Package 'tangram'

October 2, 2017

Title The Grammar of Tables

Version 0.3.2

Description Provides an extensible formula system to quickly and easily create production quality tables. The steps of the process are formula parser, statistical content generation from data, to rendering. Each step of the process is separate and user definable thus creating a set of building blocks for highly extensible table generation. A user is not limited by any of the choices of the package creator other than the formula grammar. For example, one could chose to add a different S3 rendering function and output a format not provided in the default package. Or possibly one would rather have Gini coefficients for their statistical content. Routines to achieve New England Journal of Medicine style, Lancet style and Hmisc::summaryM() statistics are provided. The package contains rendering for HTML5, Rmarkdown and an indexing format for use in tracing and tracking are provided.

Author Shawn Garbett [aut, cre],
Thomas Stewart [ctb],
Jennifer Thompson [ctb],
Frank Harrell [ctb]

Maintainer Shawn Garbett <Shawn.Garbett@Vanderbilt.edu>

Depends R (>= 3.2.3), R6, magrittr

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat, rms, knitr, rmarkdown, Hmisc

Imports stringi, stringr, base64enc, digest, htmltools

VignetteBuilder knitr

RoxygenNote 6.0.1

Collate 'compile-cell.R' 'compile-operators.R' 'compile-post.R' 'parser.R' 'compile.R' 'compile-rms.R' 'compile-typing.R' 'helper-format.R' 'hmisc-cut2.R' 'hmisc-lm.fit.qr.bare.R' 'hmisc-impute.R' 'hmisc-biVar.R' 'render-html5.R' 'render-index.R' 'render-latex-map.R' 'render-latex.R'

'render-rmd.R' 'render-rtf.R' 'render-summary.R'
'transform-hmisc.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2017-10-02 21:03:18 UTC

R topics documented:

args_flatten *Flatten variable arguments*

Description

Take variable arguments, flatten vectors and lists, but do not flatten cells (which are lists) e.g. args_flatten(NA, list(1,2,3), 4:6, c(7,8,9))

Usage

```
args_flatten(...)
```

Arguments

...	variable arguments

Value

a list of the arguments, with vectors and lists flattened

as.categorical *Convert data type to a factor if it's not already*

Description

Convert data type to a factor if it's not already

Usage

```
as.categorical(x)
```

Arguments

x	Data to convert to factor

Value

Data as a factor

Examples

```
as.categorical(1:3)
```

`ASTBranch` *A left/right branch in an Abstract Syntrax Tree. This inherits from ASTNode, and is intended to be a base class as well. Should never be instantiated directly as once again the semantic information is contained in the class name.*

Description

A left/right branch in an Abstract Syntrax Tree. This inherits from ASTNode, and is intended to be a base class as well. Should never be instantiated directly as once again the semantic information is contained in the class name.

Usage

```
ASTBranch
```

Format

`R6Class` object.

Fields

`left` A pointer to the left node below this one

`right` A pointer to the right node below this one

Methods

`distribute()` Depth first application of distribute() to left and right nodes, then modifies left and right and returns self.

`terms()` Returns the node

`string()` Returns the string formula of the node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

`ASTFunction` *A specified function call as an ASTNode*

Description

A specified function call as an ASTNode

Usage

```
ASTFunction
```

Format

R6Class object.

Fields

`value` The name of the function.

`r_expr` A string containing the raw r expression from inside the parenthesis

Methods

`new(value, r_expr)` Create one with the given value and r_expr

`terms()` Returns the node

`factors()` Returns self as a factor

`distribute()` Applies the distributive property to the node, and returns the resulting node.

`string()` Returns the string formula of the node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

Examples

```
ASTFunction$new("log", "x+2")$string()
```

ASTMultiply *The multiplication of two terms, as an ASTNode.*

Description

The multiplication of two terms, as an ASTNode.

Usage

```
ASTMultiply
```

Format

R6Class object.

Fields

`left` The AST tree to the left.

`right` The AST tree to the right.

Methods

`new(left, right)` Create addition node of given left and right node.

`terms()` Returns the node as a term vector

`factors()` Returns all terminal nodes under this as a list

`distribute()` Applies the distributive property to the node, and returns the resulting node. This is the actual workhorse of the disributing multiplication across the tree.

`string()` Returns the string formula of the node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

Examples

```
ASTMultiply$new(ASTVariable$new("x"), ASTVariable$new("y"))$string()
```

ASTNode *A Node in an Abstract Syntax Tree (AST)*

Description

This is the root R6 class of any term of the AST which is created when parsing a table formula. This should only be used as a base class as the class information carries the semantic meaning of a given node.

Usage

```
ASTNode
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`symbol` A string which tells what this node in the AST represents.

`value` A string of addtional information contained by the node.

Methods

`terms()` Returns the node itself

`distribute()` Applies the distributive property to the node, and returns the resulting node.

`string()` Returns the string formula of the node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

ASTPlus *The addition of two terms, in an ASTNode.*

Description

The addition of two terms, in an ASTNode.

Usage

```
ASTPlus
```

Format

R6Class object.

Fields

`left` The AST tree to the left.

`right` The AST tree to the right.

Methods

`new(left, right)` Create addition node of given left and right node.

`terms()` Returns the left and right branches terms

`distribute()` Applies the distributive property to the node, and returns the resulting node.

`string()` Returns the string formula of the node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

Examples

```
ASTPlus$new(ASTVariable$new("x"), ASTVariable$new("y"))$string()
```

ASTTableFormula *The root ASTNode of a formula.*

Description

The root ASTNode of a formula.

Usage

```
ASTTableFormula
```

Format

R6Class object.

Fields

`left` The AST tree for the columns.

`right` The AST tree for the rows.

Methods

`new(left, right)` Create addition node of given left and right node.

`terms()` Returns the a list of the left hand terms and right hand terms

`distribute()` Applies the distributive property to the node, and returns the resulting node. This is the actual workhorse of the disributing multiplication across the tree.

`string()` Returns the string representation of the formula

`reduce(data)` Given a set of data, perform the logical reduction of the entire AST.

Examples

