Package ‘hassediagrams’

June 11, 2025

Type Package

Title Hasse Diagram of the Layout Structure and Restricted Layout Structure

Version 1.0

Date 2025-06-01

Maintainer Damianos Michaelides <dm3g15@soton.ac.uk>

Description Returns a Hasse diagram of the layout structure (Bate and Chatfield (2016)) <doi:10.1080/00224065.2016.11918173> or the restricted layout structure (Bate and Chatfield (2016)) <doi:10.1080/00224065.2016.11918174> of an experimental design.

License GPL-2

Encoding UTF-8

URL https://github.com/GSK-Biostatistics/hassediagrams

BugReports https://github.com/GSK-Biostatistics/hassediagrams/issues

Imports igraph, methods, MASS, grDevices, graphics, stats, utils

Depends R (>= 3.5.0)

Suggests dae, knitr, rmarkdown, jsonlite, kableExtra

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Damianos Michaelides [aut, cre],
Simon Bate [aut],
Marion Chatfield [aut]

Repository CRAN

Date/Publication 2025-06-11 13:00:05 UTC

Contents

analytical 2
concrete 3
dental 4
hasselayout 5
hasserls 9
human 16
itemlist 17
print.rls 19

Index **21**

`analytical` *A cross-nested design for an analytical method investigation*

Description

The reliability of an analytical method was assessed in an experiment consisting of three batches of material, analysed by four analysts, two at each site. Within each site, there were two chromatographic systems and two columns. For each batch/analyst/system/column combination, two preparations (dissolved samples) were made. From each preparation, two injections were performed.

Usage

```
data(analytical)
```

Format

A data frame of 192 observations on 8 variables/factors:

Site Categoric factor with levels 1 and 2.

Analyst Categoric factor with levels 1-4.

Run Categoric factor with levels 1-16.

Prep Categoric factor with levels 1-96.

Injection Categoric factor with levels 1-192.

System Categoric factor with levels 1-4.

Column Categoric factor with levels 1-4.

Batch Categoric factor with levels 1-3.

Source

Bate, S.T. and Chatfield, M.J. (2016). "Identifying the Structure of the Experimental Design". *Journal of Quality Technology* 48, pp. 343-364.

Examples

```
data("analytical")
analytical
```

---

concrete — *A fractional factorial design for investigating asphalt concrete production*

Description

This is fractional factorial design given in Anderson and McLean (1974) p.256 from an experiment to investigate the effect of controllable variables/factors on the quality of asphalt concrete production.

Usage

```
data(concrete)
```

Format

A half fraction factorial design of 16 runs on 6 variables/factors. The 6 variables/factors, each at two levels, included in the design are:

Aggregate gradation (AG) Categoric factor with levels being fine and coarse.

Compaction temperature (CoT) Numeric factor with low 250 and high 300.

Asphalt content (AC) Numeric factor with low 5 and high 7.

Curing condition (CC) Categoric factor with levels wrapped and unwrapped.

Curing temperature (CuT) Numeric factor with low 45 and high 72.

Run Categoric factor with levels 1-16.

Source

Anderson, V.L. and McLean, R.A. (1974). *Design of Experiments.* Marcel Dekker Inc.: New York.

Examples

```
data("concrete")
concrete
```

dental *A crossover design for a dental study*

Description

This is a crossover design (Study H) given in Newcombe et al. (1995) to study the effects of CHX rinses on 4 day plaque regrowth. The study consisted of 24 patients assessed over 3 treatment periods. The purpose of the study was to compare 2 CHX rinses with saline. The design is based on pairs of Latin squares balanced for carry over.

Usage

```
data(dental)
```

Format

A crossover design of 72 runs on 5 variables/factors. The 5 variables/factors included in the design are:

Sequence Categoric factor with levels 1-6.

Subject Categoric factor with levels 1-32.

Period Categoric factor with levels 1-3.

Treatment Categoric factor with levels CHX1, CHX2 and Saline.

Observation Categoric factor with levels 1-72.

Source

Newcombe, R.G., Addy, M. and McKeown, S. (1995). "Residual effect of chlorhexidine gluconate in 4-day plaque regrowth crossover trials, and its implications for study design". *Journal of Periodontal Research*, 30, 5, pp. 319-324.

