Package 'tablesgg'

October 14, 2022

Version 0.8-1

Date 2021-05-31

Title Presentation-Quality Tables, Displayed Using 'ggplot2'

Author Richard Raubertas [aut, cre]

Maintainer Richard Raubertas <rrprf@emvt.net>

License GPL (>= 3)

Imports ggplot2, tables, graphics, grDevices, stats, tools, utils, grid

Suggests ggtext, gridtext, quadprog, xtable, knitr, rmarkdown

Description Presentation-quality tables are displayed as plots on an R graphics device. Although there are other packages that format tables for display, this package is unique in combining two features: (a) It is aware of the logical structure of the table being presented, and makes use of that for automatic layout and styling of the table. This avoids the need for most manual adjustments to achieve an attractive result. (b) It displays tables using 'ggplot2' graphics. Therefore a table can be presented anywhere a graph could be, with no more effort. External software such as LaTeX or HTML or their viewers is not required. The package provides a full set of tools to control the style and appearance of tables, including titles, footnotes and reference marks, horizontal and vertical rules, and spacing of rows and columns. Methods are included to display matrices; data frames; tables created by R's ftable(), table(), and xtabs() functions; and tables created by the 'tables' and 'xtable' packages. Methods can be added to display other table-like objects. A vignette is included that illustrates usage and options available in the package.

URL https://github.com/rrprf/tablesgg

LazyData yes

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-02 19:00:02 UTC

R topics documented:

tablesgg-package *Presentation-Quality Tables, Displayed Using* **ggplot2**

Description

Presentation-quality tables are displayed as plots on an R graphics device. Although there are other packages that format tables for display, this package is unique in combining two features: (a) It is aware of the logical structure of the table being presented, and makes use of that for automatic layout and styling of the table. This avoids the need for most manual adjustments to achieve an attractive result. (b) It displays tables using **ggplot2** graphics. Therefore a table can be presented anywhere a graph could be, with no more effort. External software such as LaTeX or HTML or their viewers is not required.

Methods are included to display matrices; data frames; tables created by R's `ftable`, `table`, and `xtabs` functions; and tables created by the **tables** and **xtable** packages. Methods can be added to display other table-like objects.

Other package features:

- A full set of tools is provided to control the appearance of tables, including titles, footnotes and reference marks, horizontal and vertical rules, and spacing of rows and columns. Many properties can be set automatically by specifying *styles*. Default styles are included, and the user can define custom styles.
- There are tools for low-level manipulation of the appearance of individual table elements if desired.
- All sizes and dimensions in displayed tables are specified in physical units (points for font size, millimeters for everything else). Therefore a plotted table has a well-defined physical size, independent of the size of the graphics device on which it is displayed. The user can easily increase or decrease the displayed size by a scale factor, maintaining the relative proportions of table elements.
- Since the plotted tables are ordinary `ggplot` objects, the facilities of **ggplot2** and its various extension packages are available to modify or manipulate the table. For example, the table can be inserted as an image within another plot.

A vignette is included that illustrates usage and options available in the package.

acol *Column Numbers Within the Augmented Row-Column Grid for a Table*

Description

Return the column numbers associated with a specified table part or element, or with a set of column header values, within the augmented row-column grid of a table.

Usage

```
acol(x, id=NULL, hpath=NULL)
```

Arguments

x	A textTable or a plotted table (pltdTable) object.
id	Character scalar containing the ID of a single table part, block, entry, or hvrule. (If x is a textTable, only the ID of a table part is allowed.)
hpath	Character vector with length between 0 and the number of layers in the column header. The i-th element should be one of the values in the i-th header row, or NA. See DETAILS. Only one of id and hpath should be specified.

Details

See the documentation for adim for more information about the augmented row-column grid of a table.

Only one of arguments id and hpath should be specified. id is searched for first among table parts (the only thing available for a textTable), then blocks, entries, and hvrules, in that order. The search stops at the first match. It is an error if id is not found in any of these.

hpath is short for "header path". It is used to obtain column numbers associated with specified combinations of values of the column header variables. Suppose there are L layers of column headers. If the length of hpath is less than L, NA values are added at the end to reach that length. The function returns the intersection of the column numbers for which the i-th outermost of the header layers equals the i-th element of hpath. An NA in hpath is taken to match all values in the correponding layer of column headers. Thus, if L == 4 and hpath=c("a", NA, "c"), the function will return the column numbers for which the outermost column header has a value of "a" _and_ the third outermost has a value of "c". If no column has the combination of values specified by hpath then the returned vector will have length 0.

Since hpath refers to values of column header variables, it cannot be used to get column numbers associated with table annotation, or with the row header. (Use id instead.)

Value

A numeric vector containing column numbers within the table's augmented row-column grid. The column numbers are those partially or completely occupied by the cells associated with id or hpath. They will be increasing but not necessarily consecutive.

The returned vector may have length 0 if id refers to a table part or block that spans no columns, or if hpath matches no set of column header values.

Note that for a vertical hvrule (vrule), the "column number" is actually a half-integer, bracketed by the table column numbers between which the vrule runs. For example, if the vrule runs between table columns 3 and 4, the returned value will be c(3.5).

See Also

adim to get the dimensions of the augmented row-column grid; arow for the corresponding operation on rows; ids

Examples

```
ttbl <- textTable(iris2_tab, title=c("Title 1", "2nd title"), foot="Foot")
plt <- plot(ttbl)
```

```
acol(plt, id="title")  # block "title" spans all columns
acol(plt, id="body,4,2")  # single entry
# Remove the columns for "Petal" measurements (a value in column
# header layer 2):
plot(ttbl[, -acol(ttbl, hpath=c(NA, "Petal"))])
# Remove the "Length" measurements (a value in column header layer 3):
plot(ttbl[, -acol(ttbl, hpath=c(NA, NA, "Length"))])
# Remove the "Length" measurements just for "Petal":
plot(ttbl[, -acol(ttbl, hpath=c(NA, "Petal", "Length"))])
```

`addBlock` *Define a New Block of Cells in a Table*

Description

Define a new block (rectangular set of cells) in a table. The location and graphical properties of the block are specified explicitly, rather than being generated automatically from the logical structure of the table and a style.

Usage

```
addBlock(x, arows, acols, id, props=NULL, enabled=FALSE)
```

Arguments

`x`	A plotted table (`pltdTable`) object.
`arows, acols`	Numeric vectors specifying the cells contained in the block, with respect to the augmented row-column grid of the table. The block includes the cells in row numbers from `min(arows)` to `max(arows)`, and column numbers from `min(acols)` to `max(acols)`.
`id`	Optional character string giving the ID to be assigned to the new block. It is an error if there is already a block with this ID in `x`. The default is to generate an ID of the form `block*`, where `*` is an integer.
`props`	Optional `element_block` object with graphical properties to assign to the new block. Any graphical properties not specified in `props` will be taken from `blockStyle_pkg_base` in `styles_pkg`.
`enabled`	Logical scalar, whether the new block is to be enabled for display. The default is FALSE.

Details

Normally blocks are defined automatically, based on the logical structure of the table and the style selected by the user. This function allows additional blocks to be defined "manually", explicitly specifying their position and span in terms of row and column numbers.

There are two typical situations in which one would want to define a new block. The first is to highlight a specific set of cells in the table visually, by shading or a border. For that purpose one should specify `enabled=TRUE` (so the block will be displayed) and perhaps `props` (for non-default graphical properties).

The second reason to define a new block is to use its ID as a quick way to refer to the entries within it, for example to set their graphical properties using `props<-`. In that case `enabled` for the block should be FALSE, since the block itself is not to be displayed.

Row and column numbers are with respect to the augmented row-column grid of the table. See `?adim` for more more information about this grid. The helper functions `arow` and `acol` can be used to specify arguments `arows` and `acols` in terms of table parts or previously defined blocks.

Graphical properties for blocks defined by this function will not be changed if a new block style is applied to the plotted table. Use one of the `props<-` functions instead.

There is no way to remove or undefine a block, other than recreating the plotted table object from scratch. However they can be disabled using a `props<-` function, and then will not be displayed.

Value

A plotted table object like `x`, with the new block defined.

See Also

`arow`, `acol`, `adim`, `element_block`

Examples

```
plt <- plot(iris2_tab, title="The iris data",
            subtitle="Summary statistics by species")
plt <- addBlock(plt, arows=c(8, 9), acols=c(3, 4), id="new_block",
                props=element_block(border_color="red", border_size=1.0),
                enabled=TRUE)
plt
# Can refer to the new block by its ID:
props(plt, id="new_block") <- element_entry(fontface=3)  # italics
plt
```

`addHvrule` *Add a Horizontal or Vertical Rule (Hvrule) to a Table*

Description

Add a horizontal or vertical rule (hvrule) to a table. The location, span, and graphical properties of the hvrule are specified explicitly, rather than being generated automatically from the logical structure of the table and a style.

Usage

```
addHvrule(x, direction, arows, acols, id, props=NULL, enabled=TRUE)
```

Arguments

x	A plotted table (pltdTable) object.
direction	Character string specifying whether the rule is to be horizontal (hrule) or vertical (vrule).
arows, acols	Numeric vectors specifying the location and span of the hvrule, with respect to the augmented row-column grid of the table. For an hrule, arows should be a single value: the half-integer bracketed by the table rows between which the rule runs. For example, an hrule running between rows 3 and 4 should have arows equal to 3.5. acols should be a vector of integers whose range specifies the column numbers spanned by the rule. For a vrule the roles of arows and acols are reversed: arows is a vector of integers whose range specifies the row numbers spanned by the rule, and acols is the half-integer bracketed by the table columns between which it runs.
id	Character string giving the ID to be assigned to the new hvrule. It is an error if there is already an hvrule with this ID in x. The default is to generate an ID of the form hvrule*, where * is an integer.
props	Optional element_hvrule object with graphical properties to assign to the new hvrule. Any graphical properties not specified in props will be taken from hvruleStyle_pkg_base in styles_pkg.
enabled	Logical scalar, whether the new hvrule is to be enabled for display. The default is TRUE.

Details

Normally hvrules are generated automatically, based on the logical structure of the table and the style selected by the user. This function allows additional hvrules to be added "manually", explicitly specifying their position and span in terms of row and column numbers.

Row and column numbers are with respect to the augmented row-column grid of the table. See ?adim for more more information about this grid. The helper functions arow and acol can be used to specify arguments arows and acols in terms of table parts or previously defined blocks.

For an hrule, the default for acols is to span all table columns. For a vrule, the default for arows is to span the rows containing the body and column headers, but not the annotation.

Graphical properties for hvrules defined by this function will not be changed if a new hvrule style is applied to the plotted table. Use one of the props<- functions instead.

There is no way to remove an hvrule, other than recreating the plotted table object from scratch. However they can be disabled using a props<- function, and then will not be displayed or take up any space.

Value

A plotted table object like x, with the new hvrule added.

