

Package ‘tangram’

July 22, 2025

Title The Grammar of Tables

Version 0.8.2

Description Provides an extensible formula system to quickly and easily create production quality tables. The processing steps are a formula parser, statistical content generation from data as defined by formula, followed by rendering into a table. Each step of the processing is separate and user definable thus creating a set of composable building blocks for highly customizable 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 in a resulting table. 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],
Ahra Kim [ctb]

Maintainer Shawn Garbett <Shawn.Garbett@vumc.org>

Depends R (>= 3.5.0), R6, magrittr, knitr

License GPL-3

Encoding UTF-8

LazyData true

Suggests testthat, rms, rmarkdown, Hmisc, sandwich, dplyr, Matching, epitools

Imports stringi, stringr, base64enc, digest, htmltools

RoxygenNote 7.2.3

Collate 'compile-cell.R' 'cell-hmisc.R' 'cell-lancet.R' 'cell-nejm.R'
'compile-clmm2.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' 'iify.R' 'render-csv.R'
 'render-html5.R' 'render-index.R' 'render-latex-map.R'
 'render-latex.R' 'render-rmd.R' 'render-rtf.R'
 'render-summary.R' 'smd.R' 'transform-hmisc.R'
 'transform-lancet.R' 'transform-nejm.R' 'transform-proc-tab.R'
 'transform-smd.R'

NeedsCompilation no

URL <https://github.com/spgarbet/tangram>

BugReports <https://github.com/spgarbet/tangram/issues>

Repository CRAN

Date/Publication 2023-01-26 10:20:02 UTC

Contents

+.tangram	4
add_footnote	5
add_indent	5
args_flatten	6
as.categorical	6
ASTBranch	7
ASTFunction	8
ASTMultiply	10
ASTNode	11
ASTPlus	12
ASTTableFormula	14
ASTVariable	15
cbind.tangram	17
cell	17
cell_header	18
cell_label	18
cell_n	19
cell_subheader	20
cell_transform	20
col_header	21
csv	22
custom_css	23
del_col	24
del_row	24
derive_label	25
drop_statistics	25
format_guess	26
hmisc_data_type	26
hmisc_intercept_cleanup	27
hmisc_p	27

html5	30
html5.cell	31
html5.cell_header	31
html5.cell_label	32
html5.cell_n	32
html5.cell_subheader	33
html5.character	33
html5.default	34
html5.logical	34
html5.tangram	35
index	36
index.cell_label	36
index.default	37
index.list	37
index.tangram	38
insert_column	38
insert_row	39
is.binomial	39
is.categorical	40
key	41
lancet	41
lancet_cell	42
lancet_fraction	42
lancet_mean_sd	43
latex	44
latexify	45
latex_template	46
nejm	46
nejm_cell	47
nejm_fraction	47
nejm_iqr	48
nejm_range	49
Parser	49
pbcr	52
pipe.tangram	52
print.cell	53
proc_tab	54
rbind.tangram	54
render_f	55
render_route_tangram	55
replace_cell	56
rmd	56
rows	57
rtf	58
rtf.cell	58
rtf.cell_chi2	59
rtf.cell_fstat	59
rtf.cell_header	60

rtf.cell_iqr	60
rtf.cell_label	61
rtf.cell_n	61
rtf.cell_subheader	62
rtf.default	62
rtf.tangram	63
select_col	64
select_row	64
smd	65
smd_cell	65
smd_compare	66
smd_contingency	67
smd_dist	68
smd_fraction	68
smd_meansd	69
standard_difference	69
summarize_kruskal_horz	70
summarize_nejm_horz	73
summarize_nejm_vert	74
summary.tangram	75
table_flatten	75
tangram.clmm2	76
Token	80

Index	82
--------------	-----------

<code>+. tangram</code>	<i>Provide a "+" operator for rbind of tangram tables</i>
-------------------------	---

Description

The plus operator provides an rbind for tangram tables

Usage

```
## S3 method for class 'tangram'  
x + y
```

Arguments

- x left argument for rbind
- y right argument for rbind

Value

A row wise merged tangram object

add_footnote	<i>Add a footnote to a table</i>
--------------	----------------------------------

Description

Add a footnote to a table

Usage

```
add_footnote(table, footnote)
```

Arguments

table	tangram; the tangram table to modify
footnote	character; The footnote to add

Value

the modified table

add_indent	<i>Add indentations to left column row headers</i>
------------	--

Description

Add indentations to left column row headers. Note: will only work on cell_header cells.

Usage

```
add_indent(table, amounts = 2, rows = NULL, columns = NULL)
```

Arguments

table	Output of tangram::tangram()
amounts	numeric; Specifies number of spaces to add. A vector that is either a single value or vector of the same size as the height of the table. If positions is specified then it must be the same length. Defaults to 2, which each pair of spaces converts naturally in rendering to HTML, LaTeX, etc..
rows	numeric; A vector of numeric row numbers for the rows that need to be indented. Defaults to NULL which indents all.
columns	numeric; Column to apply indent to, defaults to 1

Value

the modified table

Examples

```
x <- tangram(drug ~ bili + albumin, pbc)
add_indent(x)
add_indent(x, amounts=10)
add_indent(x, amounts=c(0, 0, 2, 4))
add_indent(x, rows=c(3))
add_indent(x, rows=c(3, 4), amounts=c(4, 2))
```

args_flatten	<i>Flatten variable arguments</i>
--------------	-----------------------------------

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	<i>Convert data type to a factor if it's not already</i>
----------------	--

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 Syntax 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 Syntax 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.

A left/right branch in an Abstract Syntax 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.

Format

`R6Class` object.

Super class

`tangram::ASTNode` -> ASTBranch

Public fields

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

Methods**Public methods:**

- `ASTBranch$distribute()`
- `ASTBranch$reduce()`
- `ASTBranch$clone()`

Method `distribute()`: Call to distribute multiplication nodes, just recursively calls left and right node distribute functions

Usage:

```
ASTBranch$distribute()
```

Method `reduce()`: Attached data to nodes by processing data.frame appropriately. Recursively calls left and right nodes to reduces on data.frame

Usage:
ASTBranch\$reduce(df)
Arguments:
df (data.frame) Data frame to reduce over

Method clone(): The objects of this class are cloneable with this method.
Usage:
ASTBranch\$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

ASTFunction	<i>A specified function call as an ASTNode</i>
-------------	--

Description

A specified function call as an ASTNode
A specified function call as an ASTNode

Format

R6Class object.

Super class

tangram::ASTNode -> ASTFunction

Public fields

r_expr A string containing the raw r expression from inside the parenthesis
data Data stored as a result of reduction

Methods

- Public methods:**
- ASTFunction\$new()
 - ASTFunction\$factors()
 - ASTFunction\$name()
 - ASTFunction\$string()
 - ASTFunction\$reduce()
 - ASTFunction\$clone()

Method new(): Construct a node representing a function call
Usage:

```
ASTFunction$new(value, r_expr)
```

Arguments:

value (character) The name of the function call

r_expr Any r expression to be evaluated inside the call

Method factors(): Returns all terminal nodes, this is a terminal node so returns self

Usage:

```
ASTFunction$factors()
```

Method name(): Returns the function call as character

Usage:

```
ASTFunction$name()
```

Method string(): Returns a re-parsable representation of the node

Usage:

```
ASTFunction$string()
```

Method reduce(): Given a data.frame execute the function in that environment and associate the result as data.