Examples

```
data("dental")
dental
```

hasselayout *Hasse diagram of the layout structure*

Description

Returns a Hasse diagram of the layout structure of an experimental design

Usage

```
hasselayout(
  datadesign,
  randomfacsid = NULL,
  showLS = "Y",
  showpartialLS = "Y",
  showdfLS = "Y",
  check.confound.df = "Y",
  maxlevels.df = "Y",
  table.out = "N",
  pdf = "N",
  example = "example",
  outdir = NULL,
  hasse.font = "sans",
  produceBWPlot = "N",
  structural.colour = "grey",
  structural.width = 2,
  partial.colour = "orange",
  partial.width = 1.5,
  objects.colour = "mediumblue",
  df.colour = "red",
  larger.fontlabelmultiplier = 1,
  middle.fontlabelmultiplier = 1,
  smaller.fontlabelmultiplier = 1
)
```

Arguments

`datadesign` A data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables/factors in the experimental design. The data frame should only include the variables/factors/columns that the user wants to include in the Hasse diagram.

`randomfacsid` An optional vector specifying whether the factors are defined as fixed (entry = 0) or random (entry = 1). The default choice is NULL and the function automatically sets all entries to 0. The length of the vector should be equal to the number of variables/factors in the design, i.e., the length of the vector should be equal to the number of columns of the argument `datadesign`.

`showLS` logical. If "N" then generation of the Hasse diagram is suppressed. The default is "Y".

`showpartialLS`	logical. If "N" then the partial crossing between structural objects (using dotted connecting lines) is not illustrated on the Hasse diagram of the layout structure. The default is "Y".
`showdfLS`	logical. If "N" then the structural object label is not displayed on the Hasse diagram of the layout structure. The default is "Y".
`check.confound.df`	logical. If "N" then the check for confounded degrees of freedom is not performed. The default is "Y".
`maxlevels.df`	logical. If "N" then the potential maximum number of levels of a generalised factor is removed from the structural object label on the Hasse diagram of the layout structure. The default is "Y".
`table.out`	logical. If "Y" then a table that shows the relationships between the structural objects in the layout structure is printed. The default is "N".
`pdf`	logical. If "Y" then a pdf file containing the Hasse diagram of the layout structure is generated. The default is "N", i.e., a pdf file is not generated.
`example`	File name for the pdf output file containing the Hasse diagram. The default is set to "example".
`outdir`	Location of the pdf output file if `pdf="Y"`. The default is set to `NULL` and in this case the pdf output file containing the Hasse diagram output is stored to a temporary file. To specify a permanent location this argument needs be specified.
`hasse.font`	The name of the font family used for all text included in the Hasse diagram. Standard and safe font families to choose are "sans", "serif", and "mono". If the design's factor labels contain Unicode characters, or for consistency with Hasse diagrams of restricted layout structures using hasserls, a Unicode friendly font family should be selected. For more details on Unicode friendly family options see the Details section in the hasserls documentation. The default is "sans".
`produceBWPlot`	logical. If "Y" then the Hasse diagram will be generated in black and white format. The default is set to "N", i.e., a coloured version of the plot is produced.
`structural.colour`	The colour of the structural lines that connect structural objects on the Hasse diagram. The default colour is "grey".
`structural.width`	The width of the structural lines on the Hasse diagram. The default width is 2.
`partial.colour`	The colour of the partial crossing dotted lines of the connecting objects on the Hasse diagram. The default colour is "orange".
`partial.width`	The width of the partial crossing dotted lines on the Hasse diagram. The default width is 1.5.
`objects.colour`	The colour of the labels of the structural objects on the Hasse diagram. The default colour is "mediumblue".
`df.colour`	The colour of the degrees of the freedom labels on the Hasse diagram. The default colour is "red".
`larger.fontlabelmultiplier`	The large font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram where there are four or less objects at that level in the diagram. The default is 1.

`middle.fontlabelmultiplier`
: The medium font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram involving a factor that is equivalent to a generalised factor. The default is 1.

`smaller.fontlabelmultiplier`
: The small font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram where there are five or more objects at that level of the diagram. The default is 1.

Details

The hasselayout function generates the Hasse diagram of the layout structure of the experimental design, as described in Bate and Chatfield (2016a). The diagram consists of a set of structural objects, corresponding to the factors and generalised factors, and the relationships between the structural objects (either crossed, nested, partially crossed or equivalent), as defined by the structure of the experimental design.

The function requires a dataframe containing only the variables corresponding to the experimental factors that define the experimental design (i.e., no response variables should be included).

In the dataframe the levels of the variables/factors must be uniquely identified and have a physical meaning, otherwise the function will not correctly identify the nesting/crossing of the variables/factors. For example, consider an experiment consisting of Factor A (with k levels) that nests Factor B (with q levels per level of Factor A). The levels of Factor B should be labelled 1 to k x q and not 1 to q (repeated k times).

Where present, two partially crossed factors are illustrated on the diagram with a dotted line connecting them. This feature can be excluded using the `showpartialLS` option.