See Also

arow, acol, adim, element_hvrule

Examples

```
plt <- plot(iris2_tab, title="The iris data")
plt <- addHvrule(plt, direction="vrule", acols=4.5, arows=arow(plt, "body"),
                 id="new_vrule",
                 props=element_hvrule(linetype=2, color="red"), enabled=TRUE)
plt
# Can refer to the new hvrule by its ID:
props(plt, id="new_vrule") <- element_hvrule(enabled=FALSE)  # don't display it
plt
```

---

addRefmark *Add a Reference Mark to Entries in a Table*

Description

Add a reference mark (a symbol placed before or after entry text to indicate cross-references; e.g. for footnotes) to entries in a table.

Usage

```
addRefmark(x, mark, before=character(0), after=character(0),
    parts=NULL, raise, ...)
```

Arguments

x	A textTable or pltdTable object.
mark	Character string containing the reference mark.
before, after	Character strings containing regular expressions (see ?regex) that will be matched against the _text_ of table entries (using grepl). One or both of before and after may be specified.
parts	Optional character vector listing table parts. If specified, only entries in those parts will be matched against before and after. The default is to use all table parts.
raise	Logical scalar. If TRUE, the reference mark will be displayed as a superscript, using plotmath. The default is TRUE except for asterisk marks, since that character is already raised relative to other characters.
...	Additional arguments passed to grepl when matching before and after to entry text.

Details

Reference marks are placed at the beginning or end of an entry's text. If raise is TRUE they will be displayed as superscripts. This is implemented by converting the text to make use of R's plotmath facility to create the superscript. A limitation of plotmath is that it ignores newline characters

within text. Therefore raised reference marks will not work with multi-line entries, and a warning will be issued.

In addition to using numbers, letters, or asterisk as reference marks, traditional symbols can be specified by their unicode values: dagger ("\u2020"), double dagger ("\u2021"), paragraph symbol ("\u00B6"), section symbol ("\u00A7"), and double vertical bars ("\u2016"). However, unicode symbols may not be available for all OS's or graphics devices.

With this function the user identifies the entries to be marked by searching the entry text itself, via regular expressions `before` and/or `after`. For plotted tables (`pltdTable` objects), an alternative way to add reference marks is to use one of the `props<-` functions to assign an `element_refmark` to it. They allow selection of entries using other descriptors.

Value

An object like `x`. The text of table cells/entries selected by `before` and `after` will be modified to include the reference mark, and if `raise` is TRUE, those cells/entries will be flagged to indicate that they should be treated as `plotmath` expressions.

See Also

`element_refmark`, `props<-`

Examples

```
# Add reference marks to a 'textTable':
ttbl <- textTable(iris2_tab, foot="sd = standard deviation")
ttbl <- addRefmark(ttbl, mark="a", before="sd =", after="sd$")
plot(ttbl)

# Add reference marks to a 'pltdTable':
plt <- plot(textTable(iris2_tab, foot="sd = standard deviation"))
plt <- addRefmark(plt, mark="*", before="sd =", after="sd$")
plt

# To add a reference mark to just the *first* appearance of "sd", use
# 'propsa<-' instead:
plt <- plot(textTable(iris2_tab, foot="sd = standard deviation"))
plt <- addRefmark(plt, mark="a", before="sd =")
propsa(plt, arows=arow(plt, hpath=c("setosa", "sd")),
       acols=acol(plt, "rowhead")[2]) <- element_refmark("a", side="after")
plt
```

---

adim *Dimensions of the Augmented Row-Column Grid for a Table*

Description

Return the dimensions of the augmented row-column grid for a table. Along with rows and columns associated with the body of the table, the augmented grid includes rows for each title, subtitle, and foot line, and rows and columns associated with the column and row headers.

Usage

```
adim(x)
```

Arguments

x	A table, in any of the forms used within the tablesgg package. This includes textTable and pltdTable objects.

Details

It is common to think of the number of rows and columns in a table as referring to the _body_ of the table. However in this package the grid of rows and columns in the body of a table is expanded into an _augmented row-column grid_, by adding a row for each title, subtitle, and foot line; a row for each layer of column headers; a row for each _interior_ row header entry; and a column for each (non-interior) layer of row headers. The augmented grid is numbered from the upper left, so column numbers increase from left to right, and row numbers from top to bottom. Title, subtitle, and foot lines, and interior row header entries span all the columns of the grid.

This function returns the dimensions of the augmented grid.

Disabled entries are included in counting rows and columns.

The function summary gives the dimensions of individual parts within a table.

Value

A two-element numeric vector containing (number of rows, number of columns).

See Also

summary.textTable, summary.pltdTable

Examples

```
ttbl <- textTable(iris2_tab, title="Summary statistics for the iris data")
adim(ttbl)
plt <- plot(ttbl)
adim(plt)
```

---

arow *Row Numbers Within the Augmented Row-Column Grid for a Table*

Description

Return the row numbers associated with a specified table part or element, or with a set of row header values, within the augmented row-column grid of a table.

Usage

```
arow(x, id=NULL, hpath=NULL)
```

Arguments

x	A textTable or a plotted table (pltdTable) object.
id	Character scalar containing the ID of a single table part, block, entry, or hvrule. (If x is a textTable, only the ID of a table part is allowed.)
hpath	Character vector with length between 0 and the number of layers in the row header. The i-th element should be one of the values in the i-th row header column, or NA. See DETAILS. Only one of id and hpath should be specified.

Details

See the documentation for adim for more information about the augmented row-column grid of a table.

Only one of arguments id and hpath should be specified. id is searched for first among table parts (the only thing available for a textTable), then blocks, entries, and hvrules, in that order. The search stops at the first match. It is an error if id is not found in any of these.

hpath is short for "header path". It is used to obtain row numbers associated with specified combinations of values of the row header variables. Suppose there are L layers of row headers. If the length of hpath is less than L, NA values are added at the end to reach that length. The function returns the intersection of the row numbers for which the i-th outermost of the header layers equals the i-th element of hpath. An NA in hpath is taken to match all values in the correponding layer of row headers. Thus, if L == 4 and hpath=c("a", NA, "c"), the function will return the row numbers for which the outermost row header has a value of "a" _and_ the third outermost has a value of "c". If no row has the combination of values specified by hpath then the returned vector will have length 0.

Since hpath refers to values of row header variables, it cannot be used to get row numbers associated with table annotation, or with the column header. (Use id instead.)

Value

A numeric vector containing row numbers within the table's augmented row-column grid. The row numbers are those partially or completely occupied by the cells associated with id or hpath. They will be increasing but not necessarily consecutive.

The returned vector may have length 0 if id refers to a table part or block that spans no rows, or if hpath matches no set of row header values.

Note that for a horizontal hvrule (hrule), the "row number" is actually a half-integer, bracketed by the table row numbers between which the hrule runs. For example, if the hrule runs between table rows 3 and 4, the returned value will be c(3.5).

See Also

adim to get the dimensions of the augmented row-column grid; acol for the corresponding operation on columns; ids

Examples

```
ttbl <- textTable(iris2_tab, title=c("Title 1", "2nd title"), foot="Foot")
plt <- plot(ttbl)
```

```
arow(plt, id="title")  # block "title" spans first two rows
arow(plt, id="body,4,2")  # single entry
# Remove the first line of the column header:
plot(ttbl[-arow(ttbl, id="colhead")[1], ])
# Remove the "versicolor" species (a value in row header layer 1):
plot(ttbl[-arow(ttbl, hpath=c("versicolor")), ])
# Remove the means for all species (a value in row header layer 2):
plot(ttbl[-arow(ttbl, hpath=c(NA, "mean")), ])
# Remove the mean just for the versicolor species:
plot(ttbl[-arow(ttbl, hpath=c("versicolor", "mean")), ])
```

`elements` *Extract Table Elements from a Plotted Table*

Description

Extract table elements (entries, blocks, or hvrules) from a plotted table, as a simple data frame with one row per element.

Usage

```
elements(x, type=c("entry", "block", "hvrule"), enabledOnly=TRUE)
```

Arguments

`x`	A `pltdTable` object, containing a plotted table.
`type`	Character scalar indicating the type of elements to extract: one of "entry", "block", or "hvrule".
`enabledOnly`	Logical scalar. If TRUE, only elements that are currently enabled in `x` are returned.

Details

A plotted table (`pltdTable` object) has three types of elements: entries, blocks, and hvrules. *Entries* are the text strings (and associated properties) displayed in table cells. *Blocks* are rectangular sets of contiguous table cells. And *hvrules* are spacers, with or without a visible line (or "rule"), used to separate or group table rows and columns. See the package vignette for more information.

This function allows one to inspect these elements. The purpose is primarily informational: an easy way to view all the elements of a table, their descriptors (e.g., as used by styles and the `propsd<-` function), and the graphical properties they have been assigned. It is not intended as a way to edit or modify elements. For that, see `update` methods for `textTable` and `pltdTable` objects, the `props<-` set of functions, and section 4 of the package vignette.

The remainder of this section describes the columns in the returned data frame, for each element type.

Columns in `elements(x, type="entry")`

Entry descriptors:

id Character string that uniquely identifies the entry. The format is `<part>,<partrow>,<partcol>` for table parts that are matrices (`rowhead`, `rowheadLabels`, `colhead`, and `body`), and `<part>,<partrow>` for table parts that are vectors (annotation parts `title`, `subtitle`, and `foot`).

part Character string identifying the part of the table to which the entry belongs: one of "body", "rowhead", "colhead", "rowheadLabels", "title", "subtitle", or "foot".

subpart Character string with further information about the nature of of the entry within its part of the table. May be NA.

partrow, partcol Row number, column number of the entry within its table part. For parts that are vectors rather than matrices, `partrow` will be the element number within the vector, and `partcol` will be NA. If an entry spans more than one row or column, the minimum row/column number of the spanned cells is used.

headlayer How far (in number of rows or columns) from the body of the table the entry is. By definition this is 0 for entries in the body of the table. It is 1 for row/column headers immediately adjacent to the body, 2 for headers one row/column further out, and so on. The layer numbers for row header labels match those for the corresponding columns of row headers. The layers for titles, subtitles, and foot lines are the number of _rows_ from the body of the table. When a table is created with `rowheadInside` set to TRUE, the `headlayer` value for the outermost layer of row headers (and for its label, if any) is changed to 0, since the headers become interleaved with the table body.

level_in_layer Numbering of entries within a given part and headlayer. For row and column headers this is based their hierarchical structure (see the descriptors for blocks, below). For other table parts it is just an integer from 1 to the number of entries in that layer of that part.

multirow, multicolumn Logicals indicating whether the entry spans multiple rows or columns of the table.

text Character string containing the formatted content of the entry, for display. It may use `plotmath` notation for mathematical notation, or markdown/HTML tags; see `textspec` below.

type Character string identifying the type of value the entry represents (e.g., "numeric"). May be NA. The default for table annotation (`title`, `subtitle`, `foot`) and `rowheadLabels` is "character".

textspec Character string indicating any special features of the text in `text`. Currently supported values are "plain", "plotmath", and "markdown". "plotmath" indicates that the entry text contains mathematical notation as described in `?plotmath`. "markdown" means the text may contain markdown or HTML tags to control its appearance; this requires the `ggtext` package.

enabled Logical indicating whether the entry is to be displayed when the table is plotted. If FALSE the entry is ignored when laying out the table or determining its size.

arow1, arow2, acol1, acol2 Range of row and column numbers occupied by an entry, with respect to the augmented row-column grid for the table. `arow1` < `arow2` and/or `acol1` < `acol2` means the entry spans multiple rows and/or columns.