Usage:

```
ASTFunction$reduce(data)
```

Arguments:

data (data.frame) The data.frame to use as the environment for the function execution

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ASTFunction$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

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.

The multiplication of two terms, as an ASTNode.

Format

`R6Class` object.

Super classes

`tangram::ASTNode` -> `tangram::ASTBranch` -> `ASTMultiply`

Public fields

`left` The AST tree to the left.

`right` The AST tree to the right.

`type` The specified type of this node

Methods**Public methods:**

- `ASTMultiply$new()`
- `ASTMultiply$distribute()`
- `ASTMultiply$factors()`
- `ASTMultiply$string()`
- `ASTMultiply$clone()`

Method `new()`: Construct a multiplication node

Usage:

`ASTMultiply$new(left, right)`

Arguments:

`left` (ASTNode) nodes to the left of the multiplication

`right` (ASTNode) nodes to the right of the multiplication

Method `distribute()`: Rearrange nodes distribution multiplication across parenthesis

Usage:

`ASTMultiply$distribute()`

Method `factors()`: return all terminal nodes on left and right

Usage:

ASTMultiply\$factores()

Method string(): Return a re-parseable string

Usage:

ASTMultiply\$string()

Method clone(): The objects of this class are cloneable with this method.

Usage:

ASTMultiply\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Examples

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

ASTNode

A Node in an Abstract Syntax Tree (AST)

Description

A Node in an Abstract Syntax Tree (AST)

A Node in an Abstract Syntax Tree (AST)

Details

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.

Public fields

format Any formatting directive passed to this node.

value A string of additional information contained by the node.

Methods

Public methods:

- [ASTNode\\$terms\(\)](#)
- [ASTNode\\$distribute\(\)](#)
- [ASTNode\\$string\(\)](#)
- [ASTNode\\$reduce\(\)](#)
- [ASTNode\\$set_format\(\)](#)
- [ASTNode\\$clone\(\)](#)

Method `terms()`: Returns this node

Usage:

`ASTNode$terms()`

Method `distribute()`: Distributes data across multiplications and rearranges nodes

Usage:

`ASTNode$distribute()`

Method `string()`: Returns string representation of node

Usage:

`ASTNode$string()`

Method `reduce()`: Given a set of data, associates it with AST nodes

Usage:

`ASTNode$reduce(data)`

Arguments:

`data` (data.frame) data to associate across nodes

Method `set_format()`: Override the formatting directive for this node

Usage:

`ASTNode$set_format(x)`

Arguments:

`x` (numeric,character) the formatting directive

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`ASTNode$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

ASTPlus

The addition of two terms, in an ASTNode.

Description

The addition of two terms, in an ASTNode.

The addition of two terms, in an ASTNode.

Format

[R6Class](#) object.

Super classes

`tangram::ASTNode` -> `tangram::ASTBranch` -> `ASTPlus`

Public fields

`data` Just returns the R6 name 'ASTPlus'

`left` The node to the left of this node

`right` The node to the right of this node

Methods**Public methods:**

- `ASTPlus$new()`
- `ASTPlus$terms()`
- `ASTPlus$string()`
- `ASTPlus$clone()`

Method `new()`: Construct a new node that represents addition

Usage:

`ASTPlus$new(left, right)`

Arguments:

`left` (`ASTNode`) Node on the left side of the addition

`right` (`ASTNode`) Node on the right side of the addition

Method `terms()`: Returns a vector of the left and right terms

Usage:

`ASTPlus$terms()`

Method `string()`: A parsable string representation of this node.

Usage:

`ASTPlus$string()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`ASTPlus$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Examples

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

ASTTableFormula	<i>The root ASTNode of a formula.</i>
-----------------	---------------------------------------

Description

The root ASTNode of a formula.

The root ASTNode of a formula.

Format

[R6Class](#) object.

Super classes

[tangram::ASTNode](#) -> [tangram::ASTBranch](#) -> [ASTTableFormula](#)

Public fields

`left` The AST tree for the columns.

`right` The AST tree for the rows.

Methods**Public methods:**

- [ASTTableFormula\\$new\(\)](#)
- [ASTTableFormula\\$terms\(\)](#)
- [ASTTableFormula\\$string\(\)](#)
- [ASTTableFormula\\$clone\(\)](#)

Method `new()`: Create a new formula node

Usage:

`ASTTableFormula$new(left, right)`

Arguments:

`left` The left side of the "~" as an AST

`right` The right side of the "~" as an AST

Method `terms()`: Returns all terminal nodes from left and right

Usage:

`ASTTableFormula$terms()`

Method `string()`: A re-parseable string representing the AST

Usage:

`ASTTableFormula$string()`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ASTTableFormula$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

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

 ASTVariable

A Variable in an Abstract Syntax Tree (AST)

Description

A Variable in an Abstract Syntax Tree (AST)

A Variable in an Abstract Syntax Tree (AST)

Format

[R6Class](#) object.

Details

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](#).

Methods

`new(identifier, format=NA, type=NA)`

`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.

Super class

[tangram::ASTNode](#) -> ASTVariable

Public fields

`data` The associated data post reduction

`type` The identified type of this node (defaults: Categorical, Numeric)

Methods

Public methods:

- `ASTVariable$new()`
- `ASTVariable$factors()`
- `ASTVariable$name()`
- `ASTVariable$string()`
- `ASTVariable$reduce()`
- `ASTVariable$clone()`

Method `new()`: 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.

Usage:

```
ASTVariable$new(identifier, format = NA, type = NA)
```

Arguments:

`identifier` (character) Variable name
`format` (character, numeric) Formatting directive
`type` (character) any additional type information

Method `factors()`: Returns all terminal nodes under this. Since this is a terminal node, returns self

Usage:

```
ASTVariable$factors()
```

Method `name()`: Returns the text name of this node. For an intercept, returns "All"

Usage:

```
ASTVariable$name()
```

Method `string()`: Returns name of variable with optional format and type information

Usage:

```
ASTVariable$string()
```

Method `reduce()`: Given a data.frame, associates correct variable with this node

Usage:

```
ASTVariable$reduce(d)
```

Arguments:

`d` (data.frame) data.frame to reduce

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ASTVariable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

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

cbind.tangram	<i>A cbind for generated table tangram objects.</i>
---------------	---

Description

Execute the equivalent of an cbind for generated tables

Usage

```
## S3 method for class 'tangram'
cbind(..., deparse.level = 1)
```

Arguments

...	tangram objects to cbind
deparse.level	numeric; not used

Value

A merged tangram object

cell	<i>Construct a table cell from an object</i>
------	--

Description

Any R object can be used as a cell value. Attributes are used to store additional classes 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 delineate the key value of a cell for key generation

Value

The modified R object

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_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	<i>Create an cell_n (S3) object of the given statistic</i>
--------	--

Description

A cell_n object contains an n value. Essentially, this is just a helper that appends the cell_n class to the given object and makes sure it's a cell S3 object as well.

