Package 'hassediagrams'

May 8, 2026

Type Package

Title Hasse Diagram of the Layout Structure and Restricted Layout Structure

Version 2.0

Date 2026-01-17

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 2026-01-17 15:30:02 UTC

Contents

`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 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 factors. The 6 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 factors. The 5 factors included in the design are:

Sequence Categoric factor with levels 1-6.

Patient 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 = TRUE,
  showpartialLS = TRUE,
  showdfLS = TRUE,
  check.confound.df = TRUE,
  maxlevels.df = TRUE,
  table.out = FALSE,
  pdf = FALSE,
  example = "example",
  outdir = NULL,
  hasse.font = "sans",
  produceBWPlot = FALSE,
  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 factors in the experimental design. The data frame should **only** include the factors/columns that the user wants to include in the Hasse diagram. All factors are treated as categorical. Moreover, the first two letters of factor names are used for interactions between factors so it is advised that these be unique.
`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 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 `FALSE` then generation of the Hasse diagram is suppressed. The default is TRUE.

`showpartialLS` logical. If `FALSE` 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 TRUE.

`showdfLS` logical. If `FALSE` then the structural object label is not displayed on the Hasse diagram of the layout structure. The default is TRUE.

`check.confound.df` logical. If `FALSE` then the check for confounded degrees of freedom is not performed. The default is TRUE.

`maxlevels.df` logical. If `FALSE` 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 TRUE.

`table.out` logical. If `TRUE` then a table that shows the relationships between the structural objects in the layout structure is printed. The default is FALSE.

`pdf` logical. If `TRUE` then a pdf file containing the Hasse diagram of the layout structure is generated. The default is FALSE, 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=TRUE`. 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 `TRUE` then the Hasse diagram will be generated in black and white format. The default is set to FALSE, 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 factors corresponding to the experimental factors that define the experimental design (i.e., no response should be included).

In the dataframe the levels of the factors must be uniquely identified and have a physical meaning, otherwise the function will not correctly identify the nesting/crossing of the 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 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 = FALSE option.

Value

The function `hasselayout` returns:

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

2. The layout structure table shows the relationships between the structural objects in the layout structure (if `table.out=TRUE`). 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. Confounded degrees of freedom often indicate that the design objects are not specified appropriately (for example a factor or pseudofactor or supremum may have been missed from the design supplied) or that the design needs to be changed. However, sometimes once appropriate randomization is performed the layout structure, modified to account for this, no longer contains confounded degrees of freedom.

In addition, the function returns an object of class `"ls"`, which is a list with the following components:

`str_objects` A character vector containing the names of all structural objects derived from the design.

`rand_template` A character vector giving a template for use in `hasserls`. Structural objects not present in the restricted layout structure are set to `"NULL"`.

`str_rand_map` A two-column matrix pairing/mapping all structural objects with their suggested randomisation entries. This matrix assists users when constructing the `rand.objects` argument for `hasserls`.

`notes` A short explanatory message describing how to use the returned objects when preparing inputs for `hasserls`.

The returned object allows programmatic access to the structural objects and can be passed to `summary()` or `print()` methods for compact inspection.

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=TRUE)

## 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=FALSE, check.confound.df=FALSE,
            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=FALSE,
              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=FALSE,
             showpartialLS=FALSE)

} 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(
  datadesign,
  rand.objects,
  rand.arrows = NULL,
  randomfacsid = NULL,
  showRLS = TRUE,
  showpartialRLS = TRUE,
  showdfRLS = TRUE,
  showrandRLS = TRUE,
  check.confound.df = TRUE,
  maxlevels.df = TRUE,
  table.out = FALSE,
  equation.out = FALSE,
  pdf = FALSE,
  example = "example",
  outdir = NULL,
  hasse.font = "sans",
  produceBWPlot = FALSE,
  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

