

# Package ‘blockTools’

October 29, 2009

**Type** Package

**Title** Block, assign, and diagnose potential interference in randomized experiments

**Version** 0.4-1

**Date** 2009-10-28

**Author** Ryan T. Moore

**Maintainer** Ryan T. Moore <rtm@wustl.edu>

**Depends** MASS, gtools, nbpMatching, xtable

**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.

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-10-29 06:51:56

## R topics documented:

blockTools-package . . . . .	2
assignment . . . . .	3
block . . . . .	4
diagnose . . . . .	8
outCSV . . . . .	9
outTeX . . . . .	10
x100 . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

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

Package: blockTools  
Type: Package  
Version: 0.4-1  
Date: 2009-10-28  
License: GPL (>=2)

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.

## Author(s)

Ryan T. Moore

Maintainer: Ryan T. Moore <rtm@wustl.edu>

## References

<http://rtm.wustl.edu/software.blockTools.htm>

## 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)
```

---

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

<code>block.obj</code>	an output object from <code>block</code> , or a user-specified block object.
<code>seed</code>	a user-specified random seed.
<code>namesCol</code>	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.

## Value

A list with elements

<code>assg</code>	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.
<code>call</code>	the original call to <code>assignment</code> .

**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 = 10^8, row.sort = NULL, level.two = FALSE, valid.var =
      valid.range = NULL, seed, verbose = FALSE, ...)
```

**Arguments**

<code>data</code>	a dataframe or matrix, with units in rows and variables in columns.
<code>vcov.data</code>	an optional matrix of data used to estimate the variance-covariance matrix for calculating multivariate distance.
<code>groups</code>	an optional column name from <code>data</code> , specifying subgroups within which blocking occurs.
<code>n.tr</code>	the number of treatment conditions per block.
<code>id.vars</code>	a required string or vector of two strings specifying which column(s) of <code>data</code> contain identifying information.

<code>block.vars</code>	an optional string or vector of strings specifying which column(s) of data contain the blocking variables.
<code>algorithm</code>	a string specifying the blocking algorithm. "optGreedy", "optimal", "naiveGreedy", "randGreedy", and "sortGreedy" algorithms are currently available. See Details for more information.
<code>distance</code>	either a) a string defining how the multivariate distance used for blocking is calculated (options include "mahalanobis", "mcd", and "mve"), or b) a user-defined $k$ -by- $k$ matrix, where $k$ is the number of rows in data.
<code>weight</code>	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.
<code>optfactor</code>	a number by which distances are multiplied then divided when <code>algorithm = "optimal"</code> .
<code>row.sort</code>	an optional vector of integers from 1 to <code>nrow(data)</code> used to sort the rows of data when <code>algorithm = "sortGreedy"</code> .
<code>level.two</code>	a logical defining the level of blocking.
<code>valid.var</code>	an optional string defining a variable on which units in the same block must fall within the range defined by <code>valid.range</code> .
<code>valid.range</code>	an optional vector defining the range of <code>valid.var</code> within which units in the same block must fall.
<code>seed</code>	an optional integer value for the random seed set in <code>cov.rob</code> , used to calculate measures of the variance-covariance matrix robust to outliers.
<code>verbose</code>	a logical specifying whether <code>groups</code> names and block numbers are printed as blocks are created.
<code>...</code>	additional arguments passed to <code>cov.rob</code> .

## 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, .... Other algorithms proceed similarly.

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.

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)`.

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; the `distance = "mve"` option calculates the Minimum Volume Ellipsoid estimate.

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"`.

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".

**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

**References**

King, Gary, Emmanuela Gakidou, Nirmala Ravishankar, Ryan T. Moore, Jason Lakin, Manett Vargas, Martha María Téllez-Rojo and Juan Eugenio Hernández Ávila and Mauricio Hernández Ávila and Héctor Hernández 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.

**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(even(i)) {x100$id[i] <- x100$id[i-1]}}
```

```
out <- 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), verbose = TRUE)
```

---

diagnose	<i>Diagnose whether units assigned to different treatment conditions may be subject to interference or pairwise imbalance</i>
----------	---

---

### 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 <code>assignment</code> .
data	a dataframe with auxiliary information on assigned units, including the specified variable <code>suspect.var</code> .
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 <code>suspect.var</code> 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))
```

---

`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, digits = 2, ...)
```

**Arguments**

<code>block.obj</code>	a list of dataframes, such as output from <code>block</code> or <code>assignment</code> .
<code>namesCol</code>	an optional vector of column names to be used in output files.
<code>digits</code>	number of digits to which to round multivariate distances in output files.
<code>...</code>	additional arguments passed to <code>write.csv</code> .

**Value**

A set of `.csv` files, one for each element of the input list of blocked or assigned units. Each file is named "GroupXXX.csv", where "XXX" is the group name taken from the input object.

**Author(s)**

Ryan T. Moore

**See Also**

[outTeX](#), [write.csv](#), [block](#), [assignment](#)

**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)

## create three .csv files of blocks
outCSV(out)
## create three .csv files of assigned blocks
## (note: overwrites blocked .csv files)
outCSV(assg)
```

---

outTeX

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

---

**Description**

Exports output from `block` or `assignment` to a set of `.tex` files using `xtable`.

**Usage**

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

**Arguments**

<code>block.obj</code>	a list of dataframes, such as output from <code>block</code> or <code>assignment</code> .
<code>namesCol</code>	an optional vector of column names to be used in output files.
<code>digits</code>	the number of digits to which to round multivariate distances in output files.
<code>...</code>	additional arguments passed to <code>xtable</code> .

**Value**

A set of `.tex` files, one for each element of the input list of blocked or assigned units. Each file is named “GroupXXX.tex”, where “XXX” is the group name taken from the input object. The tables in these `.tex` files can easily be integrated into an existing `.tex` document using LaTeX code ‘`\includeGroupXXX`’.

**Author(s)**

Ryan T. Moore

**See Also**[outCSV](#), [block](#), [assignment](#)**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)

## create three .tex files of blocks
outTeX(out)
## create three .tex files of assigned blocks
## (note: overwrites blocked .tex files)
outTeX(assg)
```

---

`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. Columns `id` and `id2` are unit and subunit identifying variables, `b1` and `b2` are blocking variables, `g` identifies which of three groups each unit is in ("a", "b", or "c"), and `ig` is an ignored variable.

**Author(s)**

Ryan T. Moore

**Examples**`data(x100)`

# Index

## \*Topic **IO**

outCSV, 9

outTeX, 10

## \*Topic **datasets**

x100, 11

## \*Topic **design**

assignment, 3

block, 4

diagnose, 8

outCSV, 9

outTeX, 10

## \*Topic **multivariate**

block, 4

## \*Topic **package**

blockTools-package, 2

assignment, 3, 7, 9–11

block, 4, 4, 9–11

blockTools (*blockTools-package*), 2

blockTools-package, 2

diagnose, 4, 7, 8

outCSV, 9, 11

outTeX, 10, 10

write.csv, 10

x100, 6, 11