Usage

```
cell_n(n, class = NULL, hdr = FALSE, possible = NULL, ...)
```

Arguments

n	The numerical value
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
hdr	logical; Construct an n value for a header (defaults to FALSE)
possible	numerical; The total N that was possible
...	optional extra information to attach

Value

A cell_n object.

Examples

```
cell_n(20)
```

cell_subheader	<i>Create a cell_subheader object of the given text.</i>
----------------	--

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	<i>Create a function to transform all cells of a table</i>
----------------	--

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

col_header	<i>A set of magrittr operators for tangram tables</i>
------------	---

Description

A set of magrittr operators for tangram tables

Usage

```
col_header(table, ..., sub = TRUE)
```

```
row_header(table, ..., sub = TRUE)
```

```
write_cell(table, x, ...)
```

```
home(table)
```

```
cursor_up(table, n = 1)
```

```
cursor_down(table, n = 1)
```

```
cursor_left(table, n = 1)
```

```
cursor_right(table, n = 1)
```

```
cursor_pos(table, nrow, ncol)
```

```
carriage_return(table)
```

```
line_feed(table, n = 1)
```

```
new_line(table)
```

```
new_row(table)
```

```
new_col(table)
```

```
table_apply(table, x, FUN, ...)
```

```
add_col(table, ...)
```

```
add_row(table, ...)
```

```
set_footnote(table, footnote)
```

```
set_id(table, id)
```

```
set_caption(table, caption)
```

```
set_style(table, style)
```

```
set_colspan(table, span)
```

```
set_rowspan(table, span)
```

Arguments

table	tangram; The tangram table being built
...	additional argument passed
sub	logical; Is this a subheader
x	object of focus in operation
n	numeric; number of times to perform operation
nrow	numeric; number of rows
ncol	numeric; number of columns
FUN	function; function to apply
footnote	character; footnote to add
id	character; id of table
caption	character; caption of table
style	character; styling in compiling table and in rendering
span	numeric; number of rows or columns to span

csv

Generate an csv from a tangram or cell object

Description

Given a tangram object create an index representation.

Usage

```
csv(object, ...)
```

```
## S3 method for class 'tangram'
csv(object, file = NULL, sep = ",", ...)
```

```
## Default S3 method:
csv(object, ...)
```

Arguments

object	The cell header to render to HTML5
...	additional arguments to renderer. Unused
file	File to write result into
sep	separator to use

Value

A string containing the csv file

custom_css	<i>Return a CSS file as a string</i>
------------	--------------------------------------

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 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	<i>Delete given column(s) from a table</i>
---------	--

Description

Given a table, remove the specified column

Usage

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

Arguments

table	the table to modify
col	vector containing column(s) to drop

Value

the modified table

del_row	<i>Delete a row(s) from a table</i>
---------	-------------------------------------

Description

Given a table, remove the specified row

Usage

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

Arguments

table	the table to modify
row	vector with row numbers to drop

Value

the modified table

derive_label	<i>Derive label of AST node.</i>
--------------	----------------------------------

Description

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

Usage

```
derive_label(node, capture_units = FALSE, ...)
```

Arguments

- node Abstract syntax tree node.
- capture_units logical; Capture units from parenthesis ending a label
- ... Other arguments, ignored

Value

A string with a label for the node

drop_statistics	<i>Drop all statistics columns from a table.</i>
-----------------	--

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	<i>Guess the best format for a given set of numerical data</i>
--------------	--

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	<i>Determine data type of a vector loosely consistent with Hmisc.</i>
-----------------	---

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.

See Also

[hmisc](#)

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	<i>Cleanup an intercept only model</i>
-------------------------	--

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_p	<i>Cell Generation functions for hmisc default</i>
---------	--

Description

Each function here is called when a cell is generated. Overriding these in a formula call will allows one to customize exactly how each cell's contents are generated. While this serves as the base template for transforms, it is by no means required if one develops their own bundle of data transforms. One can create ay number of cell level styling choices.

Usage

```

hmisc_p(p, pformat = "%1.3f", include_p = TRUE)

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

hmisc_fraction(numerator, denominator, format = 3, ...)

hmisc_fstat(f, df1, df2, p, class = NULL, ...)

hmisc_chi2(chi2, df, p, class = NULL, ...)

hmisc_spearman(S, rho, p, class = NULL, ...)

hmisc_wilcox(V, p, class = NULL, ...)

hmisc_cell

```

Arguments

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

denominator	numeric; The value of the denominator
f	The value of the f-statistic
df1	1st dimension degrees of freedom
df2	2nd dimension degrees of freedom
class	character; An optional field for additional S3 classes (e.g. could be used in html rendering for CSS)
chi2	The value of the X^2 statistic
df	degrees of freedom
S	The value of the spearman statistic
rho	The rho value of the test
V	The value of the Wilcoxon statistic

Format

An object of class `list` of length 8.

Value

A formatted string or cell as appropriate

hmisc_p

Given a style in number of digits or a `sprintf` style specifier it renders the p-value and checks to see if it's all zeros, then switches the output to a less than.

hmisc_iqr

Construct a cell which has the interquartile ranges specified.

hmisc_fraction

Construct a cell which has the fraction specified in an hmisc format

hmisc_fstat

Construct a cell which has the fstat specified in an hmisc format.

hmisc_chi2

Construct a cell which has the χ^2 specified in an hmisc format

hmisc_spearman

Construct a cell which has the spearman specified in an hmisc format

hmisc_wilcox

Construct a cell which has the Wilcoxon specified in an hmisc format

hmisc_cell

List of data transforms for a cell of a table.

```
hmisc_cell <- list(
  n      = cell_n,
  iqr    = hmisc_iqr,
  fraction = hmisc_fraction,
  fstat  = hmisc_fstat,
  chi2   = hmisc_chi2,
  spearman = hmisc_spearman,
  wilcox = hmisc_wilcox,
  p      = hmisc_p
)
```

See Also

[hmisc](#)

Examples

```
hmisc_p(1e-6)
hmisc_p(0.234)
hmisc_p(1.234e-6, 5)
hmisc_p(1.234e-6, 6)
require(stats)
hmisc_iqr(rnorm(100), '3')
hmisc_fraction(1, 4, 3)
hmisc_fstat(4.0, 10, 20, 0.004039541)
hmisc_chi2(5.33, 6, 0.2)
hmisc_spearman(20, 0.2, 0.05)
hmisc_wilcox(20, 0.2)
```

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	<i>Convert an abstract cell object into an HTML5 string</i>
------------	---

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_header	<i>Convert an abstract cell_header object into an HTML5 string</i>
-------------------	--

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_label	<i>Convert a cell_label object into an HTML5 string</i>
------------------	---

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	<i>Convert an abstract cell_n object into an HTML5 string</i>
--------------	---

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	<i>Convert an abstract cell_subheader object into an HTML5 string</i>
----------------------	---

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	<i>Default conversion to HTML5 for a character cell</i>
-----------------	---

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	<i>Default conversion to HTML5 for an abstract table element</i>
---------------	--

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.logical	<i>Default conversion to HTML5 for a logical cell</i>
---------------	---

Description

Produces table cell or nothing if it's an NA. This is useful for dealing with rowspan and colspan.