The maximum number of possible levels of each generalised factor, along with the actual number present in the design and the "skeleton ANOVA" degrees of freedom, can be included in the structural object label on the Hasse diagram.

Using the randomfacsid argument the factors that correspond to random effects can be highlighted by underlining them on the Hasse diagram. The vector should be equal to the number of variables/factors in the design and consist of fixed (entry = 0) or random (entry = 1) values.

The `hasselayout` function evaluates the design in order to identify if there are any confounded degrees of freedom across the design. It is not recommended to perform this evaluation for large designs due to the potential high computational cost. This can be controlled using the check.confound.df = "N" option.

Value

The function `hasselayout` returns:

1. The Hasse diagram of the layout structure (if `showLS="Y"`).

2. The layout structure table shows the relationships between the structural objects in the layout structure (if `table.out="Y"`). The individual entries in the table consist of blanks on the main diagonal and 0's, (0)'s or 1's elsewhere. If the factor (or generalised factor) corresponding to the ith row and the factor (or generalised factor) corresponding to the jth column are fully crossed, then a 0 is entered in the (i,j)th entry in the table. If these factors (or generalised factors) are partially crossed, or the ith row factor (or generalised factor) only has one level and nests the jth column factor (or generalised factor), then the (i,j)th entry is (0). If the ith row factor (or generalised factor)

is nested within the jth column factor (or generalised factor), then a 1 is entered in the (i,j)th entry. If two factors (or generalised factors) are equivalent, then they share a single row and column in the table, where the row and column headers include both factor (or generalised factor) names, separated by an "=" sign.

3. If there are confounded degrees of freedom, a table of the structural objects and a description of the associated degrees of freedom is printed.

Author(s)

Damianos Michaelides, Simon Bate, and Marion Chatfield

References

Bate, S.T. and Chatfield, M.J. (2016a), Identifying the structure of the experimental design. Journal of Quality Technology, 48, 343-364.

Bate, S.T. and Chatfield, M.J. (2016b), Using the structure of the experimental design and the randomization to construct a mixed model. Journal of Quality Technology, 48, 365-387.

Box, G.E.P., Hunter, J.S., and Hunter, W.G., (1978), Statistics for Experimenters. Wiley.

Joshi, D.D. (1987), Linear Estimation and Design of Experiments. Wiley Eastern, New Delhi.

Williams, E.R., Matheson, A.C. and Harwood, C.E. (2002), Experimental design and analysis for tree improvement. 2nd edition. CSIRO, Melbourne, Australia.

Examples

```
## Examples using the package build-in data concrete, dental, human, analytical.

## A fractional factorial design for investigating asphalt concrete production
hasselayout(datadesign=concrete, larger.fontlabelmultiplier=1.6,
            smaller.fontlabelmultiplier=1.3, table.out="Y")

## A crossover design for a dental study
hasselayout(datadesign=dental, randomfacsid = c(0,1,0,0,0),
            larger.fontlabelmultiplier = 1.6)

## A block design for an experiment assessing human-computer interaction
hasselayout(datadesign=human, randomfacsid = c(1,1,0,0,0,0,1),
            larger.fontlabelmultiplier=1.4)

## A cross-nested design for an analytical method investigation
hasselayout(datadesign=analytical, randomfacsid = c(0,0,1,1,1,0,0,0),
            showpartialLS="N", check.confound.df="N",
            larger.fontlabelmultiplier=1,
            smaller.fontlabelmultiplier=1.6)


## Examples using data from the dae package (conditionally loaded)

if (requireNamespace("dae", quietly = TRUE)) {

  ## Data for a balanced incomplete block experiment, Joshi (1987)
```

```
 data(BIBDWheat.dat, package = "dae")
 # remove the response from the dataset
 BIBDWheat <- BIBDWheat.dat[, -4]
 hasselayout(datadesign=BIBDWheat, example = "BIBDWheat")


 ## Data for an un-replicated 2^4 factorial experiment to investigate a chemical process
 ## from Table 10.6 of Box, Hunter and Hunter (1978)

 data(Fac4Proc.dat, package = "dae")
 # remove the response from the dataset
 Fac4Proc <- Fac4Proc.dat[, -6]
 hasselayout(datadesign=Fac4Proc, example = "Fac4Proc", showpartialLS="N",
             smaller.fontlabelmultiplier=2)


## Data for an experiment with rows and columns from p.144 of
## Williams, Matheson and Harwood (2002)

data(Casuarina.dat, package = "dae")
# remove the response from the dataset
Casuarina <- Casuarina.dat[, -7]
# create unique factor level labels
Casuarina$Row <- paste(Casuarina$Reps, Casuarina$Rows)
Casuarina$Col <- paste(Casuarina$Reps, Casuarina$Columns)
Casuarina <- Casuarina[, -c(2,3)]
hasselayout(datadesign=Casuarina, randomfacsid=c(1,0,1,0,0,0),
            example="Casuarina", check.confound.df="N",
            showpartialLS="N")

} else {
  message("Examples using data from the 'dae' package
          require 'dae' to be installed.")
}
```

