Package 'blockTools'

May 7, 2026

Type Package

Title Block, Assign, and Diagnose Potential Interference in Randomized Experiments

Version 0.6.6

Date 2025-01-21

Description Blocks units into experimental blocks, with one unit per treatment condition, by creating a measure of multivariate distance between all possible pairs of units. Maximum, minimum, or an allowable range of differences between units on one variable can be set. Randomly assign units to treatment conditions. Diagnose potential interference between units assigned to different treatment conditions. Write outputs to .tex and .csv files. For more information on the methods implemented, see Moore (2012) <doi:10.1093/pan/mps025>.

License GPL (>= 2) | file LICENSE

Encoding UTF-8

Imports dplyr, MASS, tibble

Suggests nbpMatching, RItools, testthat (>= 3.0.0), xtable

Depends R (>= 3.5.0)

Config/testthat/edition 3

URL https://www.ryantmoore.org/html/software.blockTools.html

RoxygenNote 7.3.2

NeedsCompilation yes

Author Ryan T. Moore [aut, cre] (ORCID: <https://orcid.org/0000-0002-3916-8113>), Keith Schnakenberg [aut]

Maintainer Ryan T. Moore <rtm@american.edu>

Repository CRAN

Date/Publication 2025-01-22 08:10:05 UTC

Contents

`blockTools-package` *Block, Randomly Assign, and Diagnose Potential Interference in Randomized Experiments*

Description

Block units into experimental blocks, with one unit per treatment condition, by creating a measure of multivariate distance between all possible pairs of units. Maximum, minimum, or an allowable range of differences between units on one variable can be set. Randomly assign units to treatment conditions. Diagnose potential interference problems between units assigned to different treatment conditions. Write outputs to .tex and .csv files.

Details

Given raw data, `block` creates experimental blocks, `assignment` assigns units to treatment conditions, `diagnose` detects possible interference problems, and `outTeX` and `outCSV` write block or assignment output objects to a set of .tex and .csv files, respectively. In sequential experiments, `seqblock` assigns units to treatment conditions.

Author(s)

Ryan T. Moore [aut, cre] (<rtm@american.edu>), Keith Schnakenberg [aut] (<keith.schnakenberg@gmail.com>)

References

https://www.ryantmoore.org/html/software.blockTools.html

Moore, Ryan T. Multivariate Continuous Blocking to Improve Political Science Experiments. Political Analysis, 20(4):460-479, 2012.

Moore, Ryan T. and Sally A. Moore. Blocking for Sequential Political Experiments. Political Analysis, 21(4):507-523, 2013.

See Also

Useful links:

- https://www.ryantmoore.org/html/software.blockTools.html

Examples

```
data(x100)

# block
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"),
             block.vars = c("b1", "b2"), algorithm = "optGreedy",
             distance = "mahalanobis", level.two = FALSE, valid.var = "b1",
             valid.range = c(0,500), verbose = TRUE)

# assign
assg <- assignment(out, seed = 123)

# diagnose
diag <- diagnose(object = assg, data = x100, id.vars = "id",
                 suspect.var = "b2", suspect.range = c(0,50))

# create .tex files of assigned blocks
# outTeX(assg)

# create .csv files of unassigned blocks
# outCSV(out)

# create block IDs
createBlockIDs(out, x100, id.var = "id")

# block ID integers are unique, even with several groups
axb <- assg2xBalance(assg, x100, id.var = "id", bal.vars = c("b1", "b2"))
```

`assg2xBalance` *Calculate balance statistics from an assignment object*

Description

Calculate several balance statistics for experimental units assigned to treatment conditions. Naturally accepts output from the `assignment` function, and passes it to `xBalance` from `library(RItools)`. Provides balance summaries for the entire experiment and by group.

Usage

```
assg2xBalance(assg.obj, data, id.var, bal.vars, to.report = "all")
```

Arguments

assg.obj	an output object from assignment.
data	the data frame that was input into block for blocking.
id.var	a string specifying the column of data containing identifying information.
bal.vars	a string or vector of strings specifying which column(s) of data contain the variables on which balance is to be checked.
to.report	a string or vector of strings passed to xBalance listing the measures to report for each group. See Details for more information.

Details

As of RItools version 0.1-11, to.report must be a subset of c("std.diffs","z.scores","adj.means", "adj.mean.diffs","adj.mean.diffs.null.sd", "chisquare.test","p.values", "all"). The default, all, returns all measures.

Value

A list of output objects from xBalance. For each group defined in the assignment object, one list element is assigned the name of that group and summarizes the balance in that group according to to.report. The last element of the list is named "Overall" and summarizes balance across all groups. The elements of this list are themselves objects of class c("xbal", "list").

If assg.obj has only one group, the first element of the output list is named "Group1", and the second is named "Output". In this case, these two elements will be identical.

Author(s)

Ryan T. Moore

References

Hansen, Ben B. and Jake Bowers. 2008. "Covariate balance in simple, stratified and clustered comparative studies". *Statistical Science* 23(2):219–236.

Bowers, Jake and Mark Fredrickson and Ben Hansen. 2010. "RItools:Randomization Inference Tools". R package version 0.1-11.

Moore, Ryan T. 2012. "Multivariate Continuous Blocking to Improve Political Science Experiments". *Political Analysis*, 20(4):460–479, Autumn.

See Also

assignment

Examples

```
data(x100)
b <- block(x100, groups = "g", id.vars = "id", block.vars = c("b1", "b2"))
a <- assignment(b)
axb <- assg2xBalance(a, x100, id.var = "id", bal.vars = c("b1", "b2"))
```

```
axb
# axb is a list with 4 elements (one for each of 3 groups, plus one for 'Overall')
```

`assignment` *Randomly assign blocked units to treatment conditions*

Description

Using an output object from `block`, assign elements of each row to treatment condition columns. Each element is equally likely to be assigned to each column.

Usage

```
assignment(block.obj, seed = NULL, namesCol = NULL)
```

Arguments

`block.obj`	an output object from `block`, or a user-specified block object.
`seed`	a user-specified random seed.
`namesCol`	an optional vector of column names for the output table.

