over the period t - max(lag) to t-2, where t is the time of treatment. The results are similar to those returned by `PanelEstimate()`, except t-1 is used as the period of comparison, rather than the lead window. If not specified, the placebo test is conducted for periods from t - max(lag) to t-2.
`number.iterations`	integer specifying the number of bootstrap iterations. This argument only has an effect if standard errors are calculated with the bootstrap.
`confidence.level`	confidence level for the calculated standard error intervals. Should be specified as a numeric between 0 and 1.
`plot`	logical indicating whether or not a plot should be generated, or just return the raw data from the calculations
`se.method`	character string describing the type of standard error to be used. Valid inputs include "bootstrap", "conditional" and "unconditional". When the QOI is ATE, only bootstrap can be used. See the documentation of this argument in `PanelEstimate()` for more.

`parallel`	Logical. If TRUE and `se.method = "bootstrap"`, bootstrap procedure will be parallelized. Default is FALSE. If `se.method` is not set to `bootstrap`, this option does nothing."
`num.cores`	Integer. Specifies the number of cores to use for parallelization. If `se.method = "bootstrap"` and `parallel = TRUE`, then this option will take effect. Otherwise, it will do nothing.
`...`	extra arguments to be passed to `plot()`

Value

list with 2 or 3 elements: "estimate", which contains the point estimates for the test, "standard.errors" which has the standard errors for each period and optionally "bootstrapped.estimates", containing the bootstrapped point estimates for the test for each specified lag window period.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE, placebo.test = TRUE)
placebo_test(PM.results, panel.data = dem.sub.panel, se.method = "unconditional", plot = FALSE)
```

`plot.matched.set` *Plot the distribution of control unit weights*

Description

The method creates a heatmap with the following characteristics. The heatmap grid is m x n, where m is the number of treated observations (as identified by i,t pairs) and n is the number of units. Treated observations represent the rows, and every unit in the data set form the columns. The figure then shows the calculated weights or distances (as specified) for each control unit within the matched set as identified by the row. Weights/distances that are missing or zero are not considered in the shading scheme and are both treated as NA for all practical purposes. Note that not all refinement methods will return a distance. Those that do also require `verbose = TRUE` in the `PanelMatch` specification. For example say (2, 5) is a treated observation and units 1, 4, 8 are matched as controls. Row i will represent (2,5) in the matrix, M. The columns indexed by w, x, and y, correspond to units 1, 4, and 8. M[i, w], M[i, x], M[i, y] then contain the weights or pairwise distances of units 1, 4, and 8 within that matched set.

Usage

```
## S3 method for class 'matched.set'
plot(
  x,
  ...,
  panel.data,
  type = "weights",
  include.missing = TRUE,
  low.color = "blue",
  mid.color = "white",
  high.color = "red",
  missing.color = "grey50"
)
```

Arguments

`x`	a `matched.set` object
`...`	Not used
`panel.data`	a `PanelData` object
`type`	character indicating whether or not weights or distances should be plotted
`include.missing`	logical. When TRUE, all units appear as columns, including those that are never included in any matched sets. When FALSE, only units that appear in at least one matched set are included.
`low.color`	option passed to `ggplot2::scale_fill_gradientn()`. The color representing the low weight/distance values.
`mid.color`	option passed to `ggplot2::scale_fill_gradientn()`. The color representing the medium weight/distance values.
`high.color`	option passed to `ggplot2::scale_fill_gradientn()`. The color representing the high weight/distance values.
`missing.color`	option passed to `ggplot2::scale_fill_gradientn()`. The color representing the missing/zero weight/distance values.

Value

returns a `ggplot2::geom_tile()` object producing a plot in alignment with the description above

Examples

```
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
```

```
mso <- extract(PM.results)
plot(mso, panel.data = dem.panel)
```

`plot.PanelBalance` *Plot covariate balance results Create figures displaying covariate balance results for one or more* `PanelMatch` *configurations. Users can customize these visualizations.*

Description

Plot covariate balance results Create figures displaying covariate balance results for one or more `PanelMatch` configurations. Users can customize these visualizations.

Usage