Usage

```
## S3 method for class 'logical'
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	<i>Convert a tangram class into an HTML5 string</i>
---------------	---

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 = NULL,
  caption = NULL,
  fragment = NULL,
  style = NULL,
  footnote = NULL,
  inline = NULL,
  fixed_thead = NULL,
  ...
)
```

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
fragment	A boolean flag that determines whether a fragment or a complete HTML5 document is generated
style	A string containing a style 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.
inline	DEPRECATED
fixed_thead	logical; fixes the header using position sticky in CSS defaults to FALSE
...	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	<i>Generate an index from a tangram or cell object</i>
-------	--

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	<i>Generate an index from a label object</i>
------------------	--

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	<i>Generate an index from a cell object</i>
---------------	---

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	<i>Generate an index from a list object</i>
------------	---

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	<i>Generate an an index from a tangram object</i>
---------------	---

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

insert_column	<i>Insert a column into a tangram table</i>
---------------	---

Description

Insert a column into a tangram table. Will fill with empty cells is not enough cells are specified.

Usage

```
insert_column(table, after, ..., class = NULL)
```

Arguments

table	the table to modify
after	numeric; The column to position the new row after. Can be zero for inserting a new first row.
...	Table cells to insert. Cannot be larger than existing table.
class	character; Classes to apply as directives to renderers

Value

the modified table

insert_row	<i>Insert a row into a tangram table</i>
------------	--

Description

Insert a row into a tangram table. Will fill with empty cells is not enough cells are specified.

Usage

```
insert_row(table, after, ..., class = NULL)
```

Arguments

table	the table to modify
after	numeric; The row to position the new row after. Can be zero for inserting a new first row.
...	Table cells to insert. Cannot be larger than existing table.
class	character; Classes to apply as directives to renderers

Value

the modified table

is.binomial	<i>Determine if a vector is binomial or not</i>
-------------	---

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	<i>Determine if a vector is categorical or not</i>
----------------	--

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	<i>Key derivation helper function</i>
-----	---------------------------------------

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.

lancet	<i>Style Bundle for Lancet style</i>
--------	--------------------------------------

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

lancet

Format

An object of class list of length 5.

lancet_cell	<i>Cell Generation functions for Lancet styling</i>
-------------	---

Description

Each function here is called when a cell is generated. Overriding these in a formula call will allows one to customize exactly how each cell’s contents are generated.

Usage

lancet_cell

Format

An object of class list of length 8.

lancet_fraction	<i>Create an cell_fraction (S3) in NEJM style of the given data</i>
-----------------	---

Description

A cell object contains a statistical result of a fraction/percentage in nejm style

Usage

lancet_fraction(numerator, denominator, format = NULL, ...)

Arguments

- numerator numeric; The value of the numerator
- denominator numeric; The value of the denominator
- format numeric or character; a string formatting directive
- ... optional extra information to attach

Value

A cell_fraction object.

Examples

lancet_fraction(1, 4, 3)

lancet_mean_sd	Create a mean/sd cell object of the given data in Lancet style
----------------	--

Description

Create a mean/sd cell object of the given data in Lancet style.

Usage

```
lancet_mean_sd(  
  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; ignored. For compatibility with hmisc_iqr
type	integer; ignored. For compatibility with hmisc_iqr
msd	logical; ignored. For compatibility with hmisc_iqr
quant	numeric; ignored. For compatibility with hmisc_iqr
...	additional arguments to constructing cell

Value

A cell object.

Examples

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

 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, na.blank = TRUE, ...)

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

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

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

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

## S3 method for class 'tangram'
latex(object, fragment = TRUE, filename = NULL, append = FALSE, ...)
```

Arguments

object	object; the item to render to latex
...	additional arguments
na.blank	logical; Display NAs as blanks.
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

Details

There are addition arguments possible to control the rendering, but due to some oddities between CRAN requirements and how R handles defaults (for full details see the source code) they are as follows

- * cgroup.just character; The text of the column justification used in the table
- * arraystretch numeric; The arraystretch parameter used for vertical spacing
- * style character; can be null or "nejm" for different table styling
- * rel_size numeric; a scaling to be applied to the entire table, e.g. rel_size=-2
- * placement character; placement directive, defaults to "H"

Value

the LaTeX rendering

Examples

```
latex(cell_label("123"))
latex(hmisc_iqr(rnorm(20)))
latex(hmisc_fraction(45, 137))
tbl <- tangram(drug~bili, pbc, "tbl")
latex(tbl)
```

latexify

LaTeX safe string conversion

Description

LaTeX safe string conversion. This transforms a string handling Markdown characters and UNI-CODE as best it can with an automated pass.

Usage

```
latexify(x)
```

Arguments

x string to make LaTeX safe

Value

valid LaTeX code

latex_template	<i>Return a LaTeX template that works with tangram</i>
----------------	--

Description

Pandoc in the current version of RStudio does not allow for setting package options to xcolor and this has made it incompatible with the LaTeX generated by this package. This provides a known working template.

Usage

```
latex_template()
```

Details

An example header would look like the following:

```
— title: "A Document Full of Beautiful Tables" output: pdf_document: "r tangram::latex_template()"
—
```

Value

filename of LaTeX template

nejm	<i>Style Bundle for Closer to NEJM style</i>
------	--

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

```
nejm
```

Format

An object of class `list` of length 5.

nejm_cell

Cell Generation functions for nejm default

Description

Each function here is called when a cell is generated. Overriding these in a formula call will allow one to customize exactly how each cell's contents are generated.

Usage

```
nejm_cell
```

Format

An object of class list of length 9.

Details

While this serves as the base template for transforms, it is by no means required if one develops their own bundle of data transforms. One can create any number of cell level styling choices.

nejm_fraction

Create an cell_fraction (S3) in NEJM style of the given data

Description

A cell object contains a statistical result of a fraction/percentage in nejm style

Usage

```
nejm_fraction(numerator, denominator, format = NULL, ...)
```

Arguments

numerator	numeric; The value of the numerator
denominator	numeric; The value of the denominator
format	numeric or character; a string formatting directive
...	optional extra information to attach

Value

A cell_fraction object.

Examples

```
nejm_fraction(1, 4, 3)
```

nejm_iqr

*Create a interquartile range cell object of the given data NEJM style***Description**

Construct a cell which has the 3 interquartile ranges specified.