`hasserls` *Hasse diagram of the restricted layout structure*

Description

Returns a Hasse diagram of the restricted layout structure of an experimental design

Usage

```
hasserls(
  object,
```

```
  randomisation.objects,
  random.arrows = NULL,
  showRLS = "Y",
  showpartialRLS = "Y",
  showdfRLS = "Y",
  showrandRLS = "Y",
  check.confound.df = "Y",
  maxlevels.df = "Y",
  table.out = "N",
  equation.out = "N",
  pdf = "N",
  example = "example",
  outdir = NULL,
  hasse.font = "sans",
  produceBWPlot = "N",
  structural.colour = "grey",
  structural.width = 2,
  partial.colour = "orange",
  partial.width = 1.5,
  objects.colour = "mediumblue",
  df.colour = "red",
  arrow.colour = "mediumblue",
  arrow.width = 1.5,
  arrow.pos = 7.5,
  larger.fontlabelmultiplier = 1,
  middle.fontlabelmultiplier = 1,
  smaller.fontlabelmultiplier = 1
)
```

Arguments

`object` An object of class `"rls"`. The function `itemlist` generates the class `"rls"` object. Printing the `""rls"` object will give a list of structural objects (that define the layout structure) to aid in defining the randomisation objects in the restricted layout structure.

`randomisation.objects`
This argument takes the format of the `TransferObject` item from the class `"rls"` object. The first column contains the names of all the structural objects in the layout structure (automatically generated) and the second column contains the corresponding randomisation objects in the restricted layout structure (manually generated). To begin with, the function `itemlist` should be run to generate the class `"rls"` object. The user will then need to edit the second column of the `TransferObject` matrix to define the randomisation objects that appear in the restricted layout structure. Structural objects that do not occur in the restricted layout structure must be left as "NULL" in the second column. The names specified in the second column represent the labels of the randomisation objects on the Hasse diagram of the restricted layout structure.

`random.arrows` A matrix of two columns that takes integer entries. The entries in the first column contain the object(s) at the start of the randomisation arrow and the second column contains the object(s) at the end. The values correspond to the entry number for the randomisation object in the `TransferObject` item from the class `"rls"` object). The first column specifies the randomisation object(s) at the beginning of the randomisation arrow(s) and the second column specifies the randomisation object(s) at the end of the arrow. The randomisation arrows must point downwards, hence, in each row of the matrix the second column entry must be larger than the first column entry. The randomisation object(s) involved in the randomisation arrow(s) must first be specified in the randomisation.objects argument.

`showRLS` logical. If "N" then generation of the Hasse diagram of the restricted layout structure is suppressed. The default is "Y".

`showpartialRLS` logical. If "N" then the partial crossing between randomisation objects (using dotted connecting lines) is not illustrated on the Hasse diagram of the restricted layout structure. The default is "Y".

`showdfRLS` logical. If "N" then the randomisation object label is not displayed on the Hasse diagram of the restricted layout structure. The default is "Y".

`showrandRLS` logical. If "N" then the randomisations are not illustrated (using arrows) on the Hasse diagram of the restricted layout structure. The default is "Y". If `random.arrows=NULL`, then `showrandRLS` defaults to "N".

`check.confound.df`
logical. If "N" then the check for confounded degrees of freedom is not performed. The default is "Y".

`maxlevels.df` logical. If "N" then the potential maximum number of levels of a generalised factor is removed from the randomisation object label on the Hasse diagram of the restricted layout structure. The default is "Y".

`table.out` logical. If "Y" then a table that shows the relationships between the randomisation objects in the restricted layout structure is printed. The default is "N".

`equation.out` logical. If "Y" then a recommended mixed model to use in the statistical analysis is printed. The default is "N".

`pdf` logical. If "Y" then a pdf file containing the Hasse diagram of the restricted layout structure is generated. The default is "N", i.e., a pdf file is not generated.

`example` character. Filename for the pdf output file containing the Hasse diagram. The default is set to "example".

`outdir` character. Location of the pdf output file if `pdf="Y"`. The default is set to `NULL` and in this case the pdf output file containing the Hasse diagram output is stored to a temporary file. To specify a permanent location this argument needs be specified.