```
## S3 method for class 'PanelBalance'
plot(
  x,
  ...,
  type = "panel",
  reference.line = TRUE,
  legend = TRUE,
  ylab = NULL,
  include.treatment.period = TRUE,
  include.unrefined.panel = TRUE,
  legend.position = "topleft"
)
```

Arguments

`x`	`PanelBalance` object
`...`	additional parameters to be passed to `base::plot()`
`type`	character specifying which type of plot to produce. Can be either "panel" or "scatter". When "panel," covariate balance results for covariates are shown over the lag period. When "scatter," the figure has the following characteristics. Each point on the plot represents a specific covariate at a particular time period in the lag window from t-L to t-1. The horizontal axis represents the covariate balance for this particular variable and time period before refinement is applied, while the vertical axis represents the post-refinement balance value.
`reference.line`	logical. Include a reference line at y = 0? Only applicable to the panel plot.
`legend`	logical. Describes whether or not to include a legend.
`ylab`	character. Y-axis label.

`include.treatment.period`
: Logical. Describes whether or not the treatment period should be included on the panel plot. Default is TRUE.

`include.unrefined.panel`
: logical indicating whether or not unrefined balance plots should be returned for panel plot. Only applicable to panel plot. Default is TRUE.

`legend.position`
: character. Describes where the legend should be placed on the figure. Uses base R syntax.

Value

returns a set of base R plots, depending on the specification of "panel" or "scatter" above. When `type = "panel"` and `include.unrefined.panel = TRUE`, two sets of plots are returned. The first set shows covariate balance levels for the specified `PanelMatch` configurations. The second set shows covariate balance levels for the same `PanelMatch` configurations, but with all control units receiving equal weight (i.e., balance levels prior to refinement). If `include.unrefined.panel = FALSE`, only the first set of figures are returned. The sets of figures are both returned in the same order as the `PanelMatch` configurations specified to `get_covariate_balance()` that compose the `PanelBalance` object. When `type = "scatter"`, the visualization described above is produced, with all configurations shown on the same plot with different symbols.

Examples

```
dem$rdata <- runif(runif(nrow(dem)))
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
pm.obj <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "mahalanobis",
                     panel.data = dem.panel, match.missing = TRUE,
                 covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                     size.match = 5, qoi = "att")

# create multiple configurations to compare
pm2 <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "ps.match",
                  panel.data = dem.panel, match.missing = TRUE,
                covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                  size.match = 5, qoi = "att")

pb <- get_covariate_balance(pm.obj, pm2,
                            include.unrefined = TRUE,
                            panel.data = dem.panel,
                            covariates = c("tradewb", "rdata"))
plot(pb, type = "panel", include.unrefined.panel = TRUE)
plot(pb, type = "scatter")
# only show refined balance figures
plot(pb, type = "panel", include.unrefined.panel = FALSE)
```

`plot.PanelData` *Create basic plots of PanelData objects*

Description

Create basic plots of PanelData objects

Usage

```
## S3 method for class 'PanelData'
plot(x, ..., plotting.variable = NA)
```

Arguments

`x`	`PanelData` object
`...`	Not used
`plotting.variable`	character string specifying which variable to plot in the resulting figure. The values of this variable will be used to fill in cells on the resulting heatmap. Defaults to whatever is specified as the treatment variable.

Value

Returns a ggplot2 object created by `geom_tile()`. The basic figure shows units along the y-axis and time along the x-axis. The figure takes the form of a heatmap. The value of the plotting.variable argument is used to fill in the color of the cells.

Examples

```
dem$rdata <- rnorm(nrow(dem))
d <- PanelData(dem, "wbcode2", "year", "dem", "y")
plot(d)
plot(d, plotting.variable = "rdata")
```

`plot.PanelEstimate` *Plot point estimates and standard errors from a PanelEstimate calculation.*

Description

The `plot.PanelEstimate` method takes an object returned by the `PanelEstimate` function and plots the calculated point estimates and standard errors over the specified `lead` time period. The only mandatory argument is an object of the `PanelEstimate` class.

Usage