Graphical properties for entries. Values for these are assigned by a *style*, either a default style (see `?tablesggOpt`) or user-specified.

hjust Numeric horizontal justification for entry text (0=left, 0.5=center, 1=right).

vjust Numeric vertical justification for entry text (0=top, 0.5=center, 1=bottom).

color Character string; color for entry text.

alpha Numeric value in [0, 1] specifying transparency of entry text (0=transparent, 1=opaque).

size Font size for entry text, in points (72.27 points = 1 inch, 2.845 points = 1 mm).

family Character string, the font to use for entry text. "serif", "sans", and "mono" will work for all graphics devices. Other fonts may or may not be available on a particular device.

fontface Numeric indicating 1=plain, 2=bold, 3=italic, 4=bold and italic.

lineheight Numeric multiplier that adjusts interline spacing for multi-line entry text. 1.0 gives the default spacing.

angle Rotation angle for entry text, in degrees counter-clockwise from horizontal.

hpad, vpad Padding added around the sides of entry text to keep it from touching cell borders, in millimeters. `hpad` is added on both the left and right sides of the text, and `vpad` is added on both the top and bottom.

fill Character string; background color for the cell(s) containing the entry. NA means no background color.

fill_alpha Numeric value in [0, 1] specifying transparency of the cell background color (0=transparent, 1=opaque).

border_size Thickness of the border to draw around the cell(s) containing the entry text, in millimeters.

border_color Character string; color for the border around entry text. NA means no border.

minwidth, maxwidth Minimum and maximum width for the cell(s) spanned by the entry. (Here *width* is with respect to the text itself; i.e., the direction of reading for English text, and therefore measured vertically if the text is rotated by 90 or 270 degrees.) They may be expressed in two forms. Positive values are interpreted as absolute widths in millimeters, and should include the amount of padding specified by `hpad` (when `angle` is 0 or 180 degrees) or `vpad` (when `angle` is 90 or 270 degrees). Negative values are interpreted as multiples of the natural width of the text itself, *without* including padding. Thus setting `minwidth` for an entry to `-1` will guarantee that the width of the spanned cell(s) will be at least enough to contain the text without wrapping.

- An NA value for `minwidth` means there is no constraint on minimum width, and is equivalent to 0. An `Inf` value for `maxwidth` means there is no constraint on maximum width. (However in the absence of constraints, the internal algorithm favors widths as close as possible to the natural width of the entry text, without wrapping.)
- An NA value for `maxwidth` means the maximum width will be determined passively from the `maxwidth` values of other entries in the same table column(s) (if `angle` is 0 or 180) or row(s) (if `angle` is 90 or 270). (It will never be less than `minwidth`.) This may be useful for table titles and footnotes, where long text should be wrapped to fit widths implied by the other table entries.
- Setting `maxwidth` to NA or to a finite value > `-1` and < `Inf` means the spanned cell(s) may not be wide enough to hold the text without wrapping it into multiple lines. Therefore option `tablesggOpt("allowWrap")` must be TRUE, and a warning will be raised and `maxwidth` will be ignored if not.
- The general effect of setting `minwidth` to a non-zero value is to reduce or prevent text wrapping, while the general effect of setting `maxwidth` to NA or a finite value is to encourage wrapping. Settings for one entry may affect the width and wrapping of other entries, because column widths and row heights for the table as a whole must satisfy the constraints for all their entries.

- Text representing `plotmath` expressions cannot be wrapped, so `maxwidth` should be `Inf` or `<= -1` for such entries.

Columns in `elements(x, type="block")`

Block descriptors:

id Character string that uniquely identifies the block. The format is just `<type>` for blocks types that are unique. For blocks associated with row or column headers, or with row groups formed by the plot option `rowgroupSize`, ID's begin with `<type>/<subtype>/<headlayer>/<level_in_layer>`. See Appendix B of the package vignette for details about the definitions and ID's of these blocks.

type Character string that specifies the nature or structural role of the block. One of

"table" The whole table (all cells).

"title", "subtitle", "colhead", "rowhead", "rowheadLabels", "body", "foot" The standard table parts. (If there are interior row header entries, "rowhead" and "body" are omitted because the interleaving of headers and body means neither are valid blocks.)

"titles" The union of the `title` and `subtitle` parts.

"stub" If we exclude the title/subtitle and foot annotations, a table has four quadrants: the body at the lower right, the row headers at the lower left, the column headers at the upper right, and a stub at the upper left. That is, the stub consists of the cells above the row headers and to the left of the column headers. (If there are row header labels–block "rowheadLabels"–they will be in the bottom row of the stub.)

"colhead_and_stub", "rowhead_and_stub" The unions of "stub" with "colhead" and "rowhead", respectively.

"colhead_and_body", "rowhead_and_body" The unions of "body" with "colhead" and "rowhead", respectively.

"rowblock", "colblock" Collections of blocks associated with row and column headers, reflecting their hierarchical structure. "rowblock" is also the type for blocks representing row groups formed by plot option `rowgroupSize`. See Appendix B of the package vignette for details.

subtype Character string refining block type. For types "rowblock" and "colblock", subtypes are "A", "B", "C", and (for "rowblock" only) "G". See Appendix B of the package vignette for their meaning. For other block types the subtype is set to missing (NA).

headlayer For "rowblock" and "colblock" blocks, the header layer number with which the block is associated. (Layer numbers increase from innermost to outermost layer.) For other block types, `headlayer` is NA.

level_in_layer For "rowblock" and "colblock" blocks, the level number (within the header layer) with which the block is associated. Levels are numbered from 1 to the maximum number of levels in that layer. For other block types, `level_in_layer` is NA.

group_in_level For "rowblock" blocks of subtype "G" (representing sets of rows grouped according to `rowgroupSize`), the group number within a header layer and level. NA for other block types/subtypes.

had_enabled_entries Logical, set to TRUE if there were _enabled_ table entries in `x` that intersect the block. This is set at the time the plotted table (`pltdTable` object) is created. It is not updated if entries are later enabled/disabled using `props<-`, for example.

nr, nc The number of rows and columns, respectively, that the block spans. May be 0 for empty blocks.

arow1, arow2, acol1, acol2 First and last row and column numbers spanned by the block, with respect to the augmented row-column grid for the table. Empty blocks, with no rows or no columns (`nr` or `nc` equal to 0), will have the corresponding `arow*` or `acol*` set to NA.

enabled Logical indicating whether the block is to be displayed when the table is plotted. This applies only to highlighting the rectangular region occupied by the block using a fill color or border. It has no effect on display of entries or hvrules contained within the block.

Graphical properties for blocks. Values for these are assigned by a *style*, either a default style (see `?tablesggOpt`) or user-specified.

fill Character string; color used to fill the rectangular region contained in the block. NA means the region is not colored. (Blocks are drawn before table entries or hvrules, so the block fill color will not hide those elements even if it is opaque.)

fill_alpha Numeric value in [0, 1] specifying transparency of the block fill color (0=transparent, 1=opaque).

border_size Thickness of the border to draw around the block, in millimeters.

border_color Character string; color for the border around the block. NA means no border.

Columns in `elements(x, type="hvrules")`

hvrule descriptors:

id Character string that uniquely identifies the hvrule. The format is `<block>_<side>`.

direction Character string, either "hrule" for a horizontal rule or "vrule" for a vertical rule.

block The ID of the block along the side of which the hvrule runs.

side Which side of the block the hvrule runs along, "top", "bottom", "left", or "right".

adjacent_blocks Character string containing the IDs of blocks that are adjacent to `block` on the same side as the hvrule. (I.e., blocks that are separated from `block` by the hvrule.) Block IDs within the string are separated by semicolons. If there are no adjacent blocks the string will be empty ("").

arow1, arow2, acol1, acol2 Location of the hvrule with respect to the augmented row-column grid of the table. An hrule is inserted between table rows, and therefore `arow1` and `arow2` are the same and equal to a half-integer. For example, an hrule inserted between rows 3 and 4 has `arow1 = arow2 = 3.5`. `acol1` and `acol2` for the hrule are integers indicating the range of columns that it spans. Analogously, a vrule is inserted between table columns, so `acol1` and `acol2` are identical and equal to a half-integer, while `arow1` and `arow2` are integers that indicate the range of rows spanned by the vrule.

enabled Logical indicating whether the hvrule is to be displayed when the table is plotted. If FALSE the hvrule is ignored when laying out the table or determining its size.

Graphical properties for hvrules. Values for these are assigned by a *style*, either a default style (see `?tablesggOpt`) or user-specified.

linetype Integer indicating the type of line to display. 1 = solid line; 2 = dashed; 3 = dotted. (See the documentation of `lty` in `?par` for the full set of choices.) A line type of 0 means no line will be drawn, so the hvrule just inserts empty space between table rows or columns.

size Thickness of the line, in millimeters.

color Character string; the color of the line.

alpha Numeric value in [0, 1] specifying transparency of the line color (0=transparent, 1=opaque).

space The width (for a vertical rule) or height (for a horizontal rule) of the rectangle inserted between columns or rows by the hvrule. (A line, if any, is drawn within this rectangle.) This is the amount of space the rule adds to the width or height of the table, in millimeters.

fill Character string; color used to fill the rectangle containing the hvrule. NA means the region is not colored.

fill_alpha Numeric value in [0, 1] specifying transparency of the fill color (0=transparent, 1=opaque).

Value

A data frame with one row per element. Columns include element descriptors and graphical properties assigned to each element. The row names of the data frame will be the element ID's.

See Also

tablesggOpt, styleObj, ids

Examples

```
plt <- plot(iris2_tab)
str(elements(plt, type="entry"))
str(elements(plt, type="block"))  # 0 rows, none are enabled for display
str(elements(plt, type="block", enabledOnly=FALSE))
str(elements(plt, type="hvrule"))
```

---

element_block *Specify Visual Properties for Table Blocks*

Description

Specify a set of graphical properties that can be used to highlight a rectangular block of table cells.

Usage

```
element_block(fill=NULL, fill_alpha=NULL, border_size=NULL,
    border_colour=NULL, border_color=NULL, enabled=NULL,
    inherit.blank=FALSE)
```

Arguments

fill, fill_alpha, border_size, border_color	Scalar graphical properties for the table blocks. These will be passed to ggplot2::geom_rect under the names fill, alpha, size, and colour, respectively.
border_colour	Alias for border_color.
enabled	Logical scalar, controlling whether the block is displayed (TRUE) or not (FALSE).
inherit.blank	Ignored.