`hasse.font` character. The name of the font family used for all text included on the Hasse diagram. Standard and safe font families to choose are "sans", "serif", and "mono". If any of the labels of the randomisation objects (as defined in the second column of `randomisation.objects` matrix) contain Unicode characters, a Unicode friendly font family should be selected. For more details on Unicode friendly family options see the Details section. If the font family selected fails to render, the font is automatically changed to "sans" instead. The default is "sans".

`produceBWPlot` logical. If "Y" then the Hasse diagram will be generated in black and white format. The default is set to "N", i.e., a coloured version of the plot is produced.

`structural.colour` character. The colour of the structural lines that connect randomisation objects on the Hasse diagram. The default colour is "grey".

`structural.width` numeric. The width of the structural lines on the Hasse diagram. The default width is 2.

`partial.colour` character. The colour of the partial crossing dotted lines of the connecting randomisation objects on the Hasse diagram. The default colour is "orange".

`partial.width` numeric. The width of the partial crossing dotted lines on the Hasse diagram. The default width is 1.5.

`objects.colour` character. The colour of the labels of the randomisation objects on the Hasse diagram. The default colour is "mediumblue".

`df.colour` character. The colour of the degrees of the freedom labels on the Hasse diagram. The default colour is "red".

`arrow.colour` character. The colour of the randomisation arrows on the Hasse diagram. The default colour is "mediumblue".

`arrow.width` numeric. The randomisation arrows width on the Hasse diagram. The default width is 1.5.

`arrow.pos` numeric. Specifies the position of the randomisation arrows, i.e., how far the randomisation arrows will be from the objects they point at. The default is 7.5. A smaller number specifies longer arrows and a higher number specifies shorter arrows.

`larger.fontlabelmultiplier` numeric. The large font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram where there are four or less objects at that level in the diagram. The default is 1.

`middle.fontlabelmultiplier` numeric. The medium font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram involving a factor that is equivalent to a generalised factor. The default is 1.

`smaller.fontlabelmultiplier` numeric. The small font multiplier is the multiplier for the font used for the labels of objects on the Hasse diagram where there are five or more objects at that level of the diagram. The default is 1.

Details

The `hasserls` function generates the Hasse diagram of the restricted layout structure. The Hasse diagram consists of a set of randomisation objects, corresponding to the factors and generalised factors, and the relationships between the objects (either crossed, nested, partially crossed or equivalent), as defined by the structure of the experimental design and the randomisation performed, see Bate and Chatfield (2016b).

The function requires an object of class `"rls"` containing the structural objects in the layout structure.

Where present, two partially crossed factors are illustrated on the diagram with a dotted line connecting them. This feature can be excluded using the `showpartialRLS` option.

The maximum number of possible levels of each generalised factor, along with the actual number present in the design and the "skeleton ANOVA" degrees of freedom, can be included in the randomisation object label on the Hasse diagram.

The randomisation arrows that illustrate the randomisation performed can be included on the Hasse diagram.

The `hasserls` function evaluates the design in order to identify if there are any confounded degrees of freedom across the design. It is not recommended to perform this evaluation for large designs, due to the potential high computational cost. This can be controlled using the `check.confound.df = "N"` option.

Objects that contain Unicode characters, e.g., u2192 or u2297 must be handled by Unicode friendly font families. Common font families that work with Unicode characters are: for Windows: Cambria, Embrima, Segoe UI Symbol, Arial Unicode MS, and for macOS: AppleMyungjo, .SF Compact Rounded, Arial Unicode MS, .SF Compact, .SF NS Rounded. The aforementioned fonts may not not be available in your R session. The available system fonts can be printed by systemfonts::system_fonts()$family. System available fonts can be imported by running showtext::font_import() or extrafont::font_import(). To check which fonts have been successfully imported, run showtext::fonts() or extrafont::fonts(). The Arial Unicode MS font can be downloaded from online sources. The Noto Sans Math font can be installed using sysfonts::font_add_google("Noto Sans Math").

Value

The function `hasserls` returns: 1. The Hasse diagram of the restricted layout structure (if `showRLS = "Y"`).

2. The restricted layout structure table shows the relationships between the randomisation objects in the restricted layout structure (if `table.out="Y"`). The individual entries in the table consist of blanks on the main diagonal and 0's, (0)'s or 1's elsewhere. If the factor (or generalised factor) corresponding to the ith row and the factor (or generalised factor) corresponding to the jth column are fully crossed, then a 0 is entered in the (i,j)th entry in the table. If these factors (or generalised factors) are partially crossed, or the ith row factor (or generalised factor) only has one level and nests the jth column factor (or generalised factor), then the (i,j)th entry is (0). If the ith row factor (or generalised factor) is nested within the jth column factor (or generalised factor), then a 1 is entered in the (i,j)th entry. If two factors (or generalised factor) are equivalent, then they share a single row and column in the table, where the row and column headers include both factor (or generalised factor) names, separated by an "=" sign.