```
ASTTableFormula$new(ASTVariable$new("x"), ASTVariable$new("y"))$string()
```

ASTVariable *A Variable in an Abstract Syntax Tree (AST)*

Description

This node represents a variable of interest in the AST. A variable's name is recorded in the value field, and must conform to the rules of identifiers in R. This class inherits from ASTNode.

Usage

```
ASTVariable
```

Format

R6Class object.

Fields

`value` A string containing the variable identifier

`format` A format string that is either a string containing a number representing significant digits for output, or a C-style printf string.

`type` A string that represents the type specifier for that variable

Methods

`new(identifier, format=NA, type=NA)` This method creates an AST node representing a variable of a given identifier. An optional format consisting of a string of a number or a c-style printf string. An option type denoting a forced type cast of that variable.

`terms()` Returns the node

`distribute()` Applies the distributive property to the node, and returns the resulting node.

`string()` Returns the string formula of the node

`name()` Return a human representation of a node

`reduce(data)` Given a set of data, perform the logical reduction of the current node.

Examples

```
ASTVariable$new("x", "2", "Continuous")$string()
```

---

cell *Construct a table cell from an object*

Description

Any R object can be used as a cell value. Attributes are used to store additional classs of that cell attached to the object. This is a helper function to attach all the additional attributes to the provided object

Usage

```
cell(x, ...)
```

Arguments

x	R object to attach attributes too
...	Each additional argument becomes an attribute for the object

Details

Certain attributes have special meaning: - 'names' is appended to the front of a value, e.g. "P=" for a p-value. - 'sep' is used to join values, e.g. ", " for a list of values. - 'class' denotes special rendering handling, e.g. generally passed as CSS class to HTML5 - 'reference' a list of reference symbols to put inside the cell - 'row' and 'col' should refer to the row / column if key generation is needed - 'subrow' and 'subcol' further delinate the key value of a cell for key generation

Value

The modified R object

cell.aov *AOV model as cell*

Description

Construct a cell from an analysis of variance model

Usage

```
## S3 method for class 'aov'
cell(x, pformat = "%1.3f", ...)
```

Arguments

x	The aov object to turn into a renderable cell
pformat	numeric or character; A formatting directive to be applied to p-values
...	additional specifiers for identifying this cell (see key)

Value

an S3 rendereable cell that is an F-statistic

Examples

```
cell(aov(x~y,data.frame(x=rnorm(10), y=rnorm(10))))
```

cell.htest *Construct hypothesis test cell*

Description

Construct a cell from a hypothesis test

Usage

```
## S3 method for class 'htest'
cell(x, format = 2, pformat = "%1.3f", reference = NULL,
  ...)
```

Arguments

x	The htest object to convert to a rendereable cell
format	numeric or character; A formatting directive applied to statistics
pformat	numeric or character; A formatting directive to be applied to p-values
reference	numeric or character; A reference indicator for this test
...	additional specifiers for identifying this cell (see key)

Details

Currently handles cor.test, t.test and chisq.test objects

Value

an S3 rendereable cell that is a hypothesis test

Examples

```
cell(cor.test(rnorm(10), rnorm(10), method="spearman"))
cell(cor.test(rnorm(10), rnorm(10)))
cell(chisq.test(rpois(10,1)))
cell(t.test(rnorm(10), rnorm(10)))
```

---

cell_chi2 *Create an cell_chi2 (S3) object of the given statistic*

Description

A cell_chi2 object contains a statistical result of an X^2-test.

Usage

```
cell_chi2(chi2, df, p, class = NULL, ...)
```

Arguments

chi2	The value of the X^2 statistic
df	degrees of freedom
p	p-value of resulting test
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_chi2 object.

Examples

```
cell_chi2(5.6, 2, 0.06081)
```

cell_estimate *Create a cell_estimate object of the given estimate*

Description

A cell_estimate object contains a statistical estimate. It may additionally contain an interval with a low and high of a specified width.

Usage

```
cell_estimate(value, low, high, name = NULL, class = NULL, sep = ", ",
  ...)
```

Arguments

value	The value of the estimate
low	Specifies a lower interval for the estimate.
high	Specifies an upper interval for the estimate.
name	character; An optional name to apply to the value
class	character; additional classs to apply
sep	character; option separator character for the range
...	optional extra information to attach

Value

A cell_estimate object.

Examples

```
cell_estimate(1.0, 0.5, 1.5)
cell_estimate(1.0, 0.5, 1.5, name="one")
```

cell_fraction *Create an cell_fraction (S3) object of the given statistic*

Description

A cell_fraction object contains a statistical result of a fraction/percentage.

Usage

```
cell_fraction(numerator, denominator, format = 3, class = NULL, ...)
```

Arguments

numerator	numeric; The value of the numerator
denominator	numeric; The value of the denominator
format	numeric or character; a string formatting directive
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_fraction object.

Examples

```
cell_fraction(1, 4, 0.25, 25)
```

---

cell_fstat *Create an cell_fstat (S3) object of the given statistic*

Description

A cell_fstat object contains a statistical result of an F-test.

Usage

```
cell_fstat(f, df1, df2, p, class = NULL, ...)
```

Arguments

f	The value of the f-statistic
df1	1st dimension degrees of freedom
df2	2nd dimension degrees of freedom
p	The p-value of the resulting test
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_fstat object.

Examples

```
cell_fstat(4.0, 10, 20, 0.004039541, reference=1,)
```

cell_header *Create a cell_header object of the given text.*

Description

A cell_header object represents a label cell inside a table. It can also contain units.

Usage

```
cell_header(text, units = NULL, class = NULL, ...)
```

Arguments

text	character; The text of the label. May include a subset of LaTeX greek or math.
units	character; An optional field that contains units
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_header object

Examples