Details

This function is modeled on the `element_*` functions used in `ggplot2` to specify graphical properties in themes. It is primarily used to create the value on the right-hand side of an assignment involving the `props<-` group of setter functions.

To display a table block means to highlight its rectangular region with shading (as specified by `fill`, a color, and `fill_alpha`) and/or a border (with thickness specified by `border_size`, in mm, and color `border_color`). Blocks are drawn before entries and hvrules, so the latter are drawn "on top" of blocks and will not be hidden.

Quantitative property `border_size` may be specified using the `ggplot2` function `rel()`. This function indicates that the value is to be interpreted as a multiplier to be applied to whatever the current value of the property is. For example `border_size=rel(1.2)` specifies that the thickness of the border around a block is to be increased by 20% from its current value.

Value

An object of S3 classes `element_block` and `element`.

See Also

`element_entry`, `element_hvrule`; `elements` for more detail about the available graphical properties; `props<-`, `propsa<-`, `propsd<-`.

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data")
props(plt, id="rowhead") <- element_block(fill="lightblue")
props(plt, id="colblock/C/2/1") <- element_block(border_color="red")
      # (Default fill color for blocks is 'gray85')
plt
```

---

element_entry *Specify Visual Properties for Table Entries and their Cells*

Description

Specify a set of graphical properties that can be used to display table entries and the cells that contain them.

Usage

```
element_entry(text=NULL, family=NULL, fontface=NULL, colour=NULL,
    alpha=NULL, size=NULL, hjust=NULL, vjust=NULL, angle=NULL,
    lineheight=NULL, color=NULL, hpad=NULL, vpad=NULL, fill=NULL,
    fill_alpha=NULL, border_size=NULL, border_colour=NULL,
    border_color=NULL, minwidth=NULL, maxwidth=NULL, enabled=NULL,
    textspec=NULL, inherit.blank=FALSE)
```

Arguments

`text`	Scalar character string, the text to be displayed for the entries.
`family, fontface, color, alpha, size, hjust, vjust, angle, lineheight`	Scalar values for the graphical properties that are used to display the text of table entries. Values will be passed to `ggplot2::geom_text`.
`hpad, vpad`	Amount of blank space to add on the left and right (`hpad`) and top and bottom (`vpad`) of the entry text, in millimeters. This is to keep entry text from being too close to the cell borders.
`fill, fill_alpha, border_size, border_color`	Scalar graphical properties for the _cells_ that contain table entries. These will be passed to `ggplot2::geom_rect` under the names `fill`, `alpha`, `size`, and `colour`, respectively.
`colour, border_colour`	Aliases for `color`, `border_color`.
`minwidth, maxwidth`	Minimum, maximum width for cell(s) spanned by an entry. Positive values represent absolute sizes in millimeters, and should include any padding. Negative values are interpreted as multiples of the natural width of the entry text, without padding. Use values of `-1` for `minwidth` and `Inf` for `maxwidth` to prevent text wrapping. Values closer to 0 encourage wrapping. See `?elements` for more information and restrictions on these properties.
`enabled`	Logical scalar, controlling whether the entry is displayed (TRUE) or not (FALSE). This applies to both the entry text and the cells that contain it.
`textspec`	Character string indicating any special features of the text in `text`. Currently supported values are "plain", "plotmath", and "markdown". "plotmath" indicates that the entry text contains mathematical notation as described in `?plotmath`. "markdown" means the text may contain markdown or HTML tags to control its appearance; this requires the `ggtext` package.
`inherit.blank`	Ignored.

Details

This function is modeled on the `element_*` functions used in `ggplot2` to specify graphical properties in themes. It is primarily used to create the value on the right-hand side of an assignment involving the `props<-` group of setter functions.

The text content of an entry is perhaps not strictly a graphical property, but it is convenient to have an easy way to modify it. Note that like all other properties, the provided value must be a single value (character string), not a longer vector of values.

Justification of text within a cell is specified by properties `hjust` and `vjust`. Their interpretation is with respect to the boundaries of the cell: 0 means justification toward the left/top of the cell, 1 means toward the right/bottom, and 0.5 means centered. This is different from the interpretation in `?ggplot2::geom_text`, where the justification is with respect to an (x, y) point. Note that padding (`hpad`, `vpad`) is added _after_ justification, so for example `hjust=0` will put the text at a distance `hpad` from the left border of its cell.

Quantitative properties (`size`, `lineheight`, `hpad`, etc) may be specified using the `ggplot2` function `rel()`. This function indicates that the value is to be interpreted as a multiplier to be applied to

whatever the current value of the property is. For example `lineheight=rel(1.2)` specifies that the lineheight property of an entry is to be increased by 20% from its current value.

Value

An object of S3 classes `element_entry` and `element`.

See Also

`element_hvrule`, `element_block`, `element_refmark`; `elements` for more detail about the available graphical properties; `props<-`, `propsa<-`, `propsd<-`.

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data",
            subtitle="Shown with default graphical properties")
plt

props(plt, id="body") <- element_entry(fontface=3, fill="gray")
props(plt, id="subtitle,1") <-
      element_entry(text="Entry properties changed by 'props<-'",
                    fill="gray", color="red")
plt
```

`element_hvrule` *Specify Visual Properties for Table Rules*

Description

Specify a set of graphical properties that can be used to display horizontal or vertical rules (hvrules) in a table.

Usage

```
element_hvrule(colour=NULL, alpha=NULL, linetype=NULL, size=NULL,
    fill=NULL, fill_alpha=NULL, space=NULL, color=NULL, enabled=NULL,
    inherit.blank=FALSE)
```

Arguments

`color, alpha, linetype, size`
: Scalar values for the graphical properties that are used to display the horizontal or vertical line of an hvrule. Values will be passed to `ggplot2::geom_segment`.

`fill, fill_alpha, space`
: Scalar values for the graphical properties of the (long, thin) rectangle that encloses an hvrule. `space` is the height (for a horizontal rule) or width (for a vertical rule) of the rectangle, in millimeters. `fill` and `fill_alpha` are passed to `ggplot2::geom_rect` as arguments `fill` and `alpha` respectively.

colour	Alias for color.
enabled	Logical scalar, controlling whether the hvrule is displayed (TRUE) or not (FALSE). This applies to both the actual rule (line) and the rectangle containing it.
inherit.blank	Ignored.

Details

This function is modeled on the element_* functions used in ggplot2 to specify graphical properties in themes. It is primarily used to create the value on the right-hand side of an assignment involving the props<- group of setter functions.

A horizontal or vertical rule (hvrule) is actually drawn as long, narrow rectangle, with a line centered inside it. The narrowness of the rectangle, and thus how much space the hvrule adds to the table, is controlled by space. The thickness of the line inside the rectangle is controlled by size. Setting linetype to 0 means no line will be drawn, but the enclosing rectangle will be. In that way hvrules can be used to insert extra blank space between rows or columns of a table.

Quantitative properties size and space may be specified using the ggplot2 function rel(). This function indicates that the value is to be interpreted as a multiplier to be applied to whatever the current value of the property is. For example space=rel(1.2) specifies that the space property of an entry is to be increased by 20% from its current value.

Value

An object of S3 classes element_hvrule and element.

See Also

element_entry, element_block; elements for more detail about the available graphical properties; props<-, propsa<-, propsd<-.

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data")
plt

# Enable the vertical rule between rowhead and body, and set its
# properties:
props(plt, id="rowhead_right") <- element_hvrule(enabled=TRUE, linetype=1,
                                                 color="black", space=10)
plt
# Change the properties of all enabled hvrules:
propsd(plt, subset=(enabled)) <- element_hvrule(color="red")
plt
```

element_refmark *Specify a Reference Mark that can be Added to Table Entries*

Description

Specify a reference mark (a symbol placed before or after entry text to indicate cross-references; e.g. for footnotes) that can be added to entries in a table.

Usage

```
element_refmark(mark=NULL, side=NULL, raise, ..., inherit.blank=FALSE)
```

Arguments

mark	Character string containing the character(s) to be used as the reference mark.
side	Character string indicating where the reference mark is to be placed: "before" or "after" the entry text.
raise	Logical scalar. If TRUE, the reference mark will be displayed as a superscript, using plotmath. The default is TRUE except for asterisk marks, since that character is already raised relative to other characters.
...	Additional arguments passed to element_entry. These can be used to set other graphical properties of table entries at the same time as setting the reference mark. However it is an error to set text or textspec to anything other than NULL, since they are used internally by this function.
inherit.blank	Ignored.

Details

This function is modeled on the element_* functions used in ggplot2 to specify graphical properties in themes. It is primarily used to create the value on the right-hand side of an assignment involving the props<- group of setter functions.

Value

An object of S3 classes element_refmark and element. If any arguments are specified in ..., they are passed to element_entry and the resulting object is attached as attribute extra.

See Also

addRefmark for a different way to add reference marks, and for more information about reference marks in general. element_entry, props<-, propsa<-, propsd<-.

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data",
            foot="sd = standard deviation")
props(plt, id="foot") <- element_refmark(mark="*", side="before")
# Add a reference mark to just the first appearance of 'sd' in the row header:
propsa(plt, arows=arow(plt, hpath=c("setosa", "sd")), acols=2) <-
       element_refmark(mark="*", side="after")
plt
```

---

ids *Get the Identifier Strings for Parts or Elements of a Table*

Description

Get the unique identifier strings for elements of a plotted table, or for parts of a `textTable`.

Usage

```
ids(x, type, enabledOnly=TRUE)
```

Arguments

`x`	A `textTable` or a plotted table (`pltdTable`) object.
`type`	Character scalar indicating the type of elements to get ID's for. May be "entry", "block", or "hvrule" for a plotted table, or "part" for a `textTable`. Optional for a `textTable`.
`enabledOnly`	Logical scalar. If TRUE, ID's are returned only for elements that are currently enabled in `x`. (Ignored for a `textTable`.)

Details

A plotted table (`pltdTable` object) has three types of elements: entries, blocks, and hvrules. Entries are the text strings (and associated properties) displayed in table cells. Blocks are rectangular sets of contiguous table cells. And hvrules are spacers, with or without a visible line (or "rule"), used to separate or group table rows and columns.

Each element has an ID string, unique within an element type, and this function returns a vector of those strings.

A `textTable` has parts ("title", "subtitle", "rowhead", etc.), and this function just returns the vector of the part ID's.

Value

A character vector of identifiers.

See Also

elements, which returns a data frame with full information about each element of a plotted table.

Examples

```
ttbl <- textTable(iris2_tab)
# Just the names of the standard table parts:
ids(ttbl)

ptbl <- plot(ttbl)
# ID's of all the blocks defined for the table:
ids(ptbl, type="block", enabledOnly=FALSE)
# ID's of the blocks that are enabled for display (by default, none):
ids(ptbl, type="block", enabledOnly=TRUE)
```

---

iris2 *A Reshaped Version of Anderson's Iris Data*

Description

This is the same as R's built-in `iris` data set, but reshaped into "long" format. It contains four measurements for 50 flowers from each of three species of iris.