3. An equation that suggests the mixed model to be fitted (if `equation.out="Y"`).

4. If there are confounded degrees of freedom, a table of the structural objects and a description of the associated degrees of freedom is printed.

Author(s)

Damianos Michaelides, Simon Bate, and Marion Chatfield

References

Bate, S.T. and Chatfield, M.J. (2016a), Identifying the structure of the experimental design. Journal of Quality Technology, 48, 343-364.

Bate, S.T. and Chatfield, M.J. (2016b), Using the structure of the experimental design and the randomization to construct a mixed model. Journal of Quality Technology, 48, 365-387.

Box, G.E.P., Hunter, J.S., and Hunter, W.G., (1978), Statistics for Experimenters. Wiley.

Joshi, D.D. (1987), Linear Estimation and Design of Experiments. Wiley Eastern, New Delhi.

Williams, E.R., Matheson, A.C. and Harwood, C.E. (2002), Experimental design and analysis for tree improvement. 2nd edition. CSIRO, Melbourne, Australia.

Examples

```
## NOTE TO USERS:
## Some examples below need Unicode symbols (e.g., "u2297 and "u2192"
## with a backslash, for the Kronecker and arrow symbols respectively),
## but we use ASCII fallbacks such as "x" and "->" in the code to ensure
## compatibility across systems.
## To render proper Unicode symbols in diagrams, update the labels manually
## i.e., x to \ u2297 to get the Kronecker symbol and -> to \ u2192 to get the
## arrow symbol, and set a Unicode-friendly font via the hasse.font argument.
## See the hasse.font argument and Details section in ?hasserls for guidance.

## Example: Asphalt concrete production (fractional factorial design)
concrete_objects <- itemlist(datadesign = concrete)
concrete_rls <- concrete_objects$TransferObject
concrete_rls[, 2] <- concrete_rls[, 1]
concrete_rls[27, 2] <- "AC^AG^CC^CoT^CuT -> Run"
hasserls(object = concrete_objects,
         randomisation.objects = concrete_rls,
         larger.fontlabelmultiplier = 1.6,
         smaller.fontlabelmultiplier = 1.3,
         table.out = "Y", arrow.pos = 8,
         hasse.font = "sans")

## Example: Crossover dental study
dental_objects <- itemlist(datadesign = dental, randomfacsid = c(0,1,0,0,0))
dental_rand_arrows <- matrix(c(3, 5, 4, 7), ncol = 2, byrow = TRUE)
dental_rls <- dental_objects$TransferObject
dental_rls[c(1:5,7,8), 2] <- c("Mean", "Period", "Sequence",
                               "Treatment", "Subject[Sequence]",
                               "Period x Sequence",
                               "Observation")
hasserls(object = dental_objects,
         randomisation.objects = dental_rls,
         random.arrows = dental_rand_arrows,
         larger.fontlabelmultiplier = 1.6,
         table.out = "Y", equation.out = "Y",
         arrow.pos = 15, hasse.font = "sans")

## Example: Human-computer interaction study
```

```
human_objects <- itemlist(datadesign = human,
                          randomfacsid = c(1,1,0,0,0,0,1))
human_rand_arrows <- matrix(c(3, 12, 6, 13), ncol = 2, byrow = TRUE)
human_rls <- human_objects$TransferObject
human_rls[, 2] <- c("Mean", "Day", "Method", "Period", "Room", "Sequence",
                    "NULL", "Subject | Subject -> Day x Sequence",
                    "NULL", "NULL", "NULL", "Day x Room",
                    "Period x Room", "NULL",
                    "Test = Day^Method^Period^Room^Sequence")
hasserls(object = human_objects,
         randomisation.objects = human_rls,
         random.arrows = human_rand_arrows,
         larger.fontlabelmultiplier = 1.4,