```
cell_header("Yahoo")
cell_header("Concentration", "mg/dl")
cell_header("Concentration", "mg/dl", src="A")
```

cell_iqr *Create a interquartile range cell object of the given data*

Description

Construct a cell which has the 3 interquartile ranges specified.

Usage

```
cell_iqr(x, format = NA, na.rm = TRUE, names = FALSE, type = 8,
  msd = FALSE, quant = c(0.25, 0.5, 0.75), ...)
```

Arguments

x	numeric vector whose sample quantiles are wanted. NA and NaN values are not allowed in numeric vectors unless na.rm is TRUE.
format	numeric or character; Significant digits or fmt to pass to sprintf
na.rm	logical; if true, any NA and NaN's are removed from x before the quantiles are computed.
names	logical; if true, the result has a names attribute. Set to FALSE for speedup with many probs.
type	integer; specify algorithm to use in constructing quantile. See quantile for more information.
msd	logical; compute an msd attribute containing mean and standard deviation
quant	numeric; The quantiles to display. Should be an odd length vector, since the center value is highlighted.
...	additional arguments to constructing cell

Value

A cell_quantile object.

Examples

```
require(stats)
cell_iqr(rnorm(100), '3')
```

---

cell_label *Create an cell_label (S3) object of the given text.*

Description

A cell_label object represents a label cell inside a table. It can also contain units.

Usage

```
cell_label(text, units = NULL, class = NULL, ...)
```

Arguments

text	character; The text of the label. May include a subset of LaTeX greek or math.
units	character; An optional field that contains units
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A tangram object

Examples

```
cell_label("Compaction Method")
cell_label("Concentration", "mg/dl")
cell_label("Concentration", "mg/dl", subcol="A")
```

`cell_n` *Create an cell_n (S3) object of the given statistic*

Description

A cell_n object contains a statistical result of an X^2-test.

Usage

```
cell_n(n, class = NULL, ...)
```

Arguments

`n`	The value of the X^2 statistic
`class`	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
`...`	optional extra information to attach

Value

A cell_n object.

Examples

```
cell_n(20)
```

cell_named_values *Create named value cells*

Description

A cell object with additionally contain an interval with a low and high of a specified width.

Usage

```
cell_named_values(values, names, class = NULL, sep = ", ", ...)
```

Arguments

values	vector; to create cell values from
names	character; names to apply to values
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
sep	character; separator to use when rendering
...	additional attributes to attach to cell

Value

A cell object with named values

Examples

```
cell_named_values(1.0, "one")
```

cell_range *Create a cell representing a range*

Description

Useful for things such as confidence intervals.

Usage

```
cell_range(low, high, class = NULL, sep = ", ", ...)
```

Arguments

low	character or numeric; lower value of range
high	character or numeric; upper value of range
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
sep	character; separator to use when rendering
...	additional attributes to attach to cell

Value

A cell object denoting a range

Examples

```
cell_range(-1.0, 1.0)
```

`cell_spearman` *Create an cell_spearman (S3) object of the given statistic*

Description

A cell_spearman object contains a statistical result of an spearman-test.

Usage

```
cell_spearman(S, rho, p, class = NULL, ...)
```

Arguments

`S`	The value of the spearman statistic
`rho`	The rho value of the test
`p`	p-value of resulting test
`class`	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
`...`	optional extra information to attach

Value

A cell_spearman object.

Examples

```
cell_spearman(20, 0.2, 0.05)
```

cell_studentt *Create an cell_studentt (S3) object of the given statistic*

Description

A cell_studentt object contains a statistical result of an t-test.

Usage

```
cell_studentt(t, df, p, class = NULL, ...)
```

Arguments

t	The value of the X^2 statistic
df	degrees of freedom
p	p-value of resulting test
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_studentt object.

Examples

```
cell_studentt(2.0, 20, 0.02963277)
```

cell_subheader *Create a cell_subheader object of the given text.*

Description

A cell_subheader object represents a label cell inside a table. It can also contain units.

Usage

```
cell_subheader(text, units = NULL, class = NULL, ...)
```

Arguments

text	character; The text of the label. May include a subset of LaTeX greek or math.
units	character; An optional field that contains units
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
...	optional extra information to attach

Value

A cell_subheader object.

Examples

```
cell_subheader("Concentration")
cell_subheader("Concentration", "mg/dl")
cell_subheader("Concentration", "mg/dl", src="A")
```

---

cell_transform *Create a function to transform all cells of a table*

Description

Given a function that operates on a table cell and returns the modified cell, return a function that given a table applies that function to all cells and returns the modified table.

Usage

```
cell_transform(FUN, ...)
```

Arguments

FUN	function to apply, must return the modified cell
...	additional arguments to pass into function

Value

a table modification function

---

custom_css *Return a CSS file as a string*

Description

Given a filename, this function will load the file name from the current working directory. If it is not found from the current working directory it will search in the package for a a matching filename and load that instead. If an id is specified, that will be prepended to all CSS selectors (TODO: make this substitution more robust). The result is returned as a string.

Usage

```
custom_css(filename, id = NA)
```

Arguments

filename	Name of the CSS file to load
id	CSS id to prepend to all entries

Value

String of possibly modified CSS file

Examples

```
custom_css("lancet.css", "tbl1")
```

---

del_col *Delete a given column from a table*

Description

Given a table, remove the specified column

Usage

```
del_col(table, col)
```

Arguments

table	the table to modify
col	the number of the column to drop

Value

the modified table

---

del_row *Delete a given row from a table*

Description

Given a table, remove the specified row

Usage

```
del_row(table, row)
```

Arguments

table	the table to modify
row	the number of the row to drop

Value

the modified table

---

derive_label *Derive label of AST node.*

Description

Determine the label of a given AST node. NOTE: Should have data attached via reduce before calling.

Usage