Usage

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

Arguments

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

Value

A `cell_quantile` object.

Examples

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

nejm_range	Create a NEJM style range
------------	---------------------------

Description

Construct a cell which has the range of the given data in NEJM style

Usage

```
nejm_range(x, format, ...)
```

Arguments

x	numeric vector whose range is desired
format	numeric or character; an argument to pass to the formatting function
...	additional arguments to passed to cell()

Parser	<i>The parser class for generating abstract syntax trees for given table formulas.</i>
--------	--

Description

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

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

Format

[R6Class](#) object.

References

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

Public fields

input	Storage for input string of a formula
pos	The current parsing position
len	The length of the input

Methods

Public methods:

- `Parser$new()`
- `Parser$expect()`
- `Parser$peek()`
- `Parser$eat_whitespace()`
- `Parser$next_token()`
- `Parser$format()`
- `Parser$r_expression()`
- `Parser$factor()`
- `Parser$term()`
- `Parser$expression()`
- `Parser$table_formula()`
- `Parser$run()`
- `Parser$clone()`

Method `new()`: Create a parser

Usage:

`Parser$new()`

Method `expect()`: Specify expectation of next token from lexer

Usage:

`Parser$expect(id)`

Arguments:

`id` The token id expected in stream, otherwise it's an error

Method `peek()`: Peek at the next token from parser

Usage:

`Parser$peek()`

Method `eat_whitespace()`: Remove white space to find start of next token

Usage:

`Parser$eat_whitespace()`

Method `next_token()`: Returns next lexical token

Usage:

`Parser$next_token()`

Method `format()`: Return format string as token from lexical stream

Usage:

`Parser$format()`

Method `r_expression()`: Return R expression as token from lexical stream

Usage:

Parser\$r_expression()

Method factor(): Return next factor as token.

Usage:

Parser\$factor()

Method term(): Parse and return next term in stream

Usage:

Parser\$term()

Method expression(): Parse and return next expression in stream

Usage:

Parser\$expression()

Method table_formula(): Parse and return table formula from stream

Usage:

Parser\$table_formula()

Method run(): Run the parser

Usage:

Parser\$run(x)

Arguments:

x (character,formula) The table specification to parse

Method clone(): The objects of this class are cloneable with this method.

Usage:

Parser\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

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.

pipe.tangram

Provide a "|" operator for cbind of tangram tables

Description

The pipe operator provides an cbind for tangram tables

Usage

```
## S3 method for class 'tangram'
x | y
```

Arguments

x	left argument for rbind
y	right argument for rbind

Value

A column wise merged tangram object

print.cell	<i>Print methods for tangram objects</i>
------------	--

Description

Print methods for tangram objects

Usage

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

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

## S3 method for class 'summary.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(hmisc_iqr(rnorm(20)))
print(hmisc_fraction(45, 137))
print(tangram(1,1) %>%
  row_header("row") %>%
  col_header(1,2,3) %>%
  add_col("A", "B", "C"))
print(tangram(drug~bili, pbc))
```

proc_tab

Tangram transform for proc_tab style summaries via a function

Description

Given a function that produces a vector of tangram cells, will generate a table

Usage

```
proc_tab(table, row, column, fun = NULL, overall = FALSE, ...)
```

Arguments

table	The table builder object
row	The row from the abstract syntax tree that parsed the formula
column	The column from the abstract syntax tree that parsed the formula
fun	The function to apply to the broken out categories
overall	Provide a summary of categorical breakdowns
...	additional arguments to pass to fun

rbind.tangram

An rbind for generated tables tangram objects.

Description

Execute the equivalent of an rbind for generated tables

Usage

```
## S3 method for class 'tangram'
rbind(..., deparse.level = 1)
```

Arguments

...	tangram objects to rbind
deparse.level	numeric; not used

Value

A merged tangram object

render_f	<i>Format a vector of provided numeric values</i>
----------	---

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")
```

render_route_tangram	<i>Router for rendering method</i>
----------------------	------------------------------------

Description

This functions detects if knitr is loaded, and does it's best to determine the output format from knitr and returns the appropriate rendering function.

Usage

```
render_route_tangram()
```

Value

A rendering function to use

replace_cell	<i>Replace a cell's contents</i>
--------------	----------------------------------

Description

Replace a cell in a table

Usage

```
replace_cell(table, row, col, object, ...)
```

Arguments

table	the tangram table to modify
row	numeric; The row to modify
col	numeric; The column to modify
object	The cell or object to replace in a table
...	Additional parameters passed to cell function if not given a cell object

Value

the modified table

rmd	<i>Generate an Rmd table entry from a cell object</i>
-----	---

Description

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

Usage

```
rmd(object, key = FALSE, ...)

## Default S3 method:
rmd(object, key = FALSE, ...)

## S3 method for class 'cell'
rmd(object, key = FALSE, ...)

## S3 method for class 'cell_n'
rmd(object, key = FALSE, ...)

## S3 method for class 'tangram'
rmd(object, key = NULL, append = FALSE, pad = 10, ...)
```

Arguments

object	The cell_fstat for indexing
key	A filename to write key values into. Can be false if no key file is desired.
...	additional arguments to renderer. Unused
append	logical; Should the key file be appended too, or overwritten
pad	numeric; Minimum width of columns can be a single or vector of numerics.

Value

A string representation of the table

Examples

```
rmd(tangram(drug ~ bili, pbc))
```

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)
```

Arguments

x	object; object to determine requested count
---	---

rtf	<i>S3 rtf Method function for use on abstract table class</i>
-----	---

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	<i>Given a cell class create an RTF representation.</i>
----------	---

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_chi2	<i>Convert an abstract cell_chi2 object into an rtf string</i>
---------------	--

Description

Given a cell_chi2 class create an rtf representation.

Usage

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

Arguments

object	The cell chi2 to render to HTML5
id	A unique identifier for traceability
...	additional arguments to renderer. Unused

Value

A text string rendering of the given chi2 in rtf

rtf.cell_fstat	<i>Convert an abstract cell_fstat object into an RTF string</i>
----------------	---

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	<i>Convert an abstract cell_header object into an RTF string</i>
-----------------	--

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	<i>Convert an abstract cell_iqr object into an RTF string</i>
--------------	---

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	<i>Given a cell_label class create an RTF representation.</i>
----------------	---

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	<i>Convert an abstract cell_n object into an RTF string</i>
------------	---

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	<i>Convert an abstract cell_subheader object into an RTF string</i>
--------------------	---

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	<i>Default conversion to RTF for an abstract table element</i>
-------------	--

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 generated
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

<code>select_col</code>	<i>Select given column(s) from a table</i>
-------------------------	--

Description

Given a table, select the specified column(s)

Usage

```
select_col(table, col)
```

Arguments

- `table` the table to modify
- `col` vector containing column(s) to select

Value

the modified table

<code>select_row</code>	<i>Select given row(s) from a table</i>
-------------------------	---

Description

Given a table, select the specified rows

Usage