Details

`block.obj` can be specified directly by the user. It can be a single dataframe or matrix with blocks as rows and treatment conditions as columns. `assignment` is designed to take a list with two elements. The first element should be named `$blocks`, and should be a list of dataframes. Each dataframe should have blocks as rows and treatment conditions as columns. The second element should be a logical named `$level.two`. A third element, such as `$call` in a `block` output object, is currently ignored.

Specifying the random seed yields constant assignment, and thus allows for easy replication of experimental protocols.

If `namesCol = NULL`, then "Treatment 1", "Treatment 2", ... are used. If `namesCol` is supplied by the user and is of length `n.tr` (or 2*`n.tr`, where `level.two = TRUE`), then either `"Distance"` or `"Max Distance"` is appended to it as appropriate (consistent with `namesCol` usage in `block`). If `namesCol` is supplied and is of length `n.tr + 1` (or 2 * `n.tr + 1`, where `level.two = TRUE`), then the last user-supplied name is used for the last column of each dataframe.

Value

A list with elements

- **assg**: a list of dataframes, each containing a group's blocked units assigned to treatment conditions. If there are two treatment conditions, then the last column of each dataframe displays the multivariate distance between the two units. If there are more than two treatment conditions, then the last column of each dataframe displays the largest of the multivariate distances between all possible pairs in the block.
- **call**: the original call to `assignment`.

Author(s)

Ryan T. Moore

See Also

block, diagnose

Examples

```
data(x100)

# First, block
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"), block.vars
          = c("b1", "b2"), algorithm = "optGreedy", distance = "mahalanobis",
          level.two = FALSE, valid.var = "b1", valid.range = c(0,500),
          verbose = TRUE)

# Second, assign
assigned <- assignment(out, seed = 123)

# assigned$assg contains 3 data frames
```

---

block *Block units into homogeneous experimental blocks*

Description

Block units into experimental blocks, with one unit per treatment condition. Blocking begins by creating a measure of multivariate distance between all possible pairs of units. Maximum, minimum, or an allowable range of differences between units on one variable can be set.