```
derive_label(node)
```

Arguments

node	Abstract syntax tree node.

Value

A string with a label for the node

---

drop_statistics *Drop all statistics columns from a table.*

Description

Delete from a table all columns that contain statistics

Usage

```
drop_statistics(table)
```

Arguments

table	the table to remove statistical columns

Value

the modified table

format_guess *Guess the best format for a given set of numerical data*

Description

Given a vector of data, default to 3 significant digits or all if maximum is greater than zero

Usage

```
format_guess(x)
```

Arguments

x	numeric; basic math and quantile function must work on data passed in

Value

numeric; the digits past the decimal recommended for display

Examples

```
format_guess(rnorm(100))
format_guess(rnorm(100, sd=1e-6))
```

hmisc_data_type *Determine data type of a vector loosely consistent with Hmisc.*

Description

Determine data type of a vector loosely consistent with Hmisc.

Usage

```
hmisc_data_type(x, category_threshold = NA)
```

Arguments

x	Vector to determine type of
category_threshold	The upper threshold of unique values for which a vector is considered categorical.

Value

One of the following strings: Binomial, Categorical, or Numerical.

Examples

```
hmisc_data_type(c(1,2,3))
hmisc_data_type(factor(c("A","B","C")))
hmisc_data_type(factor(c("A","B","B","A")))
hmisc_data_type(factor(c(TRUE, FALSE, TRUE, FALSE)))
```

---

hmisc_intercept_cleanup *Cleanup an intercept only model*

Description

Cleanup an intercept only table that was generated from the hmisc default transform. This drops the statistics column, and modifies the header to eliminate blank space.

Usage

```
hmisc_intercept_cleanup(table)
```

Arguments

table	the table to modify

Value

the modified table

---

hmisc_style *Style Bundle for Hmisc defaults.*

Description

List of lists, should contain a "Type" entry with a function to determine type of vector passed in. Next entries are keyed off returned types from function, and represent the type of a row. The returned list should contain the same list of types, and represents the type of a column. Thus it now returns a function to process the intersection of those two types.

Usage

```
hmisc_style
```

Format

An object of class `list` of length 4.

html5 *S3 html5 Method function for use on a tangram to generate HTML5*

Description

S3 html5 Method function for use on a tangram to generate HTML5

Usage

```
html5(object, id, ...)
```

Arguments

object	The cell to render to HTML5
id	A unique identifier for traceability in indexing
...	additional arguments to renderer.

html5.cell *Convert an abstract cell object into an HTML5 string*

Description

Given a cell class create an HTML5 representation.

Usage

```
## S3 method for class 'cell'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given cell as a <td> with several 's.

html5.cell_chi2 *Convert an abstract cell_chi2 object into an HTML5 string*

Description

Given a cell_chi2 class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_chi2'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell chi2 to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given chi2 as a <td> with several 's.

html5.cell_estimate *Convert a cell_estimate object into an HTML5 string*

Description

Given a cell_estimate class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_estimate'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell estimate to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given estimate as a <td> with several 's.

html5.cell_fraction *Convert an abstract cell_fraction object into an HTML5 string*

Description

Given a cell_fraction class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_fraction'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell fraction to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given fraction as a <td> with several 's.

html5.cell_fstat *Convert a cell_fstat object into an HTML5 string*

Description

Given a cell_fstat class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_fstat'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell fstat to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given fstat as a <td> with several 's.

html5.cell_header *Convert an abstract cell_header object into an HTML5 string*

Description

Given a cell_header class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_header'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell subheader to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	additional class attributes for CSS rendering

Value

A text string rendering of the given subheader as a <td> with several 's.

html5.cell_iqr *Convert a cell_iqr object into an HTML5 string*

Description

Given a cell_iqr class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_iqr'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell iqr to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given quantile as a <td> with several 's.

html5.cell_label *Convert a cell_label object into an HTML5 string*

Description

Given a cell_label class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_label'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell label to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given label as a <td> with several 's.

html5.cell_n *Convert an abstract cell_n object into an HTML5 string*

Description

Given a cell_n class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_n'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell n to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

A text string rendering of the given n as a <td> with several 's.

html5.cell_subheader *Convert an abstract cell_subheader object into an HTML5 string*

Description

Given a cell_subheader class create an HTML5 representation.

Usage

```
## S3 method for class 'cell_subheader'
html5(object, id, ..., class = NULL)
```

Arguments

object	The cell subheader to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	additional class attributes for CSS rendering

Value

A text string rendering of the given subheader as a <td> with several 's.

html5.character *Default conversion to HTML5 for a character cell*

Description

Produces table cell

Usage

```
## S3 method for class 'character'
html5(object, id, ..., class = NA)
```

Arguments

object	The cell to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

An empty html5 td of the given class

html5.default *Default conversion to HTML5 for an abstract table element*

Description

Gives a warning and produces an empty <td></td> cell

Usage

```
## Default S3 method:
html5(object, id, ..., class = NA)
```

Arguments

object	The cell to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
class	An additional class attribute for the HTML5 element

Value

An empty html5 td of the given class

html5.tangram *Convert a tangram class into an HTML5 string*

Description

Given a tangram class, a series of conversion creates an HTML5 representation of the table. It may be an HTML5 fragment or it may be a complete web page.

Usage

```
## S3 method for class 'tangram'
html5(object, id = NA, caption = NA, css = NA,
  fragment = TRUE, inline = NA, footnote = NA, ...)
```

Arguments

object	The cell table to render to HTML5
id	A unique identifier for the table (strongly recommended). If not provided, caption will be used.
caption	A string caption for the table
css	A string that is the href to the css for complete HTML5
fragment	A boolean flag that determines whether a fragment or a complete HTML5 document is generatedf
inline	A string containing a filename to include as inline CSS. It first searches the drive for the file, if that fails it looks inside the package for a matching css file.
footnote	Any footnotes to include under the table.
...	additional arguments to renderer. Unused

Details

The package includes several css files for styling. At present the following exist: 'hmisc.css', 'lancet.css', 'lancet-stripped.css' and 'nejm.css'

Value

A text string rendering of the given table in HTML5

---

index *Generate an index from a tangram or cell object*

Description

Given a tangram object create an index representation.