`datadesign`	A data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the factors in the experimental design. The data frame should **only** include the factors/columns that the user wants to include in the Hasse diagram. All factors are treated as categorical. Moreover, the first two letters of factor names are used for interactions between factors so it is advised that these be unique.
`rand.objects`	A character vector specifying the randomisation objects that define the Restricted Layout Structure (RLS). The vector must have the same length and order as the structural objects of the layout structure as produced by `hasselayout`. Each element is either `"NULL"` (if there is no randomisation object in the restricted layout structure that corresponds to the structural object) or the name of the randomisation object corresponding to the structural object. The user supplies labels for the randomisation objects The recommended workflow is: 1. Run `hasselayout` to obtain the list of structural objects (in the order required for `hasserls`) and a template for the randomisation objects. 2. Observe the structural objects using `hasselayout(datadesign)$str_objects` and create a vector of the randomisation objects potential using the suggested vector using `hasselayout(datadesign)$rand_template`. 3. Modify only those entries that correspond to randomisation objects present in the restricted layout structure. The labels specified in `rand.objects` represent the labels of the randomisation objects on the Hasse diagram of the restricted layout structure. If the labels include Unicode symbols (e.g., the Kronecker product symbol), a Unicode-friendly font is required.
`rand.arrows`	A matrix of two columns that takes integer entries. Each row of the matrix corresponds to one randomisation arrow on the Hasse diagram of the restricted layout structure. 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 `rand.objects`. Therefore, any randomisation object(s) involved in the randomisation arrow(s) must first be specified in the `rand.objects` argument. 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.
`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 factors in the design, i.e., the length of the vector should be equal to the number of columns of the argument datadesign.

showRLS logical. If FALSE then generation of the Hasse diagram of the restricted layout structure is suppressed. The default is TRUE.

showpartialRLS logical. If FALSE 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 TRUE.

showdfRLS logical. If FALSE then the randomisation object label is not displayed on the Hasse diagram of the restricted layout structure. The default is TRUE.

showrandRLS logical. If FALSE then the randomisations are not illustrated (using arrows) on the Hasse diagram of the restricted layout structure. The default is TRUE. If rand.arrows=NULL, then showrandRLS defaults to FALSE.

check.confound.df
logical. If FALSE then the check for confounded degrees of freedom is not performed. The default is TRUE.

maxlevels.df logical. If FALSE 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 TRUE.

table.out logical. If TRUE then a table that shows the relationships between the randomisation objects in the restricted layout structure is printed. The default is FALSE.

equation.out logical. If TRUE then a recommended mixed model to use in the statistical analysis is printed. The default is FALSE.

pdf logical. If TRUE then a pdf file containing the Hasse diagram of the restricted layout structure is generated. The default is FALSE, 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=TRUE. 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 rand.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 TRUE then the Hasse diagram will be generated in black and white format. The default is set to FALSE, 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).

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 = FALSE option.

The rendering of Unicode symbols (e.g., u2297, u2192 for Kronecker symbol and arrow, respectively) in the Hasse diagram depends on the operating system and the font selected in hasse.font.

macOS / Linux: Most system fonts that support Unicode work directly in plotting and PDF output without explicit registration. In many cases, the default "sans" family is sufficient for PDF rendering of these symbols. However, for on-screen rendering usually unicode-friendly fonts like "AppleMyungjo", "Arial Unicode MS", .SF Compact, and "Noto Sans Math" are needed.

Windows: Base R plotting often requires explicit font registration before the font can be used. Even if the font is installed, it may not be accessible to the graphics device until registered. Unicode-friendly fonts are "Lucida Sans Unicode", "Arial Unicode MS", "Segoe UI Symbol", "Cambria", "Noto Sans Math" and "Ebrima". The aforementioned fonts may 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"). For Windows, fonts might not be accessible to the graphics device until registered using: windowsFonts(LucidaSansUnicode = windowsFont("Lucida Sans Unicode")) adapted to the font need to use.

For ease of execution in the examples below, we are using –> for the arrow symbol and (x) for the kronecker symbol.

Value

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

2. The restricted layout structure table shows the relationships between the randomisation objects in the restricted layout structure (if table.out=TRUE). 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=TRUE).

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:
## In the examples below you may use 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 "-->" to ensure
## compatibility across systems.
## To render proper Unicode symbols in diagrams, update the labels manually
## and set a Unicode-friendly font via the hasse.font argument.

### Example: Asphalt concrete production (fractional factorial design)
## Obtain the structural objects from the layout structure
ls_concrete <- hasselayout(datadesign = concrete,
                           showLS = FALSE,
                           showpartialLS = FALSE,
                           showdfLS = FALSE)
## Observe the structural objects and then use the suggested
## template for randomisation objects
ls_concrete$str_objects
rand_spec <- ls_concrete$rand_template
## Fill in the randomisation objects that occur in the RLS
rand_spec[] <- ls_concrete$str_objects
rand_spec[length(rand_spec)] <- "AC^AG^CC^CoT^CuT --> Run"

## Generate the Hasse diagram of the restricted layout structure
hasserls(datadesign = concrete,
         rand.objects = rand_spec,
         larger.fontlabelmultiplier = 1.6,
         smaller.fontlabelmultiplier = 1.3)