```
## S3 method for class 'PanelEstimate'
plot(
  x,
  ylab = "Estimated Effect of Treatment",
  xlab = "Time",
  main = "Estimated Effects of Treatment Over Time",
  ylim = NULL,
  pch = NULL,
  cex = NULL,
  confidence.level = NULL,
  bias.corrected = FALSE,
  ...
)
```

Arguments

`x`	a `PanelEstimate` object
`ylab`	default is "Estimated Effect of Treatment." This is the same argument as the standard argument for `plot()`
`xlab`	default is "Time". This is the same argument as the standard argument for `plot()`
`main`	default is "Estimated Effects of Treatment Over Time". This is the same argument as the standard argument for `plot`
`ylim`	default is NULL. This is the same argument as the standard argument for `plot()`
`pch`	default is NULL. This is the same argument as the standard argument for `plot()`
`cex`	default is NULL. This is the same argument as the standard argument for `plot()`
`confidence.level`	confidence.level Confidence level to be used for confidence interval calculations. Must be numeric between 0 and 1. If NULL, confidence level from `PanelEstimate()` specification is used.
`bias.corrected`	logical indicating whether or not bias corrected estimates should be plotted Default is FALSE. This argument only applies for standard errors calculated with the bootstrap.
`...`	Additional optional arguments to be passed to `plot()`.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                            lead = 0:4,
                            forbid.treatment.reversal = FALSE)
PE.results <- PanelEstimate(sets = PM.results,
              panel.data = dem.sub.panel,
              se.method = "unconditional")
plot(PE.results)
```

`plot.PanelMatch` *Plot the distribution of the sizes of matched sets.*

Description

A plot method for creating a histogram of the distribution of the sizes of matched sets. This method accepts all standard optional `hist` arguments via the `...` argument. By default, empty matched sets (treated units that could not be matched with any control units) are noted as a vertical bar at x = 0 and not included in the regular histogram. See the `include.empty.sets` argument for more information about this. If the quantity of interest is ATE, a plot will be returned for the matched sets associated with the att and the atc.

Usage

```
## S3 method for class 'PanelMatch'
plot(
  x,
  ...,
  border = NA,
  col = "grey",
  ylab = "Frequency of Size",
  xlab = "Matched Set Size",
  lwd = NULL,
  main = "Distribution of Matched Set Sizes",
  freq = TRUE,
  include.empty.sets = FALSE
)
```

Arguments

`x`	a `PanelMatch` object
`...`	optional arguments to be passed to `hist()`
`border`	default is NA. This is the same argument as the standard argument for `hist()`
`col`	default is "grey". This is the same argument as the standard argument for `hist()`
`ylab`	default is "Frequency of Size". This is the same argument as the standard argument for `hist()`
`xlab`	default is "Matched Set Size". This is the same argument as the standard argument for `hist()`

`lwd`	default is NULL. This is the same argument as the standard argument for `hist()`
`main`	default is "Distribution of Matched Set Sizes". This is the same argument as the standard argument for `hist`
`freq`	default is TRUE. See `freq` argument in `hist()` function for more.
`include.empty.sets`	logical value indicating whether or not empty sets should be included in the histogram. default is FALSE. If FALSE, then empty sets will be noted as a separate vertical bar at x = 0. If TRUE, empty sets will be included as normal sets.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.sub.panel,
                         lag = 4,
                         refinement.method = "mahalanobis",
                         match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4, forbid.treatment.reversal = FALSE)
plot(PM.results)
plot(PM.results, include.empty.sets = TRUE)
```

`print.matched.set` *Print matched.set objects.*

Description

Print matched.set objects.

Usage