Usage

```
index(object, ...)
```

Arguments

object	The cell header to render to HTML5
...	additional arguments to renderer. Unused

Value

A matrix or list of strings containing key, source and value

index.cell_label *Generate an index from a label object*

Description

Overrides to generate no indexing on labels

Usage

```
## S3 method for class 'cell_label'
index(object, id = "tangram", key.len = 4, ...)
```

Arguments

object	cell; The cell for indexing
id	character; an additional specifier for the object key
key.len	numeric; length of key to generate
...	additional arguments to renderer. Unused

Value

A list of strings containing key, source and value

index.default *Generate an index from a cell object*

Description

Given a cell class create an index representation. If no source is specified no index will be generated.

Usage

```
## Default S3 method:
index(object, id = "tangram", name = NULL, key.len = 4,
  ...)
```

Arguments

object	cell; The cell for indexing
id	character; an additional specifier for the object key
name	character; optional names of elements inside object
key.len	numeric; length of generated key
...	additional arguments to renderer. Unused

Value

A list of strings containing key, source and value

`index.list` *Generate an index from a list object*

Description

Given a cell class create an index representation. If no source is specified no index will be generated.

Usage

```
## S3 method for class 'list'
index(object, id = "tangram", key.len = 4, ...)
```

Arguments

`object`	cell; The cell for indexing
`id`	character; an additional specifier for the object key
`key.len`	numeric; length of key to generate
`...`	additional arguments to renderer. Unused

Value

A list of strings containing key, source and value

`index.tangram` *Generate an an index from a tangram object*

Description

Given a tangram class create an index representation.

Usage

```
## S3 method for class 'tangram'
index(object, id = "tangram", key.len = 4, ...)
```

Arguments

`object`	The tangram for indexing
`id`	an additional specifier for the object key
`key.len`	numeric; length of keys generated (affects collision probability)
`...`	additional arguments to renderer. Unused

Value

A matrix of strings containing key, source and value

`is.binomial` *Determine if a vector is binomial or not*

Description

Determine if a vector is binomial or not

Usage

```
is.binomial(x, threshold = NA)
```

Arguments

`x`	Vector to determine type of
`threshold`	The upper threshold of unique values for which a vector is considered categorical.

Value

a Boolean: TRUE / FALSE

Examples

```
is.binomial(c(1,2,3))
is.binomial(factor(c("A","B","C")))
is.binomial(factor(c("A","B","B","A")))
is.binomial(factor(c(TRUE, FALSE, TRUE, FALSE)))
is.binomial(c('M', 'F', 'M', 'F'), 10)
```

`is.categorical` *Determine if a vector is categorical or not*

Description

Determine if a vector is categorical or not

Usage

```
is.categorical(x, threshold = NA)
```

Arguments

x	Vector to determine type of
threshold	The upper threshold of unique values for which a vector is considered categorical.

Value

A Boolean: TRUE / FALSE

Examples

```
is.categorical(c(1,2,3))
is.categorical(c(rep(1,20), rep(2, 20), rep(3, 20)), threshold=5)
is.categorical(c("A","B","B"))
is.categorical(factor(c("A","B","C")))
is.categorical(factor(c("A","B","B","A")))
is.categorical(factor(c(TRUE, FALSE, TRUE, FALSE)))
```

---

key *Key derivation helper function*

Description

This function should generate a string that uniquely identifies a piece of data present in a table. In a report with multiple tables the id is used to preserve uniqueness.

Usage

```
key(x, id)
```

Arguments

x	cell object to derive key for
id	the unique id of the table being keyed

Details

This function relies on the object being keyed having at a minimum character attributes for `row` and `col`. Additional specifies for embedded tables are given with `subrow` and `subcol`. The `row` and `col` are automatically appended when using a `table_builder`. However the `subrow` and `subcol` must be added by the user to a cell of a table.

latex *Render to LaTeX methods for tangram cell objects*

Description

Each of these methods will render the cell object as a LaTeX fragment

Usage

```
latex(object, ...)

## Default S3 method:
latex(object, ...)

## S3 method for class 'cell'
latex(object, ...)

## S3 method for class 'cell_label'
latex(object, ...)

## S3 method for class 'cell_n'
latex(object, ...)

## S3 method for class 'cell_header'
latex(object, ...)

## S3 method for class 'cell_subheader'
latex(object, ...)

## S3 method for class 'cell_iqr'
latex(object, ...)

## S3 method for class 'cell_estimate'
latex(object, ...)

## S3 method for class 'cell_fstat'
latex(object, ...)

## S3 method for class 'cell_fraction'
latex(object, ...)

## S3 method for class 'cell_chi2'
latex(object, ...)

## S3 method for class 'cell_studentt'
latex(object, ...)
```

```
## S3 method for class 'cell_spearman'
latex(object, ...)

## S3 method for class 'tangram'
latex(object, caption = "Table", footnote = NULL,
  fragment = TRUE, filename = NULL, append = FALSE, na.blank = TRUE,
  cgroup.just = NULL, arraystretch = 1.2, ...)
```

Arguments

object	object; the item to render to latex
...	additional arguments
caption	character; Caption to display on table
footnote	character; Footnote to include on table
fragment	logical; Is this a complete LaTeX document or just the table fragment
filename	character; filename to write LaTex into
append	logical; Should the write be an append operation or overwrite
na.blank	logical; Should NA's be displayed as blanks
cgroup.just	character; The text of the column justification used in the table
arraystretch	numeric; The arraystretch parameter used for vertical spacing

Value

the LaTeX rendering

Examples

```
latex(cell_label("123"))
latex(cell_iqr(rnorm(20)))
latex(cell_estimate(2.1,0.8, 3.3))
latex(cell_fraction(45, 137))
latex(table_builder()   %>%
        row_header("row") %>%
        col_header(1,2,3) %>%
        add_col("A","B","C"))
latex(tangram(drug~bili, pbc))
```

Parser

The parser class for generating abstract syntax trees for given table formulas.