Usage

```
block(
  data,
  vcov.data = NULL,
  groups = NULL,
  n.tr = 2,
  id.vars,
  block.vars = NULL,
  algorithm = "optGreedy",
  distance = "mahalanobis",
  weight = NULL,
  optfactor = 10^7,
  row.sort = NULL,
  level.two = FALSE,
  valid.var = NULL,
```

```
  valid.range = NULL,
  seed.dist,
  namesCol = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

data	a dataframe or matrix, with units in rows and variables in columns.
vcov.data	an optional matrix of data used to estimate the variance-covariance matrix for calculating multivariate distance.
groups	an optional column name from data, specifying subgroups within which blocking occurs.
n.tr	the number of treatment conditions per block.
id.vars	a required string or vector of two strings specifying which column(s) of data contain identifying information.
block.vars	an optional string or vector of strings specifying which column(s) of data contain the numeric blocking variables.
algorithm	a string specifying the blocking algorithm. "optGreedy", "optimal", "naiveGreedy", "randGreedy", and "sortGreedy" algorithms are currently available. See Details for more information.
distance	either a) a string defining how the multivariate distance used for blocking is calculated (options include "mahalanobis", "mcd", "mve", and "euclidean"), or b) a user-defined k-by-k matrix of distances, where k is the number of rows in data.
weight	either a vector of length equal to the number of blocking variables or a square matrix with dimensions equal to the number of blocking variables used to explicitly weight blocking variables.
optfactor	a number by which distances are multiplied then divided when algorithm = "optimal".
row.sort	an optional vector of integers from 1 to nrow(data) used to sort the rows of data when algorithm = "sortGreedy".
level.two	a logical defining the level of blocking.
valid.var	an optional string defining a variable on which units in the same block must fall within the range defined by valid.range.
valid.range	an optional vector defining the range of valid.var within which units in the same block must fall.
seed.dist	an optional integer value for the random seed set in cov.rob, used to calculate measures of the variance-covariance matrix robust to outliers.
namesCol	an optional vector of column names for the output table.
verbose	a logical specifying whether groups names and block numbers are printed as blocks are created.
...	additional arguments passed to cov.rob.

Details

If vcov.data = NULL, then block calculates the variance-covariance matrix using the block.vars from data.

If groups is not user-specified, block temporarily creates a variable in data called groups, which takes the value 1 for every unit.

Where possible, one unit is assigned to each condition in each block. If there are fewer available units than treatment conditions, available units are used.

If n.tr > 2, then the optGreedy algorithm finds the best possible pair match, then the best match to either member of the pair, then the best match to any member of the triple, etc. After finding the best pair match to a given unit, the other greedy algorithms proceed by finding the third, fourth, etc. best match to that given unit.

An example of id.vars is id.vars = c("id", "id2"). If two-level blocking is selected, id.vars should be ordered (unit id, subunit id). See details for level.two below for more information.

If block.vars = NULL, then all variables in data except the id.vars are taken as blocking variables. E.g., block.vars = c("b1", "b2").

The algorithm optGreedy calls an optimal-greedy algorithm, repeatedly finding the best remaining match in the entire dataset; optimal finds the set of blocks that minimizes the sum of the distances in all blocks; naiveGreedy finds the best match proceeding down the dataset from the first unit to the last; randGreedy randomly selects a unit, finds its best match, and repeats; sortGreedy resorts the dataset according to row.sort, then implements the naiveGreedy algorithm.

The optGreedy algorithm breaks ties by randomly selecting one of the minimum-distance pairs. The naiveGreedy, sortGreedy, and randGreedy algorithms break ties by randomly selecting one of the minimum-distance matches to the particular unit in question.

As of version 0.5-1, blocking is done in C for all algorithms except optimal (see following paragraphs for more details on the optimal algorithm implementation).

The optimal algorithm uses two functions from the **nbpMatching** package: distancematrix prepares a distance matrix for optimal blocking, and nonbimatch performs the optimal blocking by minimizing the sum of distances in blocks. nonbimatch, and thus the block algorithm optimal, requires that n.tr = 2.

Because distancematrix takes the integer floor of the distances, and one may want much finer precision, the multivariate distances calculated within block are multiplied by optfactor prior to optimal blocking. Then distancematrix prepares the resulting distance matrix, and nonbimatch is called on the output. The distances are then untransformed by dividing by optfactor before being returned by block.

The choice of optfactor can determine whether the Fortran code can allocate enough memory to solve the optimization problem. For example, blocking the first 14 units of x100 by executing block(x100[1:14,], id.vars = "id", block.vars = c("b1", "b2"), algorithm = "optimal", optfactor = 10^8) fails for Fortran memory reasons, while the same code with optfactor = 10^5 runs successfully. Smaller values of optfactor imply easier computation, but less precision.

Most of the algorithms in block make prohibited blockings by using a distance of Inf. However, the optimal algorithm calls Fortran code from **nbpMatching** and requires integers. Thus, a distance of 99999 * max(dist.mat) is used to effectively prohibit blockings. This follows the procedure demonstrated in the example of help(nonbimatch).

In order to enable comparisons of block-quality across groups, when distance is a string, $Sigma$ is calculated using units from all groups.

The distance = "mcd" and distance = "mve" options call cov.rob to calculate measures of multivariate spread robust to outliers. The distance = "mcd" option calculates the Minimum Covariance Determinant estimate (Rousseeuw 1985); the distance = "mve" option calculates the Minimum Volume Ellipsoid estimate (Rousseeuw and van Zomeren 1990). When distance = "mcd", the interquartile range on blocking variables should not be zero.

A user-specified distance matrix must have diagonals equal to 0, indicating zero distance between a unit and itself. Only the lower triangle of the matrix is used.

If weight is a vector, then it is used as the diagonal of a square weighting matrix with non-diagonal elements equal to zero. The weighting is done by using as the Mahalanobis distance scaling matrix $((((chol(Sigma))')^{-1})'W((chol(Sigma))')^{-1})^{-1}$, where $chol(Sigma)$ is the Cholesky decomposition of the usual variance-covariance matrix and W is the weighting matrix. Differences should be smaller on covariates given higher weights.

If level.two = TRUE, then the best subunit block-matches in different units are found. E.g., provinces could be matched based on the most similar cities within them. All subunits in the data should have unique names. Thus, if subunits are numbered 1 to (number of subunits in unit) within each unit, then they should be renumbered, e.g., 1 to (total number of subunits in all units). level.two blocking is not currently implemented for algorithm = "optimal". Units with no blocked subunit are put into their own blocks. However, unblocked subunits within a unit that does have a blocked subunit are not put into their own blocks.

An example of a variable restriction is valid.var = "b2", valid.range = c(10,50), which requires that units in the same block be at least 10 units apart, but no more than 50 units apart, on variable "b2". As of version 0.5-3, variable restrictions are implemented in all algorithms except optimal. Note that employing a variable restriction may result in fewer than the maximum possible number of blocks. See https://www.ryantmoore.org/html/software.blockTools.html for details.

If namesCol = NULL, then "Unit 1", "Unit 2", ... are used. If level.two = FALSE, then namesCol should be of length n.tr; if level.two = TRUE, then namesCol should be of length 2*n.tr, and in the order shown in the example below.

Value

A list with elements

- **blocks**: a list of dataframes, each containing a group's blocked units. If there are two treatment conditions, then the last column of each dataframe displays the multivariate distance between the two units. If there are more than two treatment conditions, then the last column of each dataframe displays the largest of the multivariate distances between all possible pairs in the block.
- **level.two**: a logical indicating whether level.two = TRUE.
- **call**: the original call to block.

Author(s)

Ryan T. Moore <rtm@american.edu> and Keith Schnakenberg <keith.schnakenberg@gmail.com>

References

King, Gary, Emmanuela Gakidou, Nirmala Ravishankar, Ryan T. Moore, Jason Lakin, Manett Vargas, Martha Mar\'ia T\'ellez-Rojo and Juan Eugenio Hern\'andez \'Avila and Mauricio Hern\'andez \'Avila and H\'ector Hern\'andez Llamas. 2007. "A 'Politically Robust' Experimental Design for Public Policy Evaluation, with Application to the Mexican Universal Health Insurance Program". *Journal of Policy Analysis and Management* 26(3): 479-509.

Moore, Ryan T. 2012. "Multivariate Continuous Blocking to Improve Political Science Experiments." *Political Analysis* 20(4):460-479.

Rousseeuw, Peter J. 1985. "Multivariate Estimation with High Breakdown Point". *Mathematical Statistics and Applications* 8:283-297.

Rousseeuw, Peter J. and Bert C. van Zomeren. 1990. "Unmasking Multivariate Outliers and Leverage Points". *Journal of the American Statistical Association* 85(411):633-639.

See Also

assignment, diagnose

Examples

```
data(x100)
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"),
             block.vars = c("b1", "b2"), algorithm = "optGreedy",
             distance = "mahalanobis", level.two = FALSE, valid.var = "b1",
             valid.range = c(0, 500), verbose = TRUE)

# out$blocks contains 3 data frames

# To illustrate two-level blocking, with multiple level two units per level one unit:

for(i in (1:nrow(x100))){if((i %% 2) == 0){x100$id[i] <- x100$id[i-1]}}

out2 <- block(x100, groups = "g", n.tr = 2, id.vars = c("id", "id2"),
              block.vars = c("b1", "b2"), algorithm = "optGreedy",
              distance = "mahalanobis", level.two = TRUE, valid.var = "b1",
              valid.range = c(0,500), namesCol = c("State 1", "City 1",
              "State 2", "City 2"), verbose = TRUE)
```

---

block2seqblock *Prepare prior nonsequential assignments for subsequent sequential assignments*

Description

Converts output objects from the block and assignment functions into an object in the format of one output by the seqblock function. This allows the user to block and assign multiple units at the beginning of an experiment (using block and assignment) and then sequentially block and assign more units to the experiment over time (using seqblock).

Usage