Usage

```
iris2
```

Format

A data frame with 600 observations on the following 5 variables.

`plant` a numeric vector, numbering the flowers from 1 to 150

`Species` a factor with levels `setosa` `versicolor` `virginica`

`flower_part` a factor with levels `Sepal` `Petal`

`direction` a factor with levels `Length` `Width`

`value` a numeric vector, the measured value, in centimeters

Source

- Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188.
- The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

Examples

```
data(iris2)
str(iris2)
```

iris2_tab *Table of Summary Statistics for Anderson's Iris Data*

Description

A table of means and standard deviations of the four measurements per iris plant, by species.

Usage

```
iris2_tab
```

Format

A `tabular` object as produced by version 0.9.6 of the **tables** package by Duncan Murdoch (`https://CRAN.R-project.org/p`
The table was produced with the following code, starting from the `iris2` data frame:

```
iris2_tab <- tables::tabular(
                     Species*Heading()*value*Format(digits=2)*(mean + sd) ~
                     Heading("Flower part")*flower_part*Heading()*direction,
                     data=iris2)
```

Source

- Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188.
- The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

See Also

iris2

Examples

```
if (requireNamespace("tables", quietly=TRUE)) {
  print(iris2_tab)  # uses print method for 'tabular' objects
}
plot(iris2_tab)
```

mtcars_xtab *Table of Data from Motor Trend Magazine*

Description

This is a table of selected observations and variables from the data frame `mtcars` from package **datasets**.

Usage

```
mtcars_xtab
```

Format

An `xtableList` object as produced by version 1.8-4 of the **xtable** package (`https://CRAN.R-project.org/package=xtabl`
The table was produced with the following code:

```
data("mtcars", package="datasets")
mtcars <- mtcars[, 1:6]
mtcarsList <- split(mtcars, f = mtcars$cyl)
### Reduce the size of the list elements
mtcarsList[[1]] <- mtcarsList[[1]][1,]
mtcarsList[[2]] <- mtcarsList[[2]][1:2,]
mtcarsList[[3]] <- mtcarsList[[3]][1:3,]
attr(mtcarsList, "subheadings") <- paste0("Number of cylinders = ",
                                          names(mtcarsList))
attr(mtcarsList, "message") <- c("Line 1 of Message", "Line 2 of Message")
mtcars_xtab <- xtable::xtableList(mtcarsList, digits = c(0,2,0,0,0,1,2),
                                  caption = "Caption to List")
```

Source

- Henderson and Velleman (1981), Building multiple regression models interactively. *Biometrics*, **37**, 391-411.

See Also

`mtcars`

Examples

```
ttbl <- textTable(mtcars_xtab)
plot(ttbl, rowheadInside=TRUE)
```

plot.tabular *A Plot Method for* tabular *Objects*

Description

A plot method for tabular objects, which represent a 2D data summary table. The table is displayed using ggplot2 graphics.

Usage

```
## S3 method for class 'tabular'
plot(x, ...)
```

Arguments

x	An object of class tabular, representing a 2D data summary table, as produced by the tables package by Duncan Murdoch.
...	Additional arguments passed to format.tabular or plot.textTable. See the documentation for those functions.

Details

This function is a simple wrapper that first converts x to a textTable object, and then plots that object.

Value

An object of S3 class pltdTable, inheriting from ggplot. See the documentation for plot.textTable for more information.

See Also

plot.textTable, format.tabular, textTable.tabular

plot.textTable *A Plot Method for* texttable *Objects*

Description

A plot method for textTable objects, displaying tables using ggplot2 graphics.

Usage

```
## S3 method for class 'textTable'
plot(x, title=NULL, subtitle=NULL, foot=NULL, rowheadLabels=NULL,
    entryStyle=tablesggOpt("entryStyle"),
    hvruleStyle=tablesggOpt("hvruleStyle"),
    blockStyle=tablesggOpt("blockStyle"), scale=1.0, mergeRuns=c(TRUE,
    TRUE), rowheadInside=FALSE, rowgroupSize=0,
    plot.margin=tablesggOpt("plot.margin"), sizeAdjust=c(1.0, 1.0), ...)
```

Arguments

`x`	A `textTable` object, containing a table.
`title, subtitle, foot`	Optional character vectors of annotation for the table. NULL means to leave the current annotation unchanged (the default); `character(0)` means to omit/remove it.
`rowheadLabels`	Optional character vector or 1-row matrix specifying labels for the row header columns of the table. NULL means to leave the current value unchanged (the default); `character(0)` means to omit/remove it.
`entryStyle, hvruleStyle, blockStyle`	Optional `styleObj` objects. These describe the graphical properties (color, size, font, line type, etc) to be used for displaying table entries, horizontal/vertical rules, or blocks, respectively. See `?styleObj` for details. If omitted, default styles from `tablesggOpt()` will be used.
`scale`	Numeric multiplier used to increase or decrease the displayed size of the table, with all elements scaled proportionately. If it has length two, the first element applies to table entries and blocks, and the second to hvrules.
`mergeRuns`	Two-element logical vector, the first element applying to row headers, the second to column headers. If TRUE (the default) then header cells that contain runs of the same value will be merged into a single entry that spans multiple rows or columns.
`rowheadInside`	Logical scalar. If TRUE, the outermost layer of row headers is moved inside the table by using its levels to divide the table into groups of rows, with each group labeled by their level.
`rowgroupSize`	Numeric scalar. If greater than 0, consecutive rows of the table body will be grouped into sets of this size, and extra space may be inserted between groups to improve readability. See DETAILS.
`plot.margin`	Numeric vector of length 4, giving the amount of padding to be added outside the top, right, bottom, and left sides of the table, in millimeters. The default is taken from `tablesggOpt()`.
`sizeAdjust`	Two-element numeric vector containing multipliers to adjust the calculated absolute size of table text. The first element is applied to the horizontal dimension and the second to the vertical dimension. (It should not be necessary to change this.)
`...`	Other arguments, ignored with a warning. (Included for compatibility with the `plot` generic.)

Details

textTable objects are the fundamental structure for representing tables in the tablesgg package. This function creates a publication-quality plot of such a table that can be displayed on any of R's graphics devices. Plotting is based on the ggplot2 graphics system, and the resulting plot object can be saved or manipulated in the usual ggplot way.

A feature of the plotted table produced by this function is that it has a well-defined physical size, in millimeters, given by the attribute size_mm. This natural size is determined by the graphical properties specified with arguments entryStyle and hvruleStyle, and can be rescaled up or down with the scale argument. There is a special print (display) method for pltdTable objects that ensures that the table is displayed at the correct size, independent of the size of the graphics device on which it is drawn.

The plotted table can be modified by using the props<- set of functions to update graphical properties of selected table entries, hvrules, or blocks. This includes disabling them (so that they are excluded from the plot), or re-enabling disabled elements. For broader changes there is an update method for pltdTable objects to change styles or plot scaling. See update.pltdTable for more information.

Value

An object of S3 class pltdTable, inheriting from ggplot.

The object has attributes plot.margin and sizeAdjust (equal to the arguments), and size_mm (the width and height of the plot, in millimeters). size_mm is calculated after applying scale and sizeAdjust, and includes both the table and any margins specified by plot.margin.

The object also has attributes colBoundaries and rowBoundaries giving the coordinates of the boundaries between columns and between rows, again in millimeters, and after applying scale and sizeAdjust.

The object also has attribute prTable. This a "plot-ready" version of the table, after applying the arguments provided to this function to x, but before processing by ggplot. It is included to allow convenient updating of the display properties of the table via props<- functions.

See Also

textTable, styleObj, tablesggOpt, styles_pkg, print.pltdTable, props<-, update.pltdTable

Examples

```
# Start with a 'textTable':
ttbl <- textTable(iris2_tab)
# Default display:
plot(ttbl)
# Add annotation:
plot(ttbl, title="The iris data", subtitle="Summary statistics by species",
     foot="sd = standard deviation")
# Smaller version:
plot(ttbl, title="The iris data", subtitle="Summary statistics by species",
     foot="sd = standard deviation", scale=0.8)
# Use a more "spread out" style for table entries:
plot(ttbl, entryStyle=styles_pkg$entryStyle_pkg_2)
```

```
# Internal row header labels:
plot(ttbl, rowheadInside=TRUE)
# Show effect of 'plot.margin' by putting a box around the table:
# -- default
plt <- plot(ttbl) +
       ggplot2::theme(plot.background=ggplot2::element_rect(color="blue", size=1))
plt
# -- wider margin
plt <- plot(ttbl, plot.margin=c(15, 15, 15, 15)) +
       ggplot2::theme(plot.background=ggplot2::element_rect(color="blue", size=1))
plt

# Data frame listing with rows in groups of 5:
plot(textTable(head(iris2, 15)), rowgroupSize=5)
```

`pltdSize` *Width and Height of a* `pltdtable` *Object*

Description

Width and height of a `pltdTable` object.

Usage

```
pltdSize(x, units=c("mm", "inches", "cm"))
```

Arguments

`x`	An object of class `pltdTable`, the result of plotting a `textTable` object.
`units`	String specifying the units in which size is to be returned. May be abbreviated.

Details

This function returns the size, after any scaling, of a plotted table, in physical units. The default units are millimeters to be consistent with other dimensions used in plotting tables. Inches may be useful because that is what R's graphics device functions use.

The size of a plot can depend slightly on the graphics device used to create or render it. The device name used internally to create the plot is included as an attribute of the returned value.

Value

Two-element numeric vector containing the (width, height) of the plot. It has attributes `units` and `device`.