         hasse.font = "sans")

## Example: Analytical method (cross-nested design)
analytical_objects <- itemlist(datadesign = analytical,
                               randomfacsid = c(0,0,1,1,1,0,0,0))
analytical_rand_arrows <- matrix(c(2, 19, 19, 20), ncol = 2, byrow = TRUE)
analytical_rls <- analytical_objects$TransferObject
analytical_rls[, 2] <- c("Mean", "Batch", "Site",
                         "Analyst[Site]", "Column[Site]",
                         "System[Site]", "NULL",
                         "{Analyst^Column}[Site]",
                         "{Analyst^System}[Site]", "NULL",
                         "{Column^System}[Site]",
                         "NULL", "NULL",
                         "{Analyst^Column^System}[Site] -> Run",
                         "NULL", "NULL", "NULL", "NULL",
                         "Preparation[Run]", "Injection[Run]")
hasserls(object = analytical_objects,
         randomisation.objects = analytical_rls,
         random.arrows = analytical_rand_arrows,
         showpartialRLS = "N", check.confound.df = "N",
         larger.fontlabelmultiplier = 1,
         smaller.fontlabelmultiplier = 1.6,
         hasse.font = "sans")

## Conditionally run examples requiring 'dae'
if (requireNamespace("dae", quietly = TRUE)) {
  data(BIBDWheat.dat, package = "dae")
  BIBDWheat <- BIBDWheat.dat[, -4]
  BIBDWheat$Plots <- 1:30
  BIBDWheat_objects <- itemlist(datadesign = BIBDWheat)
  IBDWheat_rand_arrows <- matrix(c(3, 4), ncol = 2, byrow = TRUE)
  IBDWheat_rls <- BIBDWheat_objects$TransferObject
  IBDWheat_rls[1:4, 2] <- c("Mean", "Blocks", "Varieties", "Plot[Block]")
  hasserls(object = BIBDWheat_objects,
           randomisation.objects = IBDWheat_rls,
           random.arrows = IBDWheat_rand_arrows,
           equation.out = "Y")

  data(Fac4Proc.dat, package = "dae")
```

```
  Fac4Proc <- Fac4Proc.dat[, -6]
  Fac4Proc_objects <- itemlist(datadesign = Fac4Proc)
  Fac4Proc_rls <- Fac4Proc_objects$TransferObject
  Fac4Proc_rls[, 2] <- Fac4Proc_rls[, 1]
  Fac4Proc_rls[16, 2] <- "Catal^Conc^Press^Temp -> Run"
  hasserls(object = Fac4Proc_objects,
           randomisation.objects = Fac4Proc_rls,
           showpartialRLS = "N", table.out = "Y",
           smaller.fontlabelmultiplier = 2,
           hasse.font = "sans")

  data(Casuarina.dat, package = "dae")
  Casuarina <- Casuarina.dat[, -7]
  Casuarina$Row <- paste(Casuarina$Reps, Casuarina$Rows)
  Casuarina$Col <- paste(Casuarina$Reps, Casuarina$Columns)
  Casuarina <- Casuarina[, -c(2, 3)]
  Casuarina_objects <- itemlist(datadesign = Casuarina,
                                randomfacsid = c(1,0,1,0,0,0))
  Casuarina_rand_objects <- c(1:6, 9, 13)
  Casuarina_rand_arrows <- matrix(c(3, 5, 4, 13), ncol = 2, byrow = TRUE)
  Casuarina_rls <- Casuarina_objects$TransferObject
  Casuarina_rls[Casuarina_rand_objects, 2] <- c("Mean", "Countries",
                                                "InocTime", "Provences",
                                                "Reps", "Countries^InocTime",
                                                "Inoc^Provences",
                                                "{Row x Col}[Rep]")
  hasserls(object = Casuarina_objects,
           randomisation.objects = Casuarina_rls,
           random.arrows = Casuarina_rand_arrows,
           check.confound.df = "N", showpartialRLS = "N",
           arrow.pos = 10,
           smaller.fontlabelmultiplier = 1.5,
           hasse.font = "sans")
} else {
  message("Install package 'dae' to run the final examples.")
}
```