### Example: Crossover dental study
## Obtain the structural objects from the layout structure
ls_dental <- hasselayout(datadesign = dental,
                         randomfacsid = c(0, 1, 0, 0, 0),
                         showLS = FALSE,
                         showpartialLS = FALSE,
                         showdfLS = FALSE)
## Observe the structural objects and then use the suggested
## template for randomisation objects
```

```
ls_dental$str_objects
rand_spec <- ls_dental$rand_template
## Fill in the randomisation objects that occur in the RLS
rand_spec[c(2:5, 7, 8)] <- c("Period", "Sequence",
                             "Treatment", "Subject[Sequence]",
                             "Period (x) Sequence",
                             "Observation")
## Create a matrix for the randomisation arrows
dental_rand_arrows <- matrix(c(3, 5, 4, 7), ncol = 2, byrow = TRUE)
## Generate the Hasse diagram of the restricted layout structure
hasserls(datadesign = dental,
         rand.objects = rand_spec,
         rand.arrows = dental_rand_arrows,
         randomfacsid = c(0, 1, 0, 0, 0),
         larger.fontlabelmultiplier = 1.6,
         arrow.pos = 15)


## Conditionally run examples requiring 'dae'
if (requireNamespace("dae", quietly = TRUE)) {
  data(BIBDWheat.dat, package = "dae")
  BIBDWheat <- BIBDWheat.dat[, -4]
  BIBDWheat$Plots <- 1:30
ls_BIBDWheat <- hasselayout(datadesign = BIBDWheat,
                            showLS = FALSE,
                            showpartialLS = FALSE,
                            showdfLS = FALSE)
## Observe the structural objects and then use the suggested
## template for randomisation objects
ls_BIBDWheat$str_objects
rand_spec <- ls_BIBDWheat$rand_template
## Fill in the randomisation objects that occur in the RLS
rand_spec[c(2:4)] <- c("Blocks", "Varieties", "Plot[Block]")
## Create a matrix for the randomisation arrows
IBDWheat_rand_arrows <- matrix(c(3, 4), ncol = 2, byrow = TRUE)
## Generate the Hasse diagram of the restricted layout structure
hasserls(datadesign = BIBDWheat,
         rand.objects = rand_spec,
         rand.arrows = IBDWheat_rand_arrows,
         equation.out = TRUE)


data(Fac4Proc.dat, package = "dae")
Fac4Proc <- Fac4Proc.dat[, -6]
## Obtain the structural objects from the layout structure
ls_Fac4Proc <- hasselayout(datadesign = Fac4Proc,
                           showLS = FALSE,
                           showpartialLS = FALSE,
                           showdfLS = FALSE)
## Observe the structural objects and then use the suggested
## template for randomisation objects
ls_Fac4Proc$str_objects
rand_spec <- ls_Fac4Proc$rand_template
```

```
## Fill in the randomisation objects that occur in the RLS
rand_spec[] <- ls_Fac4Proc$str_objects
rand_spec[length(rand_spec)] <- "Catal^Conc^Press^Temp --> Run"
## Generate the Hasse diagram of the restricted layout structure
hasserls(datadesign = Fac4Proc,
         rand.objects = rand_spec,
         showpartialRLS = FALSE,
         smaller.fontlabelmultiplier = 2)

} 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 factors The 7 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
```

`print.ls` *Print and Summary Methods for* `"ls"` *Objects*

Description

These functions provide printing and summarising methods for objects of class `"ls"`, which are returned from `hasselayout`.

Usage

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

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

Arguments

`x`	An object of class `"ls"` for `print.ls`.
`...`	Additional arguments for consistency with S3 methods (unused).
`object`	An object of class `"ls"` for `summary.ls`.

Details

Objects of class `"ls"` contain: (i) the structural objects of the layout structure; (ii) a template randomisation objects vector; and (iii) a combined mapping matrix showing structural objects alongside editable placeholders for the randomisation objects used in `hasserls`.

The `print` method displays the structural objects. The `summary` method returns a structured object for programmatic use.

Value

`print.ls` Prints the structural objects to the console. Invisibly returns the original `"ls"` object so that it may be used in further computations.

`summary.ls` Returns a list with class `"summary.ls"` containing:

- `str_objects` – vector of structural objects.
- `rand_template` – a template vector indicating how to populate `rand.objects` for `hasserls()`.
- `str_rand_map` – two-column matrix showing structural objects and user-editable randomisation objects.
- `notes` – guidance text for using this information in `hasserls()`.

Unlike `print.ls`, this method is designed for programmatic inspection, returning structured data without printing.

Note

See hasserls for examples demonstrating how to use the output of hasselayout() to construct the restricted layout structure.

Author(s)

Damianos Michaelides, Simon Bate, and Marion Chatfield

Index