Description

The parser class for generating abstract syntax trees for given table formulas.

Usage

```
Parser
```

Format

R6Class object.

Fields

`input` Storage for input string of a formula

`pos` The current parsing position

`len` The length of the input

Methods

`new()` Create a parser.

`expect(id)` Require the next token parsed to have the specified id and consume it.

`peek()` Return the next token parsed without consuming it.

`eat_whitespace()` Consume any spaces or tabs in input.

`next_token()` Return the next token parsed and consume it.

`format()` Parse a format class and return it's string.

`r_expression()` Parse an R expression class and return it's string.

`factor()` Parse a factor class and return it's AST Node.

`term()` Parse a term class and return it's AST Node.

`expression()` Parse an expression class and return it's AST Node.

`table_formula()` Parse a table formula class and return it's AST Node.

`run(input)` Run the parser on the given input, and return an AST

References

Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006) *Compilers: Principles, Techniques, and Tools*, 2nd edition. Addison Wesley.

Examples

```
Parser$new()$run("col1 + col2 + col3 ~ drug*age+spiders")
```

pbc *Mayo Clinic Primary Biliary Cirrhosis Data*

Description

D This data is from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984. A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval, met eligibility criteria for the randomized placebo controlled trial of the drug D-penicillamine. The first 312 cases in the data set participated in the randomized trial and contain largely complete data. The additional 112 cases did not participate in the clinical trial, but consented to have basic measurements recorded and to be followed for survival. Six of those cases were lost to follow-up shortly after diagnosis, so the data here are on an additional 106 cases as well as the 312 randomized participants.

Usage

```
pbc
```

Format

An object of class `data.frame` with 418 rows and 19 columns.

Details

A nearly identical data set found in appendix D of Fleming and Harrington; this version has fewer missing values.

Included for use in example from Hmisc.

print.cell *Print methods for tangram objects*

Description

Print methods for tangram objects

Usage

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

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

Arguments

x	object; the item to render
...	additional arguments passed to summary

Value

the text summary

Examples

```
print(cell_label("123"))
print(cell_iqr(rnorm(20)))
print(cell_estimate(2.1,0.8, 3.3))
print(cell_fraction(45, 137))
print(table_builder()   %>%
        row_header("row") %>%
        col_header(1,2,3) %>%
        add_col("A","B","C"))
print(tangram(drug~bili, pbc))
```

---

print.table_builder *Print a text summary of a given table_builder*

Description

Print a text summary of a given table_builder

Usage

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

Arguments

x	The table_builder to render to text
...	additional arguments, unused at present

Value

A text string rendering of the given table

render_f *Format a vector of provided numeric values*

Description

Given a vector of data return as strings formatted as requested

Usage

```
render_f(x, format)
```

Arguments

x	numeric; the data to format. Must work with quantile function.
format	numeric or character; If numeric preserve that many position past the decimal, if character pass directly into sprintf as format string

Value

character; formatted values as character strings

Examples

```
render_f(rnorm(5), 3)
render_f(round(rnorm(5), 2), "%010.03f")
```

rmd *Generate an Rmd table entry from a cell object*

Description

Given a cell object generate the corresponding piece of an Rmd table

Usage

```
rmd(object, ...)
```

Arguments

object	The cell_fstat for indexing
...	additional arguments to renderer. Unused

Value

A string representation of the table

rmd.table_builder *The default method for rendering tangram objects A tangram is a summary, so it returns itself. Otherwise convert to a text representation.*

Description

The default method for rendering tangram objects A tangram is a summary, so it returns itself. Otherwise convert to a text representation.

Usage

```
## S3 method for class 'table_builder'
rmd(object, ...)

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

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

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

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

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

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

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

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

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

## S3 method for class 'cell_chi2'
summary(object, ...)
```

Arguments

object	object; the item to render

...	additional arguments passed to summary

Value

the text summary

Examples

```
summary(cell_label("123"))
summary(cell_iqr(rnorm(20)))
summary(cell_estimate(2.1,0.8, 3.3))
summary(cell_fraction(45, 137))
summary(table_builder()   %>%
        row_header("row") %>%
        col_header(1,2,3) %>%
        add_col("A","B","C"))
summary(tangram(drug~bili, pbc))
```

---

rmd.tangram *Generate an Rmd table entry from a tangram object*

Description

Given a tangram object generate the corresponding piece of an Rmd table

Usage

```
## S3 method for class 'tangram'
rmd(object, ...)
```

Arguments

object	The cell_fstat for indexing
...	additional arguments to renderer. Unused

Value

A string representation of the table

rows *S3 object to return number of rows/cols in object*

Description

Number of rows/cols in provided object

Usage

```
rows(x)

cols(x)

## S3 method for class 'list'
rows(x)

## S3 method for class 'list'
cols(x)

## S3 method for class 'table_builder'
rows(x)

## S3 method for class 'table_builder'
cols(x)
```

Arguments

x	object; object to determine requested count

rtf *S3 rtf Method function for use on abstract table class*

Description

S3 rtf Method function for use on abstract table class

Usage

```
rtf(object, id, ...)
```

Arguments

object	The cell to render to RTF
id	A unique identifier for the table (strongly recommended). If not provided, caption will be used.
...	additional arguments to renderer. Unused at present.

Value

A text string rendering of the given table

`rtf.cell` *Given a cell class create an RTF representation.*

Description

Given a cell class create an RTF representation.

Usage

```
## S3 method for class 'cell'
rtf(object, id, ...)
```

Arguments

`object`	The cell to render to RTF
`id`	A unique identifier for traceability
`...`	additional arguments to renderer. Unused

Value

An RTF string rendering of the given cell.