```
block2seqblock(
  block.obj,
  assg.obj,
  data,
  exact.restr = NULL,
  covar.restr = NULL,
  covar.order = NULL,
  trn = NULL,
  apstat = "mean",
  mtrim = 0.1,
  apmeth = "ktimes",
  kfac = 2,
  assgpr = c(0.5, 0.5),
  distance = NULL,
  datetime = NULL,
  orig,
  seed = NULL,
  file.name = "sbout.RData",
  verbose = FALSE
)
```

Arguments

block.obj	an output object from block, or a user-specified block object
assg.obj	an output object from assignment, or a user-specified assignment object
data	a matrix or dataframe containing the original data used to block the units in the study
exact.restr	a list object containing the restricted values that the exact blocking variables can take on. Thus the first element of exact.restr is a vector containing all of the possible values that the first exact blocking variable can take on; the second element is a vector containing all of the possible values for the second exact blocking variable; and so on
covar.restr	a list object containing the restricted values that the non-exact blocking variables can take on. Thus the first element of covar.restr is a vector containing all of the possible values that the first non-exact blocking variable can take on; the second element is a vector containing all of the possible values for the second non-exact blocking variable; and so on
covar.order	a string or vector of strings containing the name of the non-exact blocking variables ordered so that the highest priority covariate comes first, followed by the second highest priority covariate, then the third, etc.
trn	a string or vector of strings containing the names of the different treatment groups
apstat	a string specifying the assignment probability summary statistic that was used
mtrim	a numeric value specifying the proportion of observations to be dropped when the assignment probability statistic takes on the value "trimmean".

apmeth	a string specifying the assignment probability algorithm that was used.
kfac	the assignment probability *kfactor*; see *assg.prob.kfac* in the Arguments section above
assgpr	a vector of assignment probabilities to each treatment group
distance	a string specifying how the multivariate distance used for blocking is calculated
datetime	the date and time that the units were assigned to the treatment group; by default this is set to be a vector of NA; however the user could also specify a specific datetime and all of the units from the block object will be given the same datetime stamp
orig	a dataframe containing the names and values for the different id and blocking variables, as well as each unit's initial treatment assignment
seed	an optional integer value for the random seed set which is used when assigning units to treatment groups
file.name	a string containing the name of the file that one would like the output to be written to. Ideally this file name should have the extension .RData
verbose	a logical stating whether the function should print the name of the output file, the current working directory, and the dataframe x returned by the function as part of the bdata list

Details

The function converts data from a blocked experiment into a form allowing subsequent sequential blocking. Minimally, the user sets only the arguments block.obj, assg.obj and data. Then, block2seqblock uses the call to block, the assignment object, and the original data to create an object that is ready to be input into seqblock.

If the user explicitly specifies groups, id.vars and block.vars in the initial block function that is used to create the block.obj, block2seqblock will order the variables in the output it produces according to the order specified in the initial block function call. If the user does not explicitly specify the blocking variables in the block function call, block2seqblock will order the variables according to the order in the initial matrix or dataframe that was used to run the original block function.

As part of the function, variables that are of class factor in the original matrix or dataframe specified in data, will be converted into class character.

The trn argument uses the n.tr argument from block to extract the names of the treatment variables. Most other arguments are set to default values that mirror those in the seqblock function. One exception is the datetime argument, which defaults to a vector of NA's instead of the current datetime.

Value

A list (called bdata) with elements

- **x**: a dataframe containing the names and values for the different ID and blocking variables, as well as each unit's initial treatment assignment.
- **nid**: a string or vector of strings containing the name(s) of the ID variable(s).

- **nex**: a string or vector of strings containing the name(s) of the exact blocking variable(s).
- **ncv**: a string or vector of strings containing the name(s) of the non-exact blocking variable(s).
- **rex**: a list of the restricted values of the exact blocking variables.
- **rcv**: a list of the restricted values of the non-exact blocking variables.
- **ocv**: a vector of the order of the non-exact blocking variables.
- **trn**: a string or vector of strings containing the name(s) of the different treatment groups.
- **apstat**: a string specifying the assignment probability summary statistic that was used.
- **mtrim**: a numeric value specifying the proportion of observations to be dropped when the assignment probability statistic takes on the value `"trimmean"`.
- **apmeth**: a string specifying the assignment probability algorithm that was used.
- **kfac**: the assignment probability *kfactor*; see *assg.prob.kfac* in the Arguments section above.
- **assgpr**: a vector of assignment probabilities to each treatment group.
- **distance**: a string specifying how the multivariate distance used for blocking is calculated
- **trd**: a list with the length equal to the number of previously assigned treatment conditions; each object in the list contains a vector of the distance between each unit in one treatment group and the new unit. Set to `NULL` when there are no non-exact blocking variables.
- **tr.sort**: a string vector of treatment conditions, sorted from the largest to the smallest
- **p**: a vector of assignment probabilities to each treatment group used in assigning a treatment condition to the new unit.
- **trcount**: a table containing the counts for each experimental/treatment conditions.
- **datetime**: the date and time that the user was assigned a treatment group.
- **orig**: a dataframe containing the names and values for the different id and blocking variables, as well as each unit's initial treatment assignment.

Author(s)

Tommy Carroll <tcarroll22@wustl.edu>, Jonathan Homola <homola@wustl.edu>, and Ryan T. Moore <rtm@american.edu>

See Also

`block`, `assignment`, `seqblock`

Examples

```
# data(x100)
# out <- block(x100, n.tr = 2, id.vars = c("id"), block.vars = c("b1", "b2"),
#              algorithm = "optGreedy", distance = "mahalanobis",
#              valid.var = "b1", valid.range = c(0,500))
# assg.out <- assignment(out, seed = 123)
# b2sb <- block2seqblock(block.obj = out, assg.obj = assg.out, data = x100)
# sb <- seqblock("sbout.RData", id.vals = 1101, covar.vals = c(100, 200), file.name = "sb101.RData")
```

createBlockIDs *Create vector of integers containing block identifiers*

Description

Creates a vector of integers which represent unique blocks in an object output from `block` or `assignment`.

Usage