```
select_row(table, row)
```

Arguments

- `table` the table to modify
- `row` vector with row numbers to select

Value

the modified table

smd	<i>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.</i>
-----	---

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

smd

Format

An object of class `list` of length 5.

smd_cell	<i>Cell Generation functions for SMD comparisons of categorical to numerical</i>
----------	--

Description

Each function here is called when a cell is generated. Overriding these in a formula call will allow one to customize exactly how each cell's contents are generated.

Usage

smd_cell

Format

An object of class `list` of length 7.

Details

While this serves as the base template for transforms, it is by no means required if one develops their own bundle of data transforms. One can create any number of cell level styling choices.

smd_compare

*Create a SMD 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

```
smd_compare(
  table,
  row,
  column,
  cell_style,
  style,
  smdformat = NULL,
  pformat = NULL,
  weight = NULL,
  test = FALSE,
  ...
)
```

Arguments

table	The table object to modify
row	The row variable object to use (numerical)
column	The column variable to use (categorical)
cell_style	list; cell styling functions
style	character; chosen styling to final table
smdformat	numeric, character or function; A formatting directive to be applied to smd
pformat	numeric, character or function; A formatting directive to be applied to p-values
weight	numeric; Vector of weights to apply to data when computing SMD
test	logical; include statistical test results
...	absorbs additional arguments. Unused at present.

Value

The modified table object

smd_contingency	Create a contingency table with SMD given a row column of a formula
-----------------	---

Description

Create a contingency table with SMD given a row column of a formula

Usage

```
smd_contingency(
  table,
  row,
  column,
  cell_style,
  style,
  smdformat = NULL,
  collapse_single = TRUE,
  weight = NULL,
  test = FALSE,
  pformat = NULL,
  ...
)
```

Arguments

table	The tablebuilder object
row	The row node from the parser of the formula
column	The column node provided by the parser of the formula
cell_style	A list of all individual cell stylings to apply
style	The global style to apply.
smdformat	The format command to apply to smd
collapse_single	Should single factor variables be collapsed
weight	Any weighting to apply to data for computation of SMD
test	logical; include statistical test results
pformat	numeric, character or function; A formatting directive to be applied to p-values
...	Additional arguments to provide cell generation functions

Value

The resulting sub table constructed

smd_dist	<i>Create an SMD distance cell</i>
----------	------------------------------------

Description

Create an SMD distance cell. It calls the smd function then formats the result. If the result rounds to all zeros then it appends a less than sign and bumps the least significant digit to one.

Usage

```
smd_dist(x, group, format, weight = NULL, ...)
```

Arguments

x	vector; variable to evaluate with smd
group	factor; A grouping to apply. Must have 2 levels.
format	formatting to apply to result
weight	numeric; Weighting to apply to computation. Defaults to NULL.
...	additional arguments to pass to cell generation

Value

a tangram cell

smd_fraction	<i>Create a fraction cell in the smd transform</i>
--------------	--

Description

Create a fraction cell in the smd transform. In this instance it print the numerator followed by percentage in parenthesis.

Usage

```
smd_fraction(num, den, format, ...)
```

Arguments

num	numerator of fraction
den	denominator of fraction
format	formatting to apply to result
...	additional arguments to pass to cell generation

Value

a tangram cell

smd_meansd	Create an SMD mean and standard deviation cell
------------	--

Description

Create an SMD mean and standard deviation cell. In this case it prints the mean with the standard deviation in parenthesis

Usage

```
smd_meansd(x, format, ...)
```

Arguments

x	vector; variable to evaluate with smd
format	formatting to apply to result
...	additional arguments to pass to cell generation

Value

a tangram cell

standard_difference	<i>Compute the standardized mean distance between 2 groups for numerical or categorical information. Using method described in 'A unified approach to measuring the effect size between two groups using SAS' by Dongsheng Yand and Jarrod E. Dalton, 2012. SAS Global Forum 2012</i>
---------------------	---

Description

Compute the standardized mean distance between 2 groups for numerical or categorical information. Using method described in 'A unified approach to measuring the effect size between two groups using SAS' by Dongsheng Yand and Jarrod E. Dalton, 2012. SAS Global Forum 2012

Usage

```
standard_difference(x, group, weight = NULL)
```

Arguments

x	vector; data to estimate effect size for groups
group	vector; the grouping variable.
weight	vector; weighting information for x

`summarize_kruskal_horz`*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. There are additionally a list of cell tranforms that can be overridden and a default footnote if none is specified.

Usage

```
summarize_kruskal_horz(  
  table,  
  row,  
  column,  
  cell_style,  
  pformat = NULL,  
  msd = FALSE,  
  quant = c(0.25, 0.5, 0.75),  
  overall = NULL,  
  test = FALSE,  
  ...  
)  
  
summarize_kruskal_vert(  
  table,  
  row,  
  column,  
  cell_style,  
  collapse_single = TRUE,  
  pformat = NULL,  
  msd = FALSE,  
  test = FALSE,  
  ...  
)  
  
summarize_chisq(  
  table,  
  row,  
  column,  
  cell_style,  
  pformat = NULL,  
  collapse_single = TRUE,  
  overall = NULL,
```

```

    test = FALSE,
    row_percents = FALSE,
    useNA = "no",
    ...
)

summarize_spearman(
  table,
  row,
  column,
  cell_style,
  pformat = NULL,
  test = FALSE,
  ...
)

hmisc

```

Arguments

table	The table object to modify
row	The row variable object to use (numerical)
column	The column variable to use (categorical)
cell_style	list; cell styling functions
pformat	numeric, character or function; 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 or character; Include overall summary statistics for a categorical column. Character values are assumed to be true and used as column header.
test	logical or function; include statistical test results. Function signature must be function(row, col, cell_style, ...)
...	absorbs additional arguments. Unused at present.
collapse_single	logical; default TRUE. Categorical variables with a two values collapse to single row.
row_percents	logical; use denominator across rows instead of columns.
useNA	character; Specifies whether to include NA counts in the table. The allowed values correspond to never "no" (Default), only if the count is positive "ifany" and even for zero counts "always". An NA column is always excluded.

Format

An object of class `list` of length 5.

Value

The modified table object

`summarize_kruskal_horz`

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

`summarize_kruskal_vert`

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

`summarize_chisq`

Given a row and column object from the parser apply a chi² test and output the results

`summarize_spearman`

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

`hmisc`

```
hmisc <- list(
  Type      = hmisc_data_type,
  Numerical = list(
    Numerical = summarize_spearman,
    Categorical = summarize_kruskal_horz
  ),
  Categorical = list(
    Numerical = summarize_kruskal_vert,
    Categorical = summarize_chisq
  ),
  Cell      = hmisc_cell,
  Footnote  = "N is the number of non-missing value. ^1^Kruskal-Wallis. ^2^Pearson. ^3^Wilcoxon."
)
```

See Also

[hmisc_data_type](#), [tangram](#), [hmisc_cell](#)

summarize_nejm_horz	<i>Create a summarization for a categorical set of column versus a numerical row in NEJM style</i>
---------------------	--

Description

Given a row and column object from the parser apply a Kruskal test and output the results horizontally. 5 X (n + no. categories + test statistic)

Usage