See Also

`plot.textTable`

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data")
pltdSize(plt)  # width, height in millimeters

# Open a graphics device just the right size to hold the table:
sz <- pltdSize(plt, units="in")
dev.new(width=sz[1], height=sz[2], noRStudioGD=TRUE)
plt
```

---

print.pltdTable *Print (Draw) a* pltdtable *Object*

Description

Print method for pltdTable objects, to display the table on the currently active graphics device. It draws the table at its natural size, as determined by the font size and dimensions specified by the styles used to create the table, and after applying any scale factors.

Usage

```
## S3 method for class 'pltdTable'
print(x, scale=NULL, newpage=TRUE, position=c(0.5, 0.5),
    vpx=grid::unit(0.5, "npc"), vpy=grid::unit(0.5, "npc"),
    just="center", ...)
```

Arguments

x	A pltdTable object, representing a table.
scale	Optional numeric multiplier used to increase or decrease the displayed size of the table, relative to the natural size implied by its styles. The default is to keep the current scaling in x.
newpage	Whether to draw the table on the current page of the graphics device, or on a new blank page.
position	Two-element numeric or character vector specifying the horizontal and vertical position of the table on the page. A value of 0 means left/bottom justification, 1 means right/top justification, and intermediate values shift the table linearly between those limits. If a character vector, valid values are "left", "center", "right" for horizontal position, and "bottom", "center", "top" for vertical position. The default is to center the table on the page.
vpx, vpy, just	Alternatives to position to specify the position of the table on the page. These are passed to grid::viewport. (vpx is passed as argument x, and vpy as argument y.) Ignored with a warning if position is specified.
...	Optional additional arguments passed to grid::viewport.

Details

The purpose of a special print method for `pltdTable` objects is to set a particular viewport size, so that the table is displayed at its natural size, adjusted for `scale`. Once the viewport is set, the table is drawn using the usual `ggplot` print method.

Note that scaling of table size is not cumulative. If `x` has already been scaled (say, by a factor of 0.8), and argument `scale` is set to 0.9, then the table will be displayed at 0.9 times its natural size, not 0.9*0.8 = 0.72.

The default is that the table is drawn centered in the current graphics viewport (usually the whole graphics device surface). This can be changed using either the `position` or the `just`, `vpx` and `vpy` arguments. `grid::viewport` uses the latter four numbers to specify position; see its documentation for details. The `position` argument simplifies this to use just two numbers, each in [0, 1]. It assumes one rarely wants to have parts of the table outside the boundary of the page, so 0 corresponds to putting the table snug against the left or bottom edge, 1 corresponds to putting it against the right or top edge, and intermediate values simply linearly interpolate between those limits.

Value

`x`, invisibly.

See Also

`plot.textTable`, `grid::viewport`, `ggplot2::print.ggplot`

Examples

```
# Start with different ways of arranging the Titanic data:
data(Titanic, package="datasets")
ftbl1 <- ftable(Titanic, row.vars=c("Class", "Sex"), col.vars="Survived")
ftbl2 <- ftable(Titanic, row.vars=c("Age", "Sex", "Survived"),
                col.vars="Class")
ftbl3 <- ftable(Titanic, row.vars=c("Sex", "Class"),
                col.vars=c("Age", "Survived"))
plt1 <- plot(textTable(ftbl1))
plt2 <- plot(textTable(ftbl2))
plt3 <- plot(textTable(ftbl3))
plt4 <- plot(textTable(ftbl3), rowheadInside=TRUE)

# Center plots in the four quadrants of the page:
print(plt1, vpx=0.25, vpy=0.75)
print(plt2, vpx=0.75, vpy=0.75, newpage=FALSE)
print(plt3, vpx=0.25, vpy=0.25, newpage=FALSE)
print(plt4, vpx=0.75, vpy=0.25, newpage=FALSE)

# Single plot at different sizes, pushed to corners of the page:
print(plt2, scale=0.8, position=c("left", "top"))
print(plt2, scale=1.2, position=c("right", "bottom"), newpage=FALSE)
```

`props<-` *Update Graphical Properties for Selected Table Elements*

Description

Update the graphical properties for selected table entries, hvrules, or blocks in a plotted table.

Usage

```
props(x, id=NULL, regex=NULL, setEnabled=TRUE, mustMatch=TRUE,
    ...) <- value
```

Arguments

`x`	A `pltdTable` object containing a plotted table.
`id`	Optional character vector of the ID's of the elements (or of table parts containing the elements) to be modified.
`regex`	Optional character string containing a regular expression. This will be used to find table entries whose text matches the regular expression. (Only valid when `value` is an `element_entry` or `element_refmark` object.)
`setEnabled`	Logical scalar. If TRUE then any element whose properties are updated by this function will have its `enabled` value set to TRUE (and thus will be displayed in a plot). `enabled` will not be changed for elements that are not updated. If `setEnabled` is FALSE, `enabled` is not changed for any elements.
`mustMatch`	Logical scalar. If TRUE, any strings in argument `id` that do not match an element or part ID in `x` will be treated as an error.
`...`	Additional arguments passed to `grepl` when `regex` is used to select table entries.
`value`	An `element_entry`, `element_refmark`, `element_hvrule`, or `element_block` object that contains the new values for graphical properties. See DETAILS.

Details

There are three similar functions that can be used to modify the graphical properties of table elements: `props<-`, `propsa<-`, and `propsd<-`. They differ only in how one specifies which elements are to be modified. `props<-` uses element or block ID's, or searches the text content of entries. `propsa<-` uses explicit row and column numbers within the augmented row-column grid. `propsd<-` uses the values of element descriptors (as described in `?elements`).

The type of elements that are updated is determined by `value`: if `value` is an `element_entry` or `element_refmark` object then entries are updated; if it is an `element_hvrule` object then hvrules are; if it is an `element_block` object then blocks are. See the documentation of those functions for the available properties. As an example, `element_entry(color="red", fontface=3, hpad=rel(0.8))` specifies that all the updated entries will be displayed in red italics, and padding on their left and right will be reduced to 80% of the current amount. Any graphical properties not mentioned in the call that creates the `element_*` object are left unchanged.

There are two special properties: enabled and textspec (the latter only for entries). enabled is a logical scalar. If enabled is set to FALSE the selected elements will not be displayed, and disabled entries/hvrules will not be allocated any space in the plotted table. textspec is a character string, one of "plain", "plotmath", or "markdown". If "plotmath", entry text will be treated as an expression that allows display of mathematical symbols and notation, including subscripts and superscripts (see ?plotmath). If "markdown", entry text may contain markdown or HTML tags that affect the appearance of the displayed text. (This requires the ggtext package, and only a subset of HTML tags is supported. See the ggtext package documentation for more information.)

element_refmark is used to add a reference mark to the selected entries. (A reference mark is a symbol placed before or after entry text to indicate cross-references; e.g. for footnotes.) It may also update any of the graphical properties accepted by element_entry, except textspec and text.

Arguments id and regex indicate which elements are to be updated. When value indicates that table entries are to be modified, id may specify the ID's of individual entries or the ID's of table parts or blocks. In the latter case, all entries completely or partially contained in the parts or blocks are selected. Thus, for example, specifying id="table" will update every entry in the table, while id="body" will update only entries in the table body. Argument regex is only valid for modifying table entries; grepl is used to identify the entries whose text matches the regex pattern. If both id and regex are specified, then only entries selected by both are modified.

When value indicates that hvrules are to be modified, id should contain the ID's of individual hvrules and/or table blocks. In the latter case, any hvrule associated with a listed block (that is, the hvrule was defined as running along one of the sides of the block) will be updated.

When value indicates that blocks are to be modified, id should contain the ID's of individual table blocks.

See ?elements for the format of element ID strings.

This function overrides graphical properties in x that may have been set by a style. Therefore the value of style_row is set to NA for any elements whose properties are updated by this function.

Value

An object like x, with updated graphical properties for the selected elements.

See Also

element_entry, element_refmark, element_hvrule, element_block, ids, propsa<-, propsd<-

Examples

```
ttbl <- textTable(iris2_tab, title="The iris data",
                  subtitle=c("Summary statistics by species",
                             "A second subtitle line"),
                  foot="sd = standard deviation")
plt <- plot(ttbl)
# Change properties of elements:
props(plt, id="body") <- element_entry(fontface=3, fill="gray85")
# This may include changing text:
props(plt, id="subtitle,2") <- element_entry(text="Properties changed by 'props<-'",
                                             fill="gray85")
# Use property 'enabled' to control whether an element is displayed:
```

```
props(plt, id="rowhead_and_body_bottom") <- element_hvrule(enabled=FALSE)
plt

# Add reference marks to entries with abbreviation "sd":
plt <- plot(ttbl)
props(plt, regex="^sd$") <- element_refmark(mark="*", side="after")
props(plt, regex="^sd =") <- element_refmark(mark="*", side="before")
plt
# If both 'id' and 'regex' are specified only the intersection is modified:
plt <- plot(ttbl)
props(plt, regex="^sd$", id="rowblock/B/2/1") <-
  element_refmark(mark="*", side="after")
props(plt, regex="^sd =") <- element_refmark(mark="*", side="before")
plt
```

---

propsa<- *Update Graphical Properties for Selected Table Elements*

Description

Update the graphical properties of table elements in selected rows and columns of a plotted table.

Usage

```
propsa(x, arows=NULL, acols=NULL, setEnabled=TRUE) <- value
```

Arguments

x	A pltdTable object containing a plotted table.
arows, acols	Numeric vectors of row and column numbers in the table's augmented row-column grid. The default value of NULL means all rows or columns. Use half-integer values to refer to the locations of horizontal or vertical rules running between rows or columns.
setEnabled	Logical scalar. If TRUE then any element whose properties are updated by this function will have its enabled value set to TRUE (and thus will be displayed in a plot). enabled will not be changed for elements that are not updated. If setEnabled is FALSE, enabled is not changed for any elements.
value	An element_entry, element_refmark, element_hvrule, or element_block object that contains the new values for graphical properties.

Details

There are three similar functions that can be used to modify the graphical properties of table elements: props<-, propsa<-, and propsd<-. They differ only in how one specifies which elements are to be modified. props<- uses element or block ID, or searches the text content of entries.

propsa<- uses explicit row and column numbers within the augmented row-column grid. propsd<- uses the values of element descriptors (as described in ?elements).

The type of elements that are updated is determined by value: if value is an element_entry or element_refmark object then entries are updated; if it is an element_hvrule object then hvrules are; if it is an element_block object then blocks are. See the documentation for props<- for discussion of their use.

For this function the selected elements are those which are (a) completely contained within the rows listed in arows and the columns listed in acols; and (b) of a type (entry, block, or hvrule) corresponding to the class of value. For (a), arows and acols are treated as _sets_ of row/column numbers, not _ranges_. For example, a horizontal rule running between rows 3 and 4 will not be selected by arows=c(3, 4); the value 3.5 must be included in arows.

The helper functions arow and acol provide a partial bridge between this function and props<-. They return row and column numbers associated with particular element IDs, or with specified values of the row and column headers. For example, propsa(plt, arows=arow(plt, id="body"), acols=acol(plt, id="body")) is equivalent to props(plt, id="body"). See the examples below for cases where the combination of propsa<- and arow/acol is more convenient than props<- alone.

This function overrides graphical properties in x that may have been set by a style. Therefore the value of style_row is set to NA for any elements whose properties are updated by this function.

Value

An object like x, with updated graphical properties for the selected elements.

See Also

element_entry, element_refmark, element_hvrule, element_block, arow, acol, props<-, propsd<-

Examples

```
plt <- plot(iris2_tab, title="Summary statistics for the iris data")

plt2 <- plt
# Change the title to italics:
propsa(plt2, arows=1) <- element_entry(fontface=3)
# Change the vertical rule between row header and body from empty to
# a solid line:
propsa(plt2, acols=2.5) <- element_hvrule(linetype=1, color="black",
                                          size=0.5)
# Put all the mean values in bold face:
propsa(plt2, arows=arow(plt2, hpath=c(NA, "mean")),
       acols=acol(plt2, id="body")) <- element_entry(fontface=2)

plt3 <- plt
# Use shading to highlight the rows for the 'versicolor' species:
propsa(plt3, arows=arow(plt3, hpath=c("versicolor"))) <-
       element_block(fill="gray85")
# Compare tables before and after modification:
```