```
createBlockIDs(obj, data, id.var)
```

Arguments

`obj`	An output object from `block` or `assignment`.
`data`	The data frame that was input into `block` for blocking.
`id.var`	A string specifying which column of `data` contains identifying information.

Details

Under the current implementation, `level.two` in `block` should be set to `FALSE`.

If blocking was performed specifying a `groups` argument, `createBlockIDs` will assign block ID values that are unique across groups. In other words, `createBlockIDs` does not restart numbering when it encounters a new group of blocks.

Value

A numeric vector of integers with `nrow(data)` elements with lowest value equal to 1, corresponding to the block each unit is in. For units in `data` that are not in `obj`, the value of NA is assigned.

Author(s)

Ryan T. Moore

See Also

`block`, `assignment`

Examples

```
data(x100)
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"),
             block.vars = c("b1", "b2"))

createBlockIDs(out, x100, id.var = "id")

# (Block ID integers are unique, even with several groups.)
```

`diagnose` *Diagnose whether units assigned to different treatment conditions may be subject to interference or pairwise imbalance*

Description

List all pairs of units assigned to different treatment conditions whose difference on a specified variable falls within a specified range.

Usage

```
diagnose(object, data, id.vars, suspect.var, suspect.range = NULL)
```

Arguments

`object`	a dataframe or list of dataframes of assigned units, such as output from `assignment`.
`data`	a dataframe with auxiliary information on assigned units, including the specified variable `suspect.var`.
`id.vars`	a required string or vector of two strings specifying which column(s) of `data` contain identifying information.
`suspect.var`	a string specifying which column of `data` contains the variable suspected of interference or imbalance.
`suspect.range`	a vector defining the range of `suspect.var` within which units in different treatment conditions must fall to be considered suspect.

Details

`object` requires rows to correspond to blocks and columns to correspond to treatment conditions, such as output from `assignment`.

`data` should include identifying variables and variable suspected of interference or imbalance. Typically, `data` may be the same dataframe input into `block`.

An example of specified identifying variables is `id.vars = c("id", "id2")`. Unlike `block`, `diagnose` requires that the length of `id.vars` correspond to the level of the original blocking. See `block` documentation for details.

An example of specified suspect range is `suspect.var = "b2"`, `suspect.range = c(0,50)` identifies all units assigned to different treatment conditions no more than 50 units apart on variable `"b2"`.

Value

A list of dataframes, each containing a group's pairs of units assigned to different treatments falling within `suspect.range` on the variable `suspect.var`. The last column of each dataframe displays the observed difference between the two units.

Author(s)

Ryan T. Moore

See Also

assignment, block

Examples

```
data(x100)

# First, block
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"),
             block.vars = c("b1", "b2"), algorithm = "optGreedy",
             distance = "mahalanobis", level.two = FALSE, valid.var = "b1",
             valid.range = c(0,500), verbose = TRUE)

# Second, assign
assg <- assignment(out, seed = 123)

# Third, diagnose
diag <- diagnose(object = assg, data = x100, id.vars = "id",
                 suspect.var = "b2", suspect.range = c(0,50))
```

---

extract_conditions *Create vector of integers containing treatment condition identifiers*

Description

Creates a vector of integers which represent unique treatment conditions in an object output from assignment.

Usage

```
extract_conditions(assg.obj, data, id.var)
```

Arguments

assg.obj	An output object from assignment.
data	The data frame that was input into block for blocking.
id.var	A string specifying which column of data contains identifying information.

Details

Under the current implementation, level.two in block should be set to FALSE.

Value

A numeric vector of integers with `nrow(data)` elements with lowest value equal to 1, corresponding to the treatment condition column from the assignment object each unit is in. For example, if the columns of the assignment object are `Treatment` and `Control` (in that order), then `Treatment` will be represented by a `1` and `Control` will be represented by a `2`.

For units in `data` that are not in `assg.obj`, the value of NA is assigned.

Author(s)

Ryan T. Moore

See Also

`assignment`

Examples

```
data(x100)
out <- block(x100, groups = "g", n.tr = 2, id.vars = c("id"),
             block.vars = c("b1", "b2"))

assg <- assignment(out)

extract_conditions(assg, x100, id.var = "id")

# (Treatment conditions are represented by integers.)
```

`invertRIconfInt` *Calculate treatment effect confidence intervals by inverting the randomization test*

Description

Using an output object from `seqblock` or any other matrix or dataframe that includes a treatment and an outcome variable for multiple units, as well as blocking and non-blocking variables for the respective unit(s), `invertRIconfInt` calculates treatment effect confidence intervals by inverting the randomization inference test.

Usage

```
invertRIconfInt(
  dat,
  outcome.var,
  tr.var,
  tau.abs.min = -1,
  tau.abs.max = 1,
  tau.length = 10,
```