---

human *A block design for an experiment in human-computer interaction*

Description

This is a block design to compare two methods (mouse and stylus) of drawing a map in a computer file. The design involved 12 subjects randomised in 6 days and 2 tests (morning/afternoon) within each day, across 2 rooms. The design is based on 2x2 Latin squares; see Example 7 Brien and Bailey (2006) for more details.

Usage

```
data(human)
```

Format

A data frame of 24 observations on 7 variables. The 7 variables/factors included in the design are:

Subject Categoric factor with levels 1-12.

Day Categoric factor with levels 1-6.

Room Categoric factor with levels A and B.

Period Categoric factor with levels Morning and Afternoon.

Method Categoric factor with levels Mouse and Stylus.

Sequence Categoric factor with levels 1 and 2.

Test Categoric factor with levels 1-24.

Source

Brien, C.J. and Bailey, R.A. (2006). "Multiple randomizations (with discussion)". *Journal of the Royal Statistical Society B*, 68, pp. 571-609.

Examples

```
data("human")
human
```

`itemlist` *Objects to be used for restricted layout structure Hasse diagram*

Description

Returns an object of class `"rls"` that contains the structural objects of the layout structure. The output can be used in a consecutive function to generate the Hasse diagram of the restricted layout structure of the experimental design.

Usage

```
itemlist(datadesign, randomfacsid = NULL)
```

Arguments

`datadesign`	A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables/factors in the experimental design. The dataframe should only include the variables/factors/columns that the user wants to evaluate in the consecutive `hasserls` function to generate the Hasse diagram of the restricted layout structure.
`randomfacsid`	An optional vector specifying whether the factors are defined as fixed (entry = 0) or random (entry = 1). The default choice is `NULL` and the function automatically sets all entries to 0. The length of the vector should be equal to the number of variables/factors in the design, i.e., the length of the vector should be equal to the number of columns of the argument `datadesign`.

Details

The function requires a dataframe containing the variables corresponding to the experimental factors only (i.e., no response variables). Using the `randomfacsid` argument, the factors and generalised factors that correspond to random effects can be identified.

Value

The function returns an object of class `"rls"` which is a list with the following components:

`design` - The design dataframe containing all of the factors present in the layout structure. This is identical to the input argument `datadesign`.

`finaleffectsnames` - A character vector consisting of all structural objects (factors and generalised factors) in the layout structure.

`finalstructure` - A table that shows the relationships between the structural objects in the layout structure.

`finaleffects` - A vector containing the final factor orders.

`finalrandomeffects` - A vector containing 0 or 1 entries for factors or generalised factors corresponding to fixed (entry = 0) or random (entry = 1) effects.

`confoundings` - Defines the equivalent factors and generalised factors.

`nfactors` - The number of factors/variables of the design dataframe.

`outputlistip1` - A list of items required to generate the restricted layout structure.

`nestednames` - List of matrices containing the nested version of the factor and generalised factor names.

`TransferObject` - A Nx2 matrix where N equals the total number of structural objects. The first column contains the structural objects in the layout structure, and the second column has "Mean" for the first entry and "NULL" for the remaining entries.

Author(s)

Damianos Michaelides, Simon Bate, and Marion Chatfield

References

Bate, S.T. and Chatfield, M.J. (2016a), Identifying the structure of the experimental design. Journal of Quality Technology, 48, 343-364.

Bate, S.T. and Chatfield, M.J. (2016b), Using the structure of the experimental design and the randomization to construct a mixed model. Journal of Quality Technology, 48, 365-387.

Box, G.E.P., Hunter, J.S., and Hunter, W.G., (1978), Statistics for Experimenters. Wiley.

Joshi, D.D. (1987), Linear Estimation and Design of Experiments. Wiley Eastern, New Delhi.

Williams, E.R., Matheson, A.C. and Harwood, C.E. (2002), Experimental design and analysis for tree improvement. 2nd edition. CSIRO, Melbourne, Australia.

Examples

```
# Examples using built-in data: concrete, dental, human, analytical

# Fractional factorial design for asphalt concrete production
concrete_objects <- itemlist(datadesign=concrete)
print(concrete_objects)

# Crossover design for a dental study
dental_objects <- itemlist(datadesign=dental, randomfacsid=c(0,1,0,0,0))
summary(dental_objects)

# Block design for an experiment assessing human-computer interaction
human_objects <- itemlist(datadesign=human, randomfacsid=c(1,1,0,0,0,0,1))
print(human_objects)

# Cross-nested design for an analytical method investigation
analytical_objects <- itemlist(datadesign=analytical,
                               randomfacsid=c(0,0,1,1,1,0,0,0))
summary(analytical_objects)

# Examples using data from the dae package (conditionally loaded)
if (requireNamespace("dae", quietly = TRUE)) {
  data(BIBDWheat.dat, package = "dae")
  BIBDWheat <- BIBDWheat.dat[, -4]
  BIBDWheat$Plots <- c(1:30)
  BIBDWheat_objects <- itemlist(datadesign=BIBDWheat)
  print(BIBDWheat_objects)
}
```

---

print.rls *Print and Summary of* `rls` *objects*

Description

Print and Summary of objects of class "rls".

Usage

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

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

Arguments

`x`	An object of class `"rls"`.
`...`	Additional arguments to be passed to other methods.
`object`	An object of class `"rls"`.

Value

The functions return the structural objects of the Layout Structure and a matrix that the users need to edit with the randomisation objects to pass in the `hasserls` function.

Note

For examples see `itemlist` and `hasserls`.

Author(s)

Damianos Michaelides, Simon Bate, and Marion Chatfield

Index

∗ **datasets**
    analytical, 2
    concrete, 3
    dental, 4
    human, 16

analytical, 2
as.data.frame, *5*

concrete, 3

dental, 4

hasselayout, 5, *7*
hasserls, *6*, 9, *12*, *13*, *17*, *20*
human, 16

itemlist, *10*, 17, *20*

print.rls, 19

summary.rls *(*print.rls*)*, 19