```
## S3 method for class 'matched.set'
print(x, ..., verbose = FALSE)
```

Arguments

`x`	a `matched.set` object
`...`	Not used. additional arguments to be passed to `print`
`verbose`	logical indicating whether or not output should be printed in expanded/raw list form. The verbose form is not recommended unless the data set is small. Default is FALSE, which prints an overview of matched set sizes.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
# create subset of data for simplicity
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
print(extract(PM.results, qoi = "att"))
```

`print.PanelBalance` *Print basic information about PanelBalance objects*

Description

Print basic information about PanelBalance objects

Usage

```
## S3 method for class 'PanelBalance'
print(x, ...)
```

Arguments

`x`	`PanelBalance` object
`...`	Not used

Value

Nothing

Examples

```
dem$rdata <- runif(runif(nrow(dem)))
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
pm.obj <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "mahalanobis",
                     panel.data = dem.panel, match.missing = TRUE,
                 covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                     size.match = 5, qoi = "att")
pb <- get_covariate_balance(pm.obj,
                            include.unrefined = TRUE,
                            panel.data = dem.panel,
                            covariates = c("tradewb", "rdata"))
print(pb)
```

print.PanelData — *Print PanelData objects and basic metadata*

Description

Print PanelData objects and basic metadata

Usage

```
## S3 method for class 'PanelData'
print(x, ...)
```

Arguments

x	`PanelData` object
...	additional arguments to be passed to `print.data.frame()`

Examples

```
d <- PanelData(dem, "wbcode2", "year", "dem", "y")
print(d)
```

print.PanelEstimate — *Print point estimates and standard errors*

Description

Print point estimates and standard errors

Usage

```
## S3 method for class 'PanelEstimate'
print(x, ...)
```

Arguments

x	`PanelEstimate` object
...	additional arguments to be passed to `print.data.frame()`

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
PE.results <- PanelEstimate(sets = PM.results,
              panel.data = dem.sub.panel,
              se.method = "unconditional")
print(PE.results)
```

`print.PanelMatch` *Print PanelMatch objects.*

Description

Print PanelMatch objects.

Usage

```
## S3 method for class 'PanelMatch'
print(x, ..., verbose)
```

Arguments

x	a `PanelMatch` object
...	additional arguments to be passed to `print`
verbose	logical indicating whether or not underlying data should be printed in expanded/raw list form. The verbose form is not recommended unless the data set is small. Default is FALSE

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem, 'wbcode2', 'year', 'dem', 'y')
PM.results <- PanelMatch(panel.data = dem.sub.panel,
                         lag = 4,
                         refinement.method = "mahalanobis",
                         match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4, forbid.treatment.reversal = FALSE)
print(PM.results)
```

summary.matched.set *Summarize information about a matched.set object and the matched sets contained within them.*

Description

A method for viewing summary data about the sizes of matched sets and metadata about how they were created. This method accepts all standard `summary` arguments.

Usage

```
## S3 method for class 'matched.set'
summary(object, ..., verbose = TRUE)
```

Arguments

`object`	a `matched.set` object
`...`	Optional additional arguments to be passed to the `summary` function
`verbose`	Logical value specifying whether or not a longer, more verbose summary should be calculated and returned. Default is TRUE.

Value

list object with either 5 or 1 element(s), depending on whether or not `verbose` is set to TRUE or not.

`overview`	A `data.frame` object containing information about the treated units (unit id, time of treatment), and the number of matched control units with weights zero and above.
`set.size.summary`	a `summary` object summarizing the minimum, maximum, and IQR of matched set sizes
`number.of.treated.units`	The number of unit, time pairs that are considered to be "treated" units
`num.units.empty.set`	The number of units treated at a particular time that were not able to be matched to any control units
`lag`	The size of the lag window used for matching on treatment history. This affects which treated and control units are matched.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(lag = 4, refinement.method = "ps.match",
                         panel.data = dem.sub.panel, match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4, forbid.treatment.reversal = FALSE)
summary(extract(PM.results, qoi = "att"))
```

---

summary.PanelBalance *Summarize covariate balance over time*

Description

Summarize covariate balance over time

Usage