```
print(plt, vpx=0.25, vpy=0.75)
print(plt2, vpx=0.75, vpy=0.75, newpage=FALSE)
print(plt3, vpx=0.5, vpy=0.25, newpage=FALSE)

# Striping every other row in a data frame listing (must include row names):
data(mtcars, package="datasets")
plt <- plot(textTable(head(mtcars, 10),
                      title="Partial listing of the 'mtcars' data frame"))
ar <- arow(plt, id="body")
propsa(plt, arows=ar[ar %% 2 == 0]) <- element_block(fill="gray85")
plt
```

`propsd<-` *Update Graphical Properties for Selected Table Elements*

Description

Update the graphical properties for table elements selected based on the values of element descriptors.

Usage

```
propsd(x, subset=NULL, setEnabled=TRUE) <- value
```

Arguments

`x`	A `pltdTable` object containing a plotted table.
`subset`	An expression that when evaluated gives a logical vector indicating which elements should be updated; missing values are taken as FALSE. See DETAILS.
`setEnabled`	Logical scalar. If TRUE then any element whose properties are updated by this function will have its `enabled` value set to TRUE (and thus will be displayed in a plot). `enabled` will not be changed for elements that are not updated. If `setEnabled` is FALSE, `enabled` is not changed for any elements.
`value`	An `element_entry`, `element_refmark`, `element_hvrule`, or `element_block` object that contains the new values for graphical properties.

Details

There are three similar functions that can be used to modify the graphical properties of table elements: `props<-`, `propsa<-`, and `propsd<-`. They differ only in how one specifies which elements are to be modified. `props<-` uses element or block ID's, or searches the text content of entries. `propsa<-` uses explicit row and column numbers within the augmented row-column grid. `propsd<-` uses the values of element descriptors (as described in `?elements`).

The type of elements that are updated is determined by `value`: if `value` is an `element_entry` or `element_refmark` object then entries are updated; if it is an `element_hvrule` object then hvrules

are; if it is an element_block object then blocks are. See the documentation for props<- for discussion of their use.

Internally entries, hvrules, and blocks are represented as data frames, with one row per element, and columns describing their content, position, and structural role in the table. (See ?elements for a list of the descriptor columns for each type of element.) Argument subset is an expression involving those columns that evaluates to a logical vector; the elements for which this vector is TRUE will be selected. (NA in the logical vector is treated as FALSE.) Thus the subset argument works in the same way as R's built-in subset function to select rows from a data frame. For example subset=(part=="body" & type=="numeric") will update graphical properties for all entries in the table body that have been tagged as representing numbers. And subset=(direction=="hrule") will update graphical properties for horizontal rules but not vertical rules.

To update every element (of the type implied by value) in the table, set subset=TRUE. To update just the currently enabled elements, set subset=enabled.

Note: Standard processing generates hvrules for all four sides of every block in the table. Since generally one doesn't want to display the blocks themselves nor most of those hvrules, they are all disabled by default. Therefore if using subset=TRUE when updating hvrules or blocks, also set setEnabled to FALSE to avoid enabling and displaying them all.

This function overrides graphical properties in x that may have been set by a style. Therefore the value of style_row is set to NA for any elements whose graphical properties are updated by this function.

Value

An object like x, with updated graphical properties for the selected elements.

See Also

element_entry, element_refmark, element_hvrule, element_block, elements, props<-, propsa<-

Examples

```
ttbl <- textTable(iris2_tab, title="The iris data",
                  subtitle=c("Summary statistics by species"),
                  foot="sd = standard deviation")
plt <- plot(ttbl)
propsd(plt, subset=(enabled)) <- element_hvrule(color="red")
propsd(plt, subset=(part == "colhead" & headlayer == 1)) <-
  element_entry(angle=90, hjust=0.5, vjust=0.5)
plt
```

`styleObj` *Create a Style Object*

Description

Create a style object that can be used to assign graphical properties to table elements (entries, hvrules, or blocks). The properties are used when displaying the elements. Styles allow assignment of graphical properties to be based on element descriptors.

Usage

```
styleObj(x, type, match_columns=character(0))
```

Arguments

`x`	Data frame with (a) column(s) that specify patterns to be used to identify and select a subset of table elements; and (b) columns specifying the graphical properties to be used for elements in the selected subset. See DETAILS. May also be a string with the path to a `.csv` file that will be read to create such a data frame.
`type`	Character string, one of "entry", "hvrule", or "block". This specifies the type of element to which the style will apply.
`match_columns`	Optional character vector (possibly empty) with the names of element descriptors that are required in order to evaluate the subset selector expressions in `x`. See DETAILS.

Details

A style specifies graphical properties to be used in displaying one of the element types in a table (i.e., entries, hvrules, or blocks). A style is similar to a theme in `ggplot2` in that it can be applied to any table, not just a particular table.

A `styleObj` object is a data frame. Each row can be thought of as a pattern plus a set of graphical properties. Table elements that are to be styled are compared to the patterns. If the pattern in a style row matches a table element, the graphical properties in that row are assigned to the element. If more than one style row matches an element, the properties from the last matching row override the earlier ones.

The graphical property columns that must be present in a `styleObj` data frame are described in `?elements`.

Specification of style patterns and how they are matched to elements is similar for table entries and blocks, and is described first. The process for hvrules is more complicated and is described second. It will be easier to follow the descriptions if one also looks at an example, such as `View(styles_pkg$entryStyle_pkg_1)` and `View(styles_pkg$hvruleStyle_pkg_1)`.

Style specification and matching: Entry and block styles

First note that table entries and blocks internally are stored in objects that are themselves data frames, with one row per element. (These data frames can be accessed using the `elements` function.) Columns include element descriptors such as the table part associated with the element, its

position in the table, whether the element spans multiple rows or columns, and other information. See `?elements` for lists of the standard descriptors.

In styles for table entries and blocks, the pattern part of the `styleObj` object consists of a single column named `condition`. `condition` should contain character strings that can be interpreted as expressions involving the element descriptors mentioned in the previous paragraph. Each `condition` expression, when evaluated within an `entries` or `blocks` data frame, should produce a logical vector with one value per element. (Vectors of length 1 are recycled to the necessary length.) Examples of such strings are `part == "rowhead" & multirow` for entries and `type == "colblock" & subtype == "A" & headlayer > 1` for blocks.

Elements for which the `condition` expression in a style row evaluates to TRUE are considered to match that row of the style, and are assigned the graphical properties in that row.

An NA value (or equivalently an empty string) as a style row's `condition` is treated specially: it matches _any_ element. The row's graphical properties will be applied to all elements, unless overridden by a later style row.

Style specification and matching: hvrule styles

The creation and styling of hvrules is closely tied to table blocks: by default, four hvrules are created for each block, one running along each side. (They are initially disabled.) Style specification for hvrules is more complicated than for table blocks because hvrules effectively *separate* blocks. Therefore one may want their appearance to depend on characteristics of the blocks on *both* sides of the hvrule.

Similar to entries and blocks, hvrules are represented internally as a data frame with one row per hvrule. Columns include: `block`, the ID of the block that generated the hvrule; `side`, the side of `block` along which the hvrule runs ("top", "bottom", "left", or "right"); and `adjacent_blocks`, a string listing the ID's of all the blocks adjacent to `block` on the same side as the hvrule. That is, the hvrule separates `block` and the blocks in `adjacent_blocks`. Note that `adjacent_blocks` may be empty.

In styles for hvrules, the pattern part of the `styleObj` object consists of three columns: `block_condition`, `side`, and `adjacent_condition`. `side` is one of "top", "bottom", "left" or "right". `block_condition` and `adjacent_condition` are like the `condition` column for block styles: they should contain character strings that can be interpreted as expressions involving block descriptors. Each expression will be evaluated within the data frame of blocks that generated the hvrules. (Not the data frame containing the hvrules themselves.) It should produce a logical vector with one element per block; if the value is TRUE for a block, the block satisfies that expression.

An hvrule matches a given style row if (a) its generating block satisfies the style row's `block_condition`; (b) they have the same value of `side`; and (c) one or more of the hvrule's `adjacent_blocks` satisfies the style row's `adjacent_condition`.

Any of `block_condition`, `side`, and `adjacent_condition` in a style row may also be set to NA (or equivalently, to an empty string). In that case the corresponding criterion (a), (b), or (c) is considered to be satisfied for all hvrules, and so does not limit matches. Note that setting `adjacent_condition` to NA is the only way to satisfy criterion (c) if an hvrule's `adjacent_blocks` is empty. In all other cases, an empty `adjacent_blocks` will never satisfy criterion (c).

Value

An object of S3 classes `styleObj` and `data.frame`. It will have the same number of rows and all the columns in `x`.

The object will have attributes `element_type` and `match_columns`, equal to the corresponding arguments.

See Also

`styles_pkg` contains predefined styles provided by the package. They can be examined as illustrations of how styles are specified, or edited to create new styles. `elements` lists the descriptors and graphical properties available for each element type.

Examples

```
# Built-in default styles:
if (interactive()) {
  View(styles_pkg$entryStyle_pkg_1)
  View(styles_pkg$blockStyle_pkg_1)
  View(styles_pkg$hvruleStyle_pkg_1)
}
```

---

styles_pkg *Built-In Styles for Table Elements*

Description

A list that contains pre-defined styles for table entries, blocks, and hvrules. Styles are used to assign graphical properties to elements, and thus control the appearance of a table when it is displayed. In addition to the package-provided styles in this list, users can modify or create new styles to customize their tables.

Usage

```
styles_pkg
```

Format

The list has the following components, each a `styleObj` object:

`entryStyle_pkg_base`, `blockStyle_pkg_base`, `hvruleStyle_pkg_base` Minimal styles that assign the same graphical properties to all elements.

`entryStyle_pkg_1`, `blockStyle_pkg_1`, `hvruleStyle_pkg_1` The default styles used by the package.

`entryStyle_pkg_2` Similar to `entryStyle_pkg_1`, but with `hpad` and `vpad` about 50 percent larger, to give a more spacious layout of table entries.

`entryStyle_pkg_null`, `blockStyle_pkg_null`, `hvruleStyle_pkg_null` "Null" styles designed to not match any element, and thus not assign graphical properties to any element. Using the null style for hvrules is a way to disable all hvrules when the plot is created (rather than afterwards using a `props<-` function).

See Also

`styleObj`, `tablesggOpt`

Examples

```
names(styles_pkg)  # built-in styles
str(styles_pkg$entryStyle_pkg_1)
```

---

summary.pltdTable *Summarize the Dimensions and Options of a Plotted Table*

Description

Summarize the dimensions and display options of a plotted table.

Usage

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

Arguments

`object`	A `pltdTable` object, a plotted 2D data summary table.
`...`	Additional arguments, ignored with a warning. (Included for compatibility with the generic.)

Details

There is a print method for objects of the returned class.

Value

An object of S3 class `summary.pltdTable`. It is a list with components

`adim`	Dimensions of the augmented row-column grid for the table. See `?adim` for details about this grid.
`parts`	Data frame with one row for each table part, giving the dimensions of the part, in columns `nr`, `nc`.
`mergeRuns`, `rowheadInside`, `rowgroupSize`, `scale`, `plot.margin`, `sizeAdjust`	Display options used by the table. See `plot.textTable` for their meaning.

See Also

adim, plot.textTable

Examples