```
  n.sb.p = 100,
  id.vars,
  id.vals,
  exact.vars = NULL,
  exact.vals = NULL,
  exact.restr = NULL,
  exact.alg = "single",
  covar.vars = NULL,
  covar.vals = NULL,
  covar.restr = NULL,
  covars.ord = NULL,
  n.tr = 2,
  tr.names = NULL,
  assg.prob = NULL,
  seed = NULL,
  seed.dist,
  assg.prob.stat = NULL,
  trim = NULL,
  assg.prob.method = NULL,
  assg.prob.kfac = NULL,
  distance = "mahalanobis",
  file.name = "sbout.RData",
  query = FALSE,
  verbose = TRUE
)
```

Arguments

dat	a matrix or dataframe containing the names and values of the different blocking and non-blocking variables, as well as each unit's treatment assignment and outcome
outcome.var	a string specifying the name of the outcome variable
tr.var	a string specifying the name of the treatment variable
tau.abs.min	lower bound of the range across which the confidence intervals will be computed
tau.abs.max	upper bound of the range across which the confidence intervals will be computed
tau.length	the number of (evenly spaced) possible treatment effects across the range specified by tau.abs.min and tau.abs.max for which location inside or outside the confidence intervals will be computed
n.sb.p	the number of times that sequential blocking will be performed on the dataset
id.vars	see the seqblock documentation
id.vals	see the seqblock documentation
exact.vars	see the seqblock documentation
exact.vals	see the seqblock documentation
exact.restr	see the seqblock documentation
exact.alg	see the seqblock documentation

covar.vars	see the seqblock documentation
covar.vals	see the seqblock documentation
covar.restr	see the seqblock documentation
covars.ord	see the seqblock documentation
n.tr	see the seqblock documentation
tr.names	see the seqblock documentation
assg.prob	see the seqblock documentation
seed	see the seqblock documentation
seed.dist	see the seqblock documentation
assg.prob.stat	see the seqblock documentation
trim	see the seqblock documentation
assg.prob.method	see the seqblock documentation
assg.prob.kfac	see the seqblock documentation
distance	see the seqblock documentation
file.name	see the seqblock documentation
query	see the seqblock documentation
verbose	see the seqblock documentation

Details

invertRIconfInt takes a data matrix (or data frame) containing names and values of different blocking and non-blocking variables, as well as each unit's treatment assignment and outcome as input and returns a list of treatment effect confidence intervals.

Apart from specifying the treatment and outcome variable, the user can set all other arguments to seqblock when running invertRIconfInt. The function will then calculate the confidence intervals by employing a method described in Ho and Imai (2006), which inverts Fisher's exact test. The resulting confidence intervals are distribution-free, nonparametric and have accurate coverage probabilities.

Value

A list with elements

- **ci95**: vector of treatment effects within the 95% confidence interval
- **ci90**: vector of treatment effects within the 90% confidence interval
- **ci80**: vector of treatment effects within the 80% confidence interval

Author(s)

Ryan T. Moore <rtm@american.edu> and Jonathan Homola <homola@wustl.edu>

References

Moore, Ryan T. and Sally A. Moore. 2013. "Blocking for Sequential Political Experiments." *Political Analysis* 21(4): 507-523.

Ho, Daniel E., and Kosuke Imai. 2006. "Randomization inference with natural experiments: An analysis of ballot effects in the 2003 California recall election." *Journal of the American Statistical Association* 101(475): 888-900.

See Also

seqblock

Examples

```
# Create an example data matrix with 50 observations that contains an ID variable,
# a dummy variable indicating gender, an age variable (between 18 and 55), a
# treatment variable and an outcome variable (between 15 and 20).
# id <- seq(1, 50, 1)
# gender <- sample(c(1, 2), 50, replace = TRUE)
# age <- sample(seq(18, 55, 1), 50, replace = TRUE)
# treat <- sample(c(1, 2), 50, replace = TRUE)
# out <- treat + sample(seq(15, 20, 1), 50, replace = TRUE)
# df <- cbind(id, gender, age, out, treat)

# Check summary statistics for the created data
# aggregate(out ~ treat, df, mean)

# Run invertRIconfInt()
# invertRIconfInt(data, outcome.var="out", tr.var="treat", tau.abs.min = -3,
#                 tau.abs.max = 3, id.vars = "id", id.vals = "id",
#                 exact.vars = c("gender", "age"), exact.vals = c("gender", "age"))
```

---

outCSV *Export blocked or assigned data to .csv format files*

Description

Exports output from `block` or `assignment` to a set of .csv files using `write.csv`.

Usage

```
outCSV(block.obj, namesCol = NULL, file.names = NULL, digits = 2, ...)
```

Arguments

`block.obj`	A list of dataframes, such as output from `block` or `assignment`.
`namesCol`	An optional character vector of column names to be used in output files.
`file.names`	An optional character vector of file names specifying the output file names.

digits	An integer representing the number of decimal places to which to round multivariate distances in output files, passed to round().
...	Additional arguments passed to write.csv.

Details

Under the default (file.names = NULL), each file is named "GroupXXX.csv", where "XXX" is the group name taken from the input object.

Value

A set of .csv files, one for each element of the input list of blocked or assigned units, written by write.csv().

Author(s)

Ryan T. Moore

See Also

outTeX, write.csv, block, assignment

Examples

```
data(x100)

# Block and assign:
out <- block(x100, groups = "g", n.tr = 2, id.vars = "id", block.vars = c("b1", "b2"))
assg <- assignment(out, seed = 123)

# create three .csv files of blocks
## Not run: outCSV(out)
# create three .csv files of assigned blocks
#   (note: overwrites blocked .csv files)
## Not run: outCSV(assg)
# create three .csv files with custom file names
## Not run: outCSV(assg, file.names = c("file1", "file2", "file3"))
```

---

outTeX *Export blocked or assigned data to .tex format files*

Description

Exports output from block or assignment to a set of .tex files using print(xtable).

Usage

```
outTeX(
  block.obj,
  namesCol = NULL,
  file.names = NULL,
  captions = NULL,
  digits = 2,
  ...
)
```

Arguments

block.obj	A list of dataframes, such as output from block or assignment.
namesCol	An optional character vector of column names to be used in output files.
file.names	An optional character vector of file names specifying the output file names.
captions	An optional character vector of file names specifying the table captions. See Details below.
digits	An integer representing the number of decimal places to which to round multivariate distances in output files, passed to round.
...	Additional arguments passed to xtable.

Details

Under the default (file.names = NULL), each file is named "GroupXXX.tex", where "XXX" is the group name taken from the input object. Under the default (captions = NULL), each caption is "Group XXX.", where "XXX" is the group name taken from the input object.

outTeX appends .tex to the user-specified file.names.

The table reference labels are created as t:XXX, where XXX is the file name (without .tex) for the .tex file containing that table.

captions takes a list of strings of length equal to the number of groups in block.obj$blocks, if block.obj is output from block, or the number of groups in block.obj$assg, if block.obj is output from assignment.

The tables in the output .tex files can be integrated into an existing .tex document using LaTeX code '\include{GroupXXX}'.

Value

A set of .tex files, one for each element of the input list of blocked or assigned units, written by print(xtable).

Author(s)

Ryan T. Moore

See Also

outCSV, xtable, block, assignment

Examples