```
## S3 method for class 'PanelBalance'
summary(
  object,
  qoi = NULL,
  include.unrefined = TRUE,
  unrefined.only = FALSE,
  ...
)
```

Arguments

`object`	`PanelBalance` object
`qoi`	Character. Valid values include "att", "art", or "atc". Specifying which QOI information to extract and summarize.
`include.unrefined`	logical. Indicates whether or not unrefined balance results should be included in the summary.
`unrefined.only`	logical. Indicates whether or not only unrefined balance results should be included in the summary.
`...`	Not used

Value

returns a list of matrices with covariate balance results calculated. Each element in the list corresponds to a `PanelMatch` configuration given to `get_covariate_balance()` and are returned in order. Note that if a configuration has `qoi = "ate"`, the corresponding element in the returned list will also be a list, containing balance results corresponding to the ATT and ATC. Otherwise, each element in the returned list will be a matrix. Each matrix entry corresponds to balance results for a particular covariate in a particular period. When unrefined balance results are included, users will see additional columns with "_unrefined" appended to covariate names. These correspond to the unrefined balance results for a particular covariate-period.

Examples

```
dem$rdata <- runif(runif(nrow(dem)))
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
pm.obj <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "mahalanobis",
                     panel.data = dem.panel, match.missing = TRUE,
                 covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                     size.match = 5, qoi = "att")
pb <- get_covariate_balance(pm.obj,
                            include.unrefined = TRUE,
                            panel.data = dem.panel,
                            covariates = c("tradewb", "rdata"))
summary(pb)
```

`summary.PanelData` *Summarize the number of unique units and time periods in a PanelData object*

Description

Summarize the number of unique units and time periods in a PanelData object

Usage

```
## S3 method for class 'PanelData'
summary(object, ...)
```

Arguments

`object`	PanelData object
`...`	Not used

Value

Returns a `data.frame` object, with columns "num.units" and "num.periods." These specify the number of unique units and time periods that appear in the balanced panel data.

Examples

```
d <- PanelData(dem, "wbcode2", "year", "dem", "y")
summary(d)
```

`summary.PanelEstimate` *Get summaries of PanelEstimate objects and calculations*

Description

`summary.PanelEstimate` takes an object returned by `PanelEstimate`, and returns a summary table of point estimates and confidence intervals

Usage

```
## S3 method for class 'PanelEstimate'
summary(
  object,
  confidence.level = NULL,
  verbose = FALSE,
  bias.corrected = FALSE,
  ...
)
```

Arguments

`object`	A `PanelEstimate` object
`confidence.level`	Confidence level to be used for confidence interval calculations. Must be numeric between 0 and 1. If NULL, confidence level from `PanelEstimate()` specification is used.
`verbose`	logical indicating whether or not output should be printed in an expanded form. Default is FALSE
`bias.corrected`	logical indicating whether or not bias corrected estimates should be provided. Default is FALSE. This argument only applies for standard errors calculated with the bootstrap.
`...`	optional additional arguments. Currently, no additional arguments are supported.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
# create subset of data for simplicity
PM.results <- PanelMatch(panel.data = dem.sub.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
PE.results <- PanelEstimate(sets = PM.results,
                            panel.data = dem.sub.panel,
                         se.method = "unconditional")
summary(PE.results)
summary(PE.results, confidence.level = .9)
```

`summary.PanelMatch` *Summarize information about a PanelMatch object and the matched sets contained within them.*

Description

A method for viewing summary data about the sizes of matched sets and metadata about how they were created. This method accepts all standard `summary` arguments. If the quantity of interest is ate, then a summary will be provided for the matched sets associated with the att and the atc.

Usage

```
## S3 method for class 'PanelMatch'
summary(object, ..., verbose = FALSE)
```

Arguments

`object`	a `PanelMatch` object
`...`	Optional additional arguments to be passed to the `summary` function
`verbose`	Logical value specifying whether or not a longer, more verbose summary should be calculated and returned. Default is `FALSE`.