```
summarize_nejm_horz(
  table,
  row,
  column,
  cell_style,
  pformat = NULL,
  msd = FALSE,
  quant = c(0.25, 0.5, 0.75),
  overall = NULL,
  test = FALSE,
  useNA = "no",
  ...
)
```

Arguments

table	The table object to modify
row	The row variable object to use (numerical)
column	The column variable to use (categorical)
cell_style	list; cell styling functions
pformat	numeric, character or function; 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 or character; Include overall summary statistics for a categorical column. Character values are assumed to be true and used as column header.
test	logical or function; include statistical test results. Function signature must be function(row, col, cell_style, ...)
useNA	character; Specifies whether to include NA counts in the table. The allowed values correspond to never "no" (Default), only if the count is positive "ifany" and even for zero counts "always". An NA column is always excluded.
...	absorbs additional arguments. Unused at present.

Value

The modified table object

summarize_nejm_vert	Create a summarization for a categorical row versus X 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_nejm_vert(  
  table,  
  row,  
  column,  
  cell_style,  
  collapse_single = TRUE,  
  pformat = NULL,  
  msd = FALSE,  
  test = FALSE,  
  quant = c(0.25, 0.5, 0.75),  
  ...  
)
```

Arguments

table	The table object to modify
row	The row variable object to use (categorical)
column	The column variable to use (numerical)
cell_style	list; cell styling functions
collapse_single	logical; default TRUE. Categorical variables with a two values collapse to single row.
pformat	numeric, character or function; A formatting directive to be applied to p-values
msd	logical; include msd in summary
test	logical; include statistical test results
quant	numeric; vector of quantiles to include. Should be an odd number since the middle value is highlighted on display.
...	absorbs additional arguments. Unused at present.

Value

The modified table object

summary.tangram	<i>The default method for rendering tangram objects</i>
-----------------	---

Description

A tangram is a summary, so it returns itself. Otherwise convert to a text representation.

Usage

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

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

Arguments

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

Value

the text summary

Examples

```
summary(cell_label("123"))
summary(hmisc_iqr(rnorm(20)))
summary(hmisc_fraction(45, 137))
summary(tangram(1,1) %>%
  row_header("row") %>%
  col_header(1,2,3) %>%
  add_col("A", "B", "C"))
summary(tangram(drug~bili, pbc))
```

table_flatten	<i>Given a tangram object with embedded tables, flattens to a single table.</i>
---------------	---

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.clmm2	<i>Table creation methods</i>
---------------	-------------------------------

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

```
## S3 method for class 'clmm2'
tangram(
  x,
  id = NULL,
  style = "hmisc",
  caption = NULL,
  footnote = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'summary.clmm2'
tangram(
  x,
  id = NULL,
  style = "hmisc",
  caption = NULL,
  footnote = NULL,
  digits = NULL,
  pformat = "%1.3f",
  include_p = FALSE,
  ...
)

tangram(x, ...)
```

```
## S3 method for class 'numeric'
tangram(
  x,
  cols,
  id = NULL,
  caption = NULL,
  style = "hmisc",
  footnote = NULL,
  fixed_thead = NULL,
  ...
)

## S3 method for class 'anova.lme'
tangram(
  x,
  id = NULL,
  style = "hmisc",
  caption = NULL,
  footnote = NULL,
  digits = NULL,
  fixed_thead = NULL,
  ...
)

## S3 method for class 'data.frame'
tangram(
  x,
  id = NULL,
  colheader = NA,
  caption = NULL,
  style = "hmisc",
  footnote = NULL,
  after = NA,
  quant = seq(0, 1, 0.25),
  msd = TRUE,
  as.character = NULL,
  fixed_thead = NULL,
  exclude = NULL,
  ...
)

## S3 method for class 'formula'
tangram(
  x,
  data = NULL,
  id = NULL,
  transforms = NULL,
  caption = NULL,
```

```
    style = "hmisc",
    footnote = NULL,
    after = NA,
    digits = NA,
    fixed_thead = NULL,
    exclude = NULL,
    ...
)

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

## S3 method for class 'table'
tangram(
  x,
  id = NULL,
  percents = FALSE,
  digits = 1,
  test = FALSE,
  footnote = NULL,
  ...
)

## S3 method for class 'ftable'
tangram(x, id = NULL, ...)

## S3 method for class 'matrix'
tangram(x, digits = NULL, ...)

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

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

## S3 method for class 'summary.lm'
tangram(x, id = NULL, format = NULL, pformat = NULL, tformat = NULL, ...)

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

Arguments

<code>x</code>	object; depends on S3 type, could be rows, formula, string of a formula, data.frame or numerical rows, an rms.model
<code>id</code>	character; A unique charcter id used to identify this table over multiple runs. No spaces.
<code>style</code>	character; Desired rendering style, currently supports "hmisc", "nejm", and "lancet". Defaults to "hmisc"
<code>caption</code>	character; A string with the desired caption
<code>footnote</code>	character; A vector of character strings as footnotes
<code>digits</code>	numeric; default number of digits to use for display of numerics
<code>...</code>	addition models or data supplied to table construction routines
<code>pformat</code>	function or character; A function to format p values
<code>include_p</code>	logical; Include p-value when printing statistic
<code>cols</code>	numeric; An integer of the number of cols to create
<code>fixed_thead</code>	logical; On conversion to HTML5 should headers be treated as fixed?
<code>colheader</code>	character; Use as column headers in final table
<code>after</code>	function or list of functions; one or more functions to further process an abstract table
<code>quant</code>	numeric; A vector of quantiles to use for summaries
<code>msd</code>	logical; Include mean and standard deviation in numeric summary
<code>as.character</code>	logical; if true data.frames all variables are passed through as.character and no numerical summary is provided.
<code>exclude</code>	vector or list; When x is a data.frame this exclusion criteria is applied to the data. If this is a list then each list pair is the (column name, criteria). It is preferred to use a list to be specific.
<code>data</code>	data.frame; data to use for rendering tangram object
<code>transforms</code>	list of lists of functions; that contain the transformation to apply for summarization
<code>percents</code>	logical; Display percents when rendering a table object. Defaults to FALSE
<code>test</code>	logical or function; Perform default test or a statistical function that will return a test result when passed a row and column
<code>format</code>	numeric or character; Format to apply to statistic
<code>tformat</code>	numeric or character; format to apply to t-value
<code>short.labels</code>	numeric; Named vector of variable labels to replace in interaction rows. Must be in format c("variable name" = "shortened label").
<code>rnd.digits</code>	numeric; Digits to round reference, comparison, result and CI values to. Defaults to 2.
<code>rnd.stats</code>	numeric; Digits to round model LR, R2, etc to. Defaults to rnd.digits.

Details

Note that additional arguments are passed to any subsequent transform. This means that a lot of possible arguments are not documented here but in the transform applied. Examine their documentations for additional possible arguments if needed.

Value

A tangram object (a table).

See Also

Possible transforms are (see [hmisc](#)) (*default*), [nejm](#) and [lancet](#).

Examples