`rtf.cell_fstat` *Convert an abstract cell_fstat object into an RTF string*

Description

Given a cell_fstat class create an RTF representation.

Usage

```
## S3 method for class 'cell_fstat'
rtf(object, id, ...)
```

Arguments

`object`	The cell fstat to render to RTF
`id`	A unique identifier for traceability
`...`	additional arguments to renderer. Unused

Value

A text string rendering of the given fstat as a <td> with several 's.

rtf.cell_header *Convert an abstract cell_header object into an RTF string*

Description

Given a cell_header class create an RTF representation.

Usage

```
## S3 method for class 'cell_header'
rtf(object, id, ...)
```

Arguments

object	The cell header to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused

Value

An RTF string rendering of the given header

rtf.cell_iqr *Convert an abstract cell_iqr object into an RTF string*

Description

Given a cell_quantile class create an RTF representation.

Usage

```
## S3 method for class 'cell_iqr'
rtf(object, id, ..., point = 9)
```

Arguments

object	The cell quantile to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
point	numeric; The font point size to use in display

Value

An RTF string rendering of the given quantile.

rtf.cell_label *Given a cell_label class create an RTF representation.*

Description

Given a cell_label class create an RTF representation.

Usage

```
## S3 method for class 'cell_label'
rtf(object, id, ..., point = 18)
```

Arguments

object	The cell label to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
point	size of main font for cell label

Value

An RTF text string rendering of the given label.

rtf.cell_n *Convert an abstract cell_n object into an RTF string*

Description

Given a cell_n class create an RTF representation.

Usage

```
## S3 method for class 'cell_n'
rtf(object, id, ...)
```

Arguments

object	The cell n to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused

Value

An RTF string rendering of the given n.

rtf.cell_subheader *Convert an abstract cell_subheader object into an RTF string*

Description

Given a cell_subheader class create an RTF representation.

Usage

```
## S3 method for class 'cell_subheader'
rtf(object, id, ..., point = 9)
```

Arguments

object	The cell header to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused
point	numeric; The font point size to use in display

Value

An RTF string rendering of the given header

rtf.default *Default conversion to RTF for an abstract table element*

Description

Gives a warning and produces an empty cell

Usage

```
## Default S3 method:
rtf(object, id, ...)
```

Arguments

object	The cell to render to RTF
id	A unique identifier for traceability
...	additional arguments to renderer. Unused

Value

A RTF string rendering of the given cell

rtf.tangram *Convert a tangram into an RTF string or file*

Description

Given a tangram class, a series of conversion creates an rtf representation of the table.

Usage

```
## S3 method for class 'tangram'
rtf(object, id = NA, caption = NA, fragment = FALSE,
  widths = NA, footnote = NA, filename = NA, append = FALSE,
  point = 9, ...)
```

Arguments

object	The cell table to render to RTF
id	A unique identifier for the table (strongly recommended).
caption	A string caption for the table
fragment	A boolean flag that determines whether a fragment or a complete RTF document is generatedf
widths	RTF requires specified left margin and column widths, this allows user control over these (inches)
footnote	Any footnotes to include under the table.
filename	A filename to write resulting rtf file to
append	A boolean for whether or not to append to given filename
point	Main font point size
...	additional arguments Fto renderer. Unused

Value

A text string rendering of the given table

summarize_chisq *Create a summarization for a categorical row versus a categorical column*

Description

Given a row and column object from the parser apply a chi^2 test and output the results

Usage

```
summarize_chisq(table, row, column, pformat = NULL, collapse_single = TRUE,
  overall = NULL, ...)
```

Arguments

table	The table object to modify
row	The row variable object to use (categorical)
column	The column variable to use (categorical)
pformat	numeric or character; A formatting directive to be applied to p-values
collapse_single	logical; default TRUE. Categorical variables with a two values collapse to single row.
overall	logical; Include the overall summary column
...	absorbs extra parameters. Currently unused.

Value

The modified table object

---

summarize_kruskal_horz

Create a summarization for a categorical set of column versus a numerical row

---

Description

Given a row and column object from the parser apply a Kruskal test and output the results horizontally. 1 X (n + no. categories + test statistic)

Usage

```
summarize_kruskal_horz(table, row, column, pformat = NULL, msd = FALSE,
  quant = c(0.25, 0.5, 0.75), overall = NULL, ...)
```

Arguments

table	The table object to modify
row	The row variable object to use (numerical)
column	The column variable to use (categorical)
pformat	numeric or character; A formatting directive to be applied to p-values
msd	logical; Include mean and standard deviation with quantile statistics
quant	numeric; Vector of quantiles to include. Should be an odd number since the middle value is highlighted on display.
overall	logical; Include overall summary statistics for a categorical column
...	absorbs additional arugments. Unused at present.

Value

The modified table object

`summarize_kruskal_vert` *Create a summarization for a categorical row versus a numerical column*

Description

Given a row and column object from the parser apply a Kruskal test and output the results vertically (#Categories+1) X (N, Summary, Statistic)

Usage

```
summarize_kruskal_vert(table, row, column, pformat = NULL, ...)
```

Arguments

`table`	The table object to modify
`row`	The row variable object to use (categorical)
`column`	The column variable to use (numerical)
`pformat`	numeric or character; A formatting directive to be applied to p-values
`...`	absorbs additional arugments. Unused at present.

Value

The modified table object

`summarize_spearman` *Create a summarization for a numerical row versus a numerical column*

Description

Given a row and column object from the parser apply a Spearman test and output the results in a 1X3 format.

Usage

```
summarize_spearman(table, row, column, pformat = NULL, ...)
```

Arguments

table	The table object to modify
row	The row variable object to use (numerical)
column	The column variable to use (numerical)
pformat	numeric or character; A formatting directive to be applied to p-values
...	absorbs additional arguments. Unused at present.