```
data(x100)

# Block and assign:
out <- block(x100, groups = "g", n.tr = 2, id.vars = "id", block.vars = c("b1", "b2"))
assg <- assignment(out, seed = 123)

# create three .tex files of blocks
## Not run: outTeX(out)
# create three .tex files of assigned blocks
#   (note: overwrites blocked .tex files)
## Not run: outTeX(assg)
# create three .tex files with custom file names and captions
## Not run: outTeX(assg, file.names = c("f1", "f2", "f3"), captions = c("C 1.", "C 2.", "C 3."))
```

`seqblock` *Sequential assignment of unit(s) into experimental conditions using covariates*

Description

Sequentially assign units into experimental conditions. Blocking begins by creating a measure of multivariate distance between a *current* unit and one or multiple *prior*, already-assigned unit(s). Then, average distance between current unit and each treatment condition is calculated, and random assignment is biased toward conditions more dissimilar to current unit. Argument values can be specified either as argument to the function, or via a query. The query directly asks the user to identify the blocking variables and to input, one-by-one, each unit's variable values.

Usage

```
seqblock(
  object = NULL,
  id.vars,
  id.vals,
  exact.vars = NULL,
  exact.vals = NULL,
  exact.restr = NULL,
  exact.alg = "single",
  covar.vars = NULL,
  covar.vals = NULL,
  covar.restr = NULL,
  covars.ord = NULL,
  n.tr = 2,
  tr.names = NULL,
  assg.prob = NULL,
  seed = NULL,
  seed.dist,
```

```
  assg.prob.stat = NULL,
  trim = NULL,
  assg.prob.method = NULL,
  assg.prob.kfac = NULL,
  distance = NULL,
  file.name = NULL,
  query = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	a character string giving the file name of a `.RData` file containing a list output from the `seqblock` function which contains at least one previously assigned unit.
`id.vars`	a string or vector of strings specifying the name of the identifying variable(s); if `query = FALSE` and the object argument is not given, then the `id.vars` argument is required.
`id.vals`	a vector of ID values for every unit being assigned to a treatment group; those are corresponding to the `id.vars`.
`exact.vars`	a string or vector of strings containing the names of each of the exact blocking variables.
`exact.vals`	a vector containing the unit's values on each of the exact blocking variables.
`exact.restr`	a list object containing the restricted values that the exact blocking variables can take on. Thus the first element of `exact.restr` is a vector containing all of the possible values that the first exact blocking variable (see `exact.vars` above) can take on; the second element is a vector containing all of the possible values for the second exact blocking variable; and so on.
`exact.alg`	a string specifying the blocking algorithm. Currently the only acceptable value is `"single"`. This algorithm creates a variable with a unique level for every possible combination of the values in all of the exact variables. See Details section below.
`covar.vars`	a string or vector of strings containing the names of each of the non-exact blocking variables.
`covar.vals`	a vector containing the unit's values on each of the non-exact blocking variables.
`covar.restr`	a list object containing the restricted values that the non-exact blocking variables can take on. Thus the first element of `covar.restr` is a vector containing all of the possible values that the first non-exact blocking variable (see `covar.vars` above) can take on; the second element is a vector containing all of the possible values for the second non-exact blocking variable; and so on.
`covars.ord`	a string or vector of strings containing the name of the non-exact blocking variables ordered so that the highest priority covariate comes first, followed by the second highest priority covariate, then the third, etc.
`n.tr`	the number of treatment groups. If not specified, this defaults to `n.tr = 2`.

tr.names	a string or vector of strings containing the names of the different treatment groups.
assg.prob	a numeric vector containing the probabilities that a unit will be assigned to the treatment groups; this vector should sum to 1.
seed	an optional integer value for the random seed, which is used when assigning units to treatment groups.
seed.dist	an optional integer value for the random seed set in cov.rob, used to calculate measures of the variance-covariance matrix robust to outliers.
assg.prob.stat	a string specifying which assignment probability summary statistic to use; valid values are mean, median, and trimmean. If not specified, this defaults to assg.prob.stat = "mean".
trim	a numeric value specifying what proportion of the observations are to be dropped from each tail when the assignment probability summary statistic (assg.prob.stat) is set equal to trimmean. Blocks on each tail of the distribution are dropped before the mean is calculated. If not specified, this defaults to trim = 0.1.
assg.prob.method	a string specifying which algorithm should be used when assigning treatment probabilities. Acceptable values are ktimes, fixed, prop, prop2, and wprop. If not specified, this defaults to assg.prob.method = "ktimes".
assg.prob.kfac	a numeric value for k, the factor by which the most likely experimental condition will be multiplied relative to the other conditions. If not specified, this defaults to assg.prob.kfac = 2.
distance	a string specifying how the multivariate distance used for blocking covariates are calculated. If not specified, this defaults to distance = "mahalanobis".
file.name	a string containing the name of the file that one would like the output to be written to. Ideally this file name should have the extension .RData.
query	a logical stating whether the console should ask the user questions to input the data and assign a treatment condition. If not specified, this defaults to query = FALSE.
verbose	a logical stating whether the function should print the name of the output file, the current working directory, the treatment group that the most recent unit was assigned to, and the dataframe x returned by the function as part of the bdata list. If not specified, this defaults to verbose = TRUE.
...	additional arguments.