```
tangram(1, 1)
tangram(data.frame(x=1:3, y=c('a','b','c')), id="mytbl1")
tangram(drug ~ bili + albumin + protime + sex + age + spiders, pbc, id="mytbl2")
tangram("drug~bili+albumin+stage::Categorical+protime+sex+age+spiders", pbc,"mytbl3")
```

Token	<i>A token in the formula grammar</i>
-------	---------------------------------------

Description

A token in the formula grammar
A token in the formula grammar

Format

[R6Class](#) object.

Public fields

id The token identifier, E.g. "LPAREN"
name Information about the token, useful with IDENTIFIERS.

Methods

Public methods:

- [Token\\$new\(\)](#)
- [Token\\$clone\(\)](#)

Method new(): Construct a lexical token

Usage:

```
Token$new(id, name = "")
```

Arguments:

id (character) The lexical id of the token

name (character) Additional token information if needed

Method clone(): The objects of this class are cloneable with this method.

Usage:

Token\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Index

- * **data**
 - ASTBranch, 7
 - ASTFunction, 8
 - ASTMultiply, 10
 - ASTNode, 11
 - ASTPlus, 12
 - ASTTableFormula, 14
 - ASTVariable, 15
 - hmisc_p, 27
 - lancet, 41
 - lancet_cell, 42
 - nejm, 46
 - nejm_cell, 47
 - Parser, 49
 - pbc, 52
 - smd, 65
 - smd_cell, 65
 - summarize_kruskal_horz, 70
 - Token, 80
- +. tangram, 4
- add_col (col_header), 21
- add_footnote, 5
- add_indent, 5
- add_row (col_header), 21
- args_flatten, 6
- as.categorical, 6
- ASTBranch, 7
- ASTFunction, 8
- ASTMultiply, 10
- ASTNode, 11, 15
- ASTPlus, 12
- ASTTableFormula, 14
- ASTVariable, 15
- carriage_return (col_header), 21
- cbind.tangram, 17
- cell, 17
- cell_header, 18
- cell_label, 18
- cell_n, 19
- cell_subheader, 20
- cell_transform, 20
- col_header, 21
- cols (rows), 57
- csv, 22
- cursor_down (col_header), 21
- cursor_left (col_header), 21
- cursor_pos (col_header), 21
- cursor_right (col_header), 21
- cursor_up (col_header), 21
- custom_css, 23
- del_col, 24
- del_row, 24
- derive_label, 25
- drop_statistics, 25
- format_guess, 26
- hmisc, 27, 30, 80
- hmisc (summarize_kruskal_horz), 70
- hmisc_cell, 72
- hmisc_cell (hmisc_p), 27
- hmisc_chi2 (hmisc_p), 27
- hmisc_data_type, 26, 72
- hmisc_fraction (hmisc_p), 27
- hmisc_fstat (hmisc_p), 27
- hmisc_intercept_cleanup, 27
- hmisc_iqr (hmisc_p), 27
- hmisc_p, 27
- hmisc_spearman (hmisc_p), 27
- hmisc_wilcox (hmisc_p), 27
- home (col_header), 21
- html5, 30
- html5.cell, 31
- html5.cell_header, 31
- html5.cell_label, 32
- html5.cell_n, 32
- html5.cell_subheader, 33

[html5.character](#), 33
[html5.default](#), 34
[html5.logical](#), 34
[html5.tangram](#), 35

[index](#), 36
[index.cell_label](#), 36
[index.default](#), 37
[index.list](#), 37
[index.tangram](#), 38
[insert_column](#), 38
[insert_row](#), 39
[is.binomial](#), 39
[is.categorical](#), 40

[key](#), 41

[lancet](#), 41, 80
[lancet_cell](#), 42
[lancet_fraction](#), 42
[lancet_mean_sd](#), 43
[latex](#), 44
[latex_template](#), 46
[latexify](#), 45
[line_feed \(col_header\)](#), 21

[nejm](#), 46, 80
[nejm_cell](#), 47
[nejm_fraction](#), 47
[nejm_iqr](#), 48
[nejm_range](#), 49
[new_col \(col_header\)](#), 21
[new_line \(col_header\)](#), 21
[new_row \(col_header\)](#), 21

[Parser](#), 49
[pbc](#), 52
[pipe.tangram](#), 52
[print.cell](#), 53
[print.summary.tangram \(print.cell\)](#), 53
[print.tangram \(print.cell\)](#), 53
[proc_tab](#), 54

[R6Class](#), 7, 8, 10, 12, 14, 15, 49, 80
[rbind.tangram](#), 54
[render_f](#), 55
[render_route_tangram](#), 55
[replace_cell](#), 56
[rmd](#), 56
[row_header \(col_header\)](#), 21

[rows](#), 57
[rtf](#), 58
[rtf.cell](#), 58
[rtf.cell_chi2](#), 59
[rtf.cell_fstat](#), 59
[rtf.cell_header](#), 60
[rtf.cell_iqr](#), 60
[rtf.cell_label](#), 61
[rtf.cell_n](#), 61
[rtf.cell_subheader](#), 62
[rtf.default](#), 62
[rtf.tangram](#), 63

[select_col](#), 64
[select_row](#), 64
[set_caption \(col_header\)](#), 21
[set_colspan \(col_header\)](#), 21
[set_footnote \(col_header\)](#), 21
[set_id \(col_header\)](#), 21
[set_rowspan \(col_header\)](#), 21
[set_style \(col_header\)](#), 21
[smd](#), 65
[smd_cell](#), 65
[smd_compare](#), 66
[smd_contingency](#), 67
[smd_dist](#), 68
[smd_fraction](#), 68
[smd_meansd](#), 69
[standard_difference](#), 69
[summarize_chisq](#)
 ([summarize_kruskal_horz](#)), 70
[summarize_kruskal_horz](#), 70
[summarize_kruskal_vert](#)
 ([summarize_kruskal_horz](#)), 70
[summarize_nejm_horz](#), 73
[summarize_nejm_vert](#), 74
[summarize_spearman](#)
 ([summarize_kruskal_horz](#)), 70
[summary.cell \(summary.tangram\)](#), 75
[summary.tangram](#), 75

[table_apply \(col_header\)](#), 21
[table_flatten](#), 75
[tangram](#), 72
[tangram \(tangram.clmm2\)](#), 76
[tangram.clmm2](#), 76
[tangram::ASTBranch](#), 10, 13, 14
[tangram::ASTNode](#), 7, 8, 10, 13–15
[Token](#), 80

`write_cell(col_header)`, [21](#)