Value

A list of lists containing a summary of the matched sets associated with the specified qoi. Each sublist object will either have 5 or 1 element(s), depending on whether or not `verbose` is set to `TRUE` or not.

`overview`	A `data.frame` object containing information about the treated units (unit id, time of treatment), and the number of matched control units with weights zero and above.
`set.size.summary`	a `summary` object summarizing the minimum, maximum, and IQR of matched set sizes
`number.of.treated.units`	The number of unit, time pairs that are considered to be "treated" units
`num.units.empty.set`	The number of units treated at a particular time that were not able to be matched to any control units
`lag`	The size of the lag window used for matching on treatment history. This affects which treated and control units are matched.

Examples

```
dem.sub <- dem[dem[, "wbcode2"] <= 100, ]
dem.sub.panel <- PanelData(dem.sub, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.sub.panel,
                         lag = 4,
                         refinement.method = "mahalanobis",
                         match.missing = TRUE,
                         covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                         size.match = 5, qoi = "att",
                         lead = 0:4, forbid.treatment.reversal = FALSE)
summary(PM.results)
```

`weights` *Get weights of matched control units See weights.matched.set method*

Description

Get weights of matched control units See weights.matched.set method

Usage

```
weights(object)
```

Arguments

`object`	`Matched.set` object

`weights.matched.set` *Extract the weights of matched control units*

Description

Extract the weights of matched control units

Usage

```
## S3 method for class 'matched.set'
weights(object)
```

Arguments

`object`	matched.set object, extracted using the `get.PanelMatch()` method

Value

list of named vectors. Each list element corresponds to a particular treated observation and contains the matched control units, along with their weights. These correspond to the "weights" attribute, which are calculated in the `PanelMatch` refinement process.

Examples

```
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
PM.results <- PanelMatch(panel.data = dem.panel, lag = 4,
                         refinement.method = "ps.match",
                         match.missing = TRUE,
                         covs.formula = ~ tradewb,
                         size.match = 5, qoi = "att",
                         lead = 0:4,
                         forbid.treatment.reversal = FALSE)
r1 <- extract(PM.results, qoi = "att")
lt <- weights(r1)
```

`[.matched.set` *Subset matched.set object*

Description

Subsets `matched.set` objects while preserving attributes.

Usage

```
## S3 method for class 'matched.set'
x[i, j = NULL, drop = NULL]
```

Arguments

x	`matched.set` object
i	numeric. specifies the index of which element to extract.
j	NULL
drop	NULL

`[.PanelBalance` *Subset PanelBalance objects*

Description

Subset PanelBalance objects

Usage

```
## S3 method for class 'PanelBalance'
x[i, ...]
```

Arguments

`x`	`PanelBalance` object
`i`	numeric. Specifies which element to extract. Substantively, it specifies which `PanelMatch` configuration data to extract.
`...`	Not used

Value

Returns balance information for specified `PanelMatch` configuration. Note that results are still returned as a `PanelBalance` object. In order to return a list, use the [[operator

Examples

```
dem$rdata <- runif(runif(nrow(dem)))
dem.panel <- PanelData(dem, "wbcode2", "year", "dem", "y")
pm.obj <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "mahalanobis",
                     panel.data = dem.panel, match.missing = TRUE,
                 covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                     size.match = 5, qoi = "att")

# create multiple configurations to compare
pm2 <- PanelMatch(lead = 0:3, lag = 4, refinement.method = "ps.match",
                  panel.data = dem.panel, match.missing = TRUE,
                covs.formula = ~ tradewb + rdata + I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                  size.match = 5, qoi = "att")

pb <- get_covariate_balance(pm.obj, pm2,
                            include.unrefined = TRUE,
                            panel.data = dem.panel,
                            covariates = c("tradewb", "rdata"))
bal.maha <- pb[1]
bal.ps <- pb[2]
```

Index