Details

The seqblock function's code is primarily divided into two parts: the first half deals with instances, in which the unit being assigned is the first unit in a given study to receive an assignment; the second half addresses subsequent units that are assigned after at least one first assignment has already been made. If the object argument is left as NULL, the function will run the first half; if the object argument is specified, the second part will be executed. When object = NULL, the researcher has no past file from which to pull variable names and past data; this corresponds to the case when the unit being assigned is the first one. If the researcher does specify object, it implies the user is drawing data from a past file, which means this is not the first unit in the study to be assigned to a treatment.

However, the function can be called for subsequent units even when `object` is not specified. By setting `query = TRUE`, the console will ask the researcher whether this is the first unit to be assigned in the study. Based on the researcher's response, it will decide which part of the code to run.

If the `object` and `file.name` arguments are set to the same value, then `seqblock` overwrites the specified file with a new file, which now contains both the previously-assigned units and the newly-assigned unit. To create a new file when a new unit is assigned, use a new `file.name`.

The `single` algorithm (see `exact.alg` in the Arguments section above) creates a variable that has a unique level for every possible combination of the exact variables. As an example, say that there were 3 exact blocking variables: *party* (Democrat, Republican); *region* (North, South); and *education* (HS, NHS). The `single` algorithm creates one level for units with the following values: Democrat-North-HS. It would create another level for Democrat-North-NHS; a third level for Republican-North-HS; and so forth, until every possible combination of these 3 variables has its own level. Thus if there are k exact blocking variables and each exact blocking variable has q_i values it can take on, then there are a total of $\prod_1^k q_i$ levels created.

The `distance = "mcd"` and `distance = "mve"` options call `cov.rob` to calculate measures of multivariate spread robust to outliers. The `distance = "mcd"` option calculates the Minimum Covariance Determinant estimate (Rousseeuw 1985); the `distance = "mve"` option calculates the Minimum Volume Ellipsoid estimate (Rousseeuw and van Zomeren 1990). When `distance = "mcd"`, the interquartile range on blocking variables should not be zero. The `distance = "euclidean"` option calculates the Euclidean distance between the new unit and the previously-assigned units. The default `distance = "mahalanobis"` option calculates the Mahalanobis distance.

Value

A list (called `bdata`) with elements

- **x**: a dataframe containing the names and values for the different ID and blocking variables, as well as each unit's initial treatment assignment.
- **nid**: a string or vector of strings containing the name(s) of the ID variable(s).
- **nex**: a string or vector of strings containing the name(s) of the exact blocking variable(s).
- **ncv**: a string or vector of strings containing the name(s) of the non-exact blocking variable(s).
- **rex**: a list of the restricted values of the exact blocking variables.
- **rcv**: a list of the restricted values of the non-exact blocking variables.
- **ocv**: a vector of the order of the non-exact blocking variables.
- **trn**: a string or vector of strings containing the name(s) of the different treatment groups.
- **apstat**: a string specifying the assignment probability summary statistic that was used.
- **mtrim**: a numeric value specifying the proportion of observations to be dropped when the assignment probability statistic takes on the value `"trimmean"`.
- **apmeth**: a string specifying the assignment probability algorithm that was used.
- **kfac**: the assignment probability *kfactor*; see *assg.prob.kfac* in the Arguments section above.
- **assgpr**: a vector of assignment probabilities to each treatment group.
- **distance**: a string specifying how the multivariate distance used for blocking was calculated.
- **trd**: a list with the length equal to the number of previously assigned treatment conditions; each object in the list contains a vector of the distance between each unit in one treatment group and the new unit. This will be `NULL` when there are no non-exact blocking variables.

- **tr.sort**: a string vector of treatment conditions, sorted from the largest to the smallest. Set to NULL when there are no non-exact blocking variables.
- **p**: a vector of assignment probabilities to each treatment group used in assigning a treatment condition to the new unit.
- **distance**: a string specifying how the multivariate distance used for blocking is calculated
- **trcount**: a table containing the counts for each experimental/treatment conditions.
- **datetime**: the date and time at which each unit was assigned their treatment group.
- **orig**: a dataframe containing the names and values for the different id and blocking variables, as well as each unit's treatment assignment.

Author(s)

Ryan T. Moore <rtm@american.edu>, Tommy Carroll <tcarroll22@wustl.edu>, Jonathan Homola <homola@wustl.edu> and Jeong Hyun Kim <jeonghyun.kim@wustl.edu>

References

Moore, Ryan T. and Sally A. Moore. 2013. "Blocking for Sequential Political Experiments." *Political Analysis* 21(4):507-523.

Moore, Ryan T. 2012. "Multivariate Continuous Blocking to Improve Political Science Experiments." *Political Analysis* 20(4):460-479.

Rousseeuw, Peter J. 1985. "Multivariate Estimation with High Breakdown Point". *Mathematical Statistics and Applications* 8:283-297.

Rousseeuw, Peter J. and Bert C. van Zomeren. 1990. "Unmasking Multivariate Outliers and Leverage Points". *Journal of the American Statistical Association* 85(411):633-639.

See Also

assignment, block

Examples

```
# Assign first unit (assume a 25 year old member of the Republican Party)
# to a treatment group. Save the results in file "sdata.RData":
# seqblock(query = FALSE, id.vars = "ID", id.vals = 001, exact.vars = "party",
# exact.vals = "Republican", covar.vars = "age", covar.vals = 25, file.name = "sdata.RData")

# Assign next unit (age 30, Democratic Party):
# seqblock(query = FALSE, object = "sdata.RData", id.vals = 002, exact.vals = "Democrat",
# covar.vars = "age", covar.vals = 30, file.name = "sdata.RData")
```

x100 *Simulated data for demonstrating blockTools functionality*

Description

Simulated data for demonstrating `blockTools` functionality.

Usage

```
data(x100)
```

Format

A dataframe with 100 rows and 6 columns.

id Identifies units.

id2 Identifies subunits.

b1 Numeric blocking variable 1.

b2 Numeric blocking variable 2.

g Character variable to identify which of three groups each unit is in ("a", "b", or "c").

ig Numeric variable that can be ignored.

Author(s)

Ryan T. Moore

Examples

```
data(x100)
head(x100)
```

Index