Value

The modified table object

---

table_builder *Table Construction Toolset*

Description

These functions help build a table. A table can be embedded inside another table as a cell as well. The typical transform functions that provide bundles of functionality utilize this approach and each row column pair are rendered as a cell that is a table and later the whole table is flattened.

Usage

```
table_builder(row = NA, column = NA, embedded = FALSE)

col_header(table_builder, ..., sub = TRUE)

row_header(table_builder, ..., sub = TRUE)

write_cell(table_builder, x, ...)

home(table_builder)

cursor_up(table_builder, n = 1)

cursor_down(table_builder, n = 1)

cursor_left(table_builder, n = 1)

cursor_right(table_builder, n = 1)

cursor_pos(table_builder, nrow, ncol)

carriage_return(table_builder)

line_feed(table_builder, n = 1)
```

```
new_line(table_builder)

new_row(table_builder)

new_col(table_builder)

table_builder_apply(table_builder, X, FUN, ...)

add_col(table_builder, ...)

add_row(table_builder, ...)
```

Arguments

row	character; Value to use for indexing
column	character; Value to use for indexing
embedded	logical; is this to be embedded in another table
table_builder	The table builder object to modify
...	object; the elements to add or additional values to pass to FUN
sub	logical; treat as subheader if after first header, defaults to TRUE
x	any; a value to use for a cell in operation
n	integer; Number of positions to move cursor, defaults to 1
nrow	integer; specifies desired row
ncol	integer; specifies desired col
X	list or vector; items to iterate over
FUN	the function to use in iteration

Details

This library is designed to use a core `table_builder` object that is passed from function to function using the pipe `%>%` operator. First create a `table_builder` using the `table_builder()` function and use the operators to build out the table. The row and column given to the `table_builder` are what is used in later construction of an index key. The table_builder object contains an item table which is the current table being built.

Column and row headers are attached as attributes to each table constructed are are tables in their own right that should match the proper dimension of the contained table. When later flattening a table of embedded tables, only the left and top most headers are used.

The table builder also has a cursor which maintains the state of where cell items are being written in table construction. It is possible to move the cursor into undefined portions of the table. Therefore it is best to use cursor movement to move in defined rows or columns of information.

Value

the modified table_builder

Examples

```
library(magrittr)
table_builder()                          %>%
col_header("One","Two","Three","Four") %>%
row_header("A",   "B",   "C")           %>%
write_cell("A1")                         %>%
cursor_right()                           %>%
add_col("A2", "A3")                      %>%
home()                                   %>%
new_line()                               %>%
table_builder_apply(1:3, FUN=function(tb, x) {
  tb %>% write_cell(paste0("B",x)) %>% cursor_right()
})                                       %>%
new_col()                                %>%
add_row(paste0(c("A","B","C"), 4))       %>%
cursor_up(2)                             %>%
line_feed()                              %>%
cursor_left(3)                           %>%
add_col(paste0("C", 1:4))
```

`table_flatten` *Given a tangram object with embedded tables, flattens to a single table.*

Description

Flattening function to expanded embedded tables inside table cells.

Usage

```
table_flatten(table)
```

Arguments

`table`	the table object to flatten

Value

the flattened table object

tangram *Table creation methods*

Description

The tangram method is the principal method to create tables. It uses R3 method dispatch. If one specifies rows and columns, one gets an empty table of the given size. A formula or character will invoke the parser and process the specified data into a table like `Hmisc::summaryM`. Given an `rms` object it will summarize that model in a table. A `data.frame` is converted directly into a table as well for later rendering. Can create tables from summary.rms(), anova.rms(), and other rms object info to create a single pretty table of model results. The rms and Hmisc packages are required.

Usage

```
tangram(x, ...)

## S3 method for class 'numeric'
tangram(x, cols, embedded = FALSE, ...)

## S3 method for class 'data.frame'
tangram(x, colheader = NA, ...)

## S3 method for class 'formula'
tangram(x, data, transforms = hmisc_style, after = NA,
  digits = NA, ...)

## S3 method for class 'character'
tangram(x, data, transforms = hmisc_style, after = NA,
  digits = NA, ...)

## S3 method for class 'rms'
tangram(x, data = NULL, short.labels = NULL,
  footnote = NULL, rnd.digits = 2, rnd.stats = rnd.digits, ...)
```

Arguments

`x`	object; depends on S3 type, could be rows, formula, string of a formula, data.frame or numerical rows, an rms.model
`...`	addition models or data supplied to table construction routines
`cols`	numeric; An integer of the number of cols to create
`embedded`	logical; Will this table be embedded inside another
`colheader`	character; Use as column headers in final table
`data`	data.frame; data to use for rendering tangram object
`transforms`	list of lists of functions; that contain the transformation to apply for summarization

after	function or list of functions; one or more functions to further process an abstract table
digits	numeric; default number of digits to use for display of numerics
short.labels	numeric; Named vector of variable labels to replace in interaction rows. Must be in format c("variable name" = "shortened label").
footnote	character; A string to add to the table as a footnote.
rnd.digits	numeric; Digits to round reference, comparison, result and CI values to. Defaults to 2.
rnd.stats	numeric; Digits to round model LR, R2, etc to. Defaults to rnd.digits.

Value

A tangram object (a table).

Examples

```
tangram(1, 1)
tangram(data.frame(x=1:3, y=c('a','b','c')))
tangram(drug ~ bili + albumin + protime + sex + age + spiders, pbc)
tangram("drug ~ bili + albumin + stage::Categorical + protime + sex + age + spiders", pbc)
```

Token *A token in the formula grammar*

Description

A token in the formula grammar

Usage

```
Token
```

Format

R6Class object.

Fields

`id` The token identifier, E.g. "LPAREN"

`name` Information about the token, useful with IDENTIFIERs.

Methods

`new(id, name="")` Create a new token in the grammar.

Index