Package 'eatDesign'

December 18, 2012

Type Package

Title eatDesign

Version 0.0.2

Depends R (>= 2.14.0), methods, igraph

Imports Matrix

Date 2012-11-27

Author Martin Hecht, Nicole Haag, Karoline Sachse, Thilo Siegle and Sebastian Weirich

Maintainer Martin Hecht <martin.hecht@iqb.hu-berlin.de>

Description compute descriptives of data structures (''designs'')

License GPL (>= 2)

LazyLoad yes

LazyData yes

R topics documented:

`defineDesign` *define a data structure ("design")*

Description

This function can be used to define a design. An object of class `"design"` is created. Descriptives can be computed.

Usage

```
defineDesign ( dsgn , def = data.frame() , append = FALSE , descriptives = TRUE ,
verbose = FALSE )
```

Arguments

`dsgn`	object of class `"design"`
`def`	a data frame containing design elements in columns and units of these elements in rows, the data frame must be in "long" format, so that design elements are in columns and each row defines the units of the design elements that are combined, see example
`append`	logical, append `def` to `dsgn` or overwrite
`descriptives`	logical, compute descriptives (can be time consuming)
`verbose`	logical, print information while processing

Value

returns an object of class `"design"`

Warning

This version is alpha. Use with care.

Author(s)

Martin Hecht

See Also

`updateDesign` `design-class`

Examples

```
# Table 7 (Frey, 2009)
table7 <- data.frame ( "Booklet" = c(1,1,2,2,3,3) , "Position" = c(1,2,1,2,1,2) ,
"Cluster" = c(1,2,2,3,3,1) )

# create a new design object
( design7 <- new ( "design" ) )

# use table7 as the definition of the design
design7 <- defineDesign ( dsgn = design7 , def = table7 )
```

```
# print design object (with auto-generated descriptives)
design7

# add some more cases, using option 'append' in 'defineDesign'
add <- data.frame ( "Booklet" = c(4,4) , "Position" = c(3,4) , "Cluster" = c(4,5) )
( design8 <- defineDesign ( dsgn = design7 , def = add , append = TRUE ) )

# add even more cases, this time using '+' operator on 2 designs
# (results are the same, but one of either method might be more convenient)
add2 <- data.frame ( "Booklet" = c(5,5,6,6) , "Position" = c(5,6,5,6) , "Cluster" = c(6,7,7,8) )
( design2 <- defineDesign ( def = add2 ) )
( design9 <- design7 + design2 )

# add items that are nested within clusters
add3 <- data.frame ( "Item" = paste ( "item" , 1:12 , sep = "" ) ,
"Cluster" = as.vector ( sapply ( 1:3 , rep , 4 ) ) )
( design3 <- defineDesign ( def = add3 ) )
( design10 <- design3 + design7 )
```

`design-class` *Class* `"design"`

Description

A design object contains definition and descriptives of a data structure ("design").

Objects from the Class

Objects can be created by calls of the form `new("design")`.

Slots

`definition`: Object of class `"data.frame"`

contains the definition of the design

`elements`: Object of class `"character"`

contains the names of the elements of the design (these are simply the column names of `definition`

`units`: Object of class `"list"`

contains the unique units of `elements`

`nunits`: Object of class `"integer"`

contains the number of `units`

`structure`: Object of class `"data.frame"`

contains information on the relation between `elements`, the relation between two elements can be "equivalent", "unconnected", "nested", "nestor" (this is the grouping variable of "nested" units), "crossedpartially" or "crossedcompletely"

`structureList`: Object of class `"list"`

contains units of one element in relation to one unit of another element

`descriptives`: Object of class `"data.frame"`
: contains information on the number of units of one element with reference to another element

`linkList`: Object of class `"list"`
: contains a graph of class "igraph" for each pair of `elements`

`adjacency`: Object of class `"list"`
: contains "adjacency" matrices, see get.adjacency

`link`: Object of class `"data.frame"`
: contains link descriptives:
: average path length, see average.path.length
: relative frequency of realized (unique) pairwise links in reference to all possible pairwise links
: relative frequency of realized pairwise links in reference to all theoretically possible pairwise links if elements were completely crossed
: mean degree of units, see degree
: standard deviation of degree of units, see degree

`varCovMatrix`: Object of class `"matrix"`, contains the Variance-Covariance Matrix of the design, see Frey (2009) for details

`designDescriptives`: Object of class `"list"`, contains the D-optimality index that is computed from `varCovMatrix`

Methods

show `signature(object = "design")`: displays an object of class "design"

`+ signature(e1 = "design", e2 = "design")`: add one design to another ("merge" two designs)

`- signature(e1 = "design", e2 = "design")`: distract one design from another, this is functional only for designs that contain the same elements

Warning

This version is alpha. Use with care.

Author(s)

Martin Hecht

References

Frey, A., Hartig, J., & Rupp, A. A. (2009). An NCME Instructional Module on Booklet Designs in Large-Scale Assessments of Student Achievement: Theory and Practice. Educational Measurement: Issues and Practice, 28(3), 39-53.

See Also

`defineDesign`

`updateDesign`

Examples

```
showClass("design")
```

updateDesign *update an object of class "design"*

Description

This function can be used to update a design. This might be useful to compute descriptives on a previously created design object.

Usage

```
updateDesign ( dsgn , descriptives = TRUE , verbose = FALSE )
```

Arguments

dsgn	Object of class "design"
descriptives	logical, compute descriptives (can be time consuming)
verbose	logical, print information while processing

Value

returns an object of class "design"

Warning

This version is alpha. Use with care.

Author(s)

Martin Hecht

See Also

defineDesign design-class

Examples

```
# Table 7 (Frey, 2009)
table7 <- data.frame ( "Booklet" = c(1,1,2,2,3,3) , "Position" = c(1,2,1,2,1,2) ,
"Cluster" = c(1,2,2,3,3,1) )

# create a new design object
( design7 <- new ( "design" ) )

# use table7 as the definition of the design, do not compute descriptives
( design7 <- defineDesign ( dsgn = design7 , def = table7 , descriptives = FALSE ) )

# compute descriptives
( design7 <- updateDesign ( dsgn = design7 , descriptives = TRUE ) )
```

Index