```
ttbl <- textTable(iris2_tab, title="Summary statistics for the iris data")
plt <- plot(ttbl, rowheadInside=TRUE)
summary(plt)
```

---

summary.textTable *Summarize the Dimensions of a Table and Its Parts*

Description

Summarize the dimensions of a table and its parts.

Usage

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

Arguments

object	A textTable object, representing a 2D data summary table.
...	Additional arguments, ignored with a warning. (Included for compatibility with the generic.)

Details

There is a print method for objects of the returned class.

Value

An object of S3 class summary.textTable. It is a list with components

adim	Dimensions of the augmented row-column grid for the table. See ?adim for details about this grid.
parts	Data frame with one row for each table part, giving the dimensions of the part, in columns nr, nc.

See Also

adim, textTable

Examples

```
ttbl <- textTable(iris2_tab, title="Summary statistics for the iris data")
summary(ttbl)
```

tablesggOpt *Get or Reset Package Options*

Description

Get the values of package options, or reset all options to their "factory-fresh" defaults.

Usage

```
tablesggOpt(x=NULL, reset=FALSE)
```

Arguments

x	Character string with the name of a single package option, or NULL.
reset	Logical scalar. If TRUE, all options will be reset to their initial, "factory-fresh" values.

Details

The user can change option values using the tablesggSetOpt function. The new values will stay in effect for the rest of the R session or until they are changed again by the user.

Value

If x is the name of a single package option, the value of that option. Otherwise, a named list with the current values of all package options. In both cases the result is after resetting if reset is TRUE.

The result is invisible if reset is TRUE.

The available options are documented in ?tablesggSetOpt.

See Also

tablesggSetOpt, styles_pkg

Examples

```
# See names of available options:
names(tablesggOpt())
# The current value of option 'plot.margin':
tablesggOpt("plot.margin")
```

tablesggSetOpt *Set the Values of Package Options*

Description

Set the values of package options.

Usage

```
tablesggSetOpt(...)
```

Arguments

...	Arguments in `tag = value` form, or a list of tagged values. The tags must come from the names of package options described in DETAILS below. The named options will be set to the values provided.

Details

The new option values persist until the end of the R session or until they are changed by another call to this function or to `tablesggOpt(reset=TRUE)`.

The options that may be set are:

entryStyle A `styleObj` object, with element type `entry`. This is the default style for table entries.

blockStyle A `styleObj` object, with element type `block`. This is the default style for blocks.

hvruleStyle A `styleObj` object, with element type `hvrule`. This is the default style for hvrules.

plot.margin A numeric vector of length 4, containing the amount of empty space to add around the four sides of a plotted table, in millimeters. The order of sides is top, right, bottom, left.

allowMarkdown Logical scalar. If TRUE then text for table entries is allowed to contain markdown and HTML tags to control its appearance. TRUE is valid only if package `ggtext` is available.

allowWrap Logical scalar. If TRUE then automatic wrapping of text for table entries is allowed. TRUE is valid only if packages `ggtext` and `quadprog` are available.

Facilities to handle markdown and automatic wrapping of entry text are provided by Claus Wilke's `ggtext` package (https://CRAN.R-project.org/package=ggtext). Therefore `allowMarkdown` and `allowWrap` can be set to TRUE only when that package has been installed. Note that only a subset of HTML tags are available.

Value

A list with the old values of the named options, invisibly.

See Also

`tablesggOpt` to get current values of options, or to reset options to their "factory-fresh" setting. `styles_pkg` for the set of package-provided table styles.

Examples

```
oldopt <- tablesggOpt()
tablesggOpt(reset=TRUE)
plt1 <- plot(iris2_tab, title="Factory-fresh default styles")
# Set new default style for table entries:
tablesggSetOpt(entryStyle=styles_pkg$entryStyle_pkg_2)
plt2 <- plot(iris2_tab, title="Changed default entry style")
# Compare:
print(plt1, vpy=0.75)
print(plt2, vpy=0.25, newpage=FALSE)

# Change the values of multiple options:
tablesggSetOpt(list(hvruleStyle=styles_pkg$hvruleStyle_pkg_base,
                    plot.margin=c(5, 5, 5, 5)))
# ... plot some tables using the new defaults ...
# Restore the old options:
tablesggSetOpt(oldopt)
identical(tablesggOpt(), oldopt)
```

`textTable` *Create a Structure Representing a 2D Table*

Description

Create a structure representing the content and organization of a 2D table: table body, row and column headers, and annotation. All table cells are formatted as character strings. This is an S3 generic.

Usage

```
textTable(x, ...)
```

Arguments

`x`	Object to be formatted as a 2D table.
`...`	Additional arguments passed to specific methods.

Details

`textTable` objects are the fundamental structure used to represent table _content_ and _organization_ in the `tablesgg` package.

Components `body`, `rowhead`, `rowheadLabels`, `colhead`, `title`, `subtitle`, and `foot` correspond to the table _parts_ with those names. Empty parts should be of type character: either a 0-length vector or a matrix with one or both dimensions equal to 0, depending on the component.

Character strings representing table content may be prefixed with either "MATH_" or "MKDN_". The former indicates the string is to be interpreted as a `plotmath` expression, the latter that the string contains markdown or HTML tags.

Components partdim, rowhier, and colhier are automatically derived from the other components whenever a textTable is created or updated.

See Appendix A of the package vignette for more information about writing textTable methods.

Value

An object with S3 class textTable. This is a list with components:

- **body** — Character matrix containing the body of the table.
- **rowhead** — Character matrix with the same number of rows as the table body, containing row headers for the table. Row headers are displayed as a set of columns to the left of the table body. May be empty (0 columns).
- **rowheadLabels** — Character matrix with as many columns as rowhead and at most one row, specifying labels for the rowhead columns. May be empty (0 rows).
- **colhead** — Character matrix with the same number of columns as the table body, containing column headers for the table. Column headers are displayed as a set of rows above the table body. If rowheadLabels are present, colhead must have at least one row, but otherwise it may be empty (0 rows).
- **title, subtitle, foot** — Character vectors providing annotation for the table. May be empty (length 0).
- **partdim** — Numeric matrix with one row per table part (i.e., the components listed above), and columns:
  - **nr, nc:** Number of rows, columns in the part (nc equal to NA for annotation parts).
  - **arow1, arow2, acol1, acol2:** First and last rows, first and last columns occupied by the part within the table's augmented row-column grid. arow* should be NA if nr is 0, acol* should be NA if nc is 0.
- **rowhier, colhier** — Lists describing the hierarchical structure of row and column headers, respectively. Each list has one component per header layer (column of rowhead, row of colhead), in order from outermost layer to innermost. In turn, each of these components is a data frame with one row per node in the hierarchy at that layer.

Components body, rowhead, and colhead will each have an attribute type. For body this will be a character matrix with the same dimensions, containing an arbitrary string describing the type of value represented in each cell (e.g., "numeric"), or NA. For rowhead and colhead, it will be a character vector with length equal to the number of layers of headers (i.e., number of columns in rowhead, number of rows in colhead), again containing a string describing the type of values in each layer, or NA.

Components body, rowhead, rowheadLabels, colhead, title, subtitle, and foot will each have an attribute justification. It will be a character matrix or vector of the same size and shape as the component. Values "l", "c", "r" specify left, centered, and right horizontal justification of text, respectively, for the corresponding table entry. Value NA means that the type of justification is not specified–it will be set by the entry style used when plotting the table.

See Also

Specific methods for creating textTable's from other objects.

`textTable.data.frame` *Create a* `texttable` *as a Simple Listing of a Data Frame*

Description

Create a `textTable` object representing a simple listing of a data frame.

Usage

```
## S3 method for class 'data.frame'
textTable(x, title=character(0), subtitle=character(0),
    foot=character(0), row.names="", na="NA", ...)
```

Arguments

`x`	A data frame.
`title, subtitle, foot`	Optional character vectors providing annotation for the table. May be empty (i.e., `character(0)`, the default).
`row.names`	A logical scalar or a character string. If FALSE, the row names of `x` are not included in the table. If TRUE, the row names are included as row headers, with a row header label of "row.names". If a character string, row names are included as row headers, and if the string is not empty, it is used as the row header label.
`na`	Character string used to represent missing values (NAs) in the body of the table.
`...`	Additional arguments passed to `format.data.frame`.

Details

This function processes a data frame into a table that is simply a listing of the data. There is one row in the body of the table per observation in `x`, and one column per variable in `x`. There is at most one layer of row headers (depending on argument `row.names`), and exactly one layer of column headers (the variable names in `x`).

Value

An object with S3 class `textTable`. The body of the table will contain the values of the data frame variables, after formatting `x` with `format(x, ...)`. The variable names will be used as the column header, and if `row.names` is not FALSE, the row names will form the row header.

Examples

```
data(iris, package="datasets")
ttbl <- textTable(head(iris, 10), row.names="Obs. #",
                  title=c("The iris data", "(First 10 observations)"))
summary(ttbl)
plot(ttbl)
```

`textTable.ftable` *Create a* `texttable` *from an* `ftable`

Description

Create a `textTable` object representing a flattened multiway contingency table.

Usage

```
## S3 method for class 'ftable'
textTable(x, colheadLabels=c("layers", "none", "paste"), sep=": ",
    title=character(0), subtitle=character(0), foot=character(0), ...)
```

Arguments

`x`	An `ftable` object, as produced by R's `ftable` function, representing a flattened multiway contingency table.
`colheadLabels`	Character scalar; how to display names of column header variables. "none" means to not display them. "layers" (the default) means to display them as additional column header layers (so each header variable occupies two rows instead of one). "paste" means to paste the variable name in front of each of its values, separated by `sep`.
`sep`	Character scalar; string that separates a variable name from its values when `colheadLabels` is "paste".
`title, subtitle, foot`	Optional character vectors providing annotation for the table. May be empty (i.e., `character(0)`, the default).
`...`	Ignored, with a warning. (Included for compatibility with the generic.)

Value

An object with S3 class `textTable`. See the documentation for the generic for details about its structure.

See Also

`ftable`, `format.ftable`

Examples

```
# From examples in '?ftable':
data(Titanic, package="datasets")
ft <- ftable(Titanic, row.vars = 1:2, col.vars = "Survived")
ttbl <- textTable(ft, title="Plotting an 'ftable'")
plot(ttbl)

data(mtcars, package="datasets")
```

```
ft <- ftable(mtcars$cyl, mtcars$vs, mtcars$am, mtcars$gear, row.vars = c(2, 4),
             dnn = c("Cylinders", "V/S", "Transmission", "Gears"))
ttbl <- textTable(ft, colheadLabels="none")
plt1 <- plot(ttbl, title="Plotting an 'ftable'",
             subtitle="No colheadLabels")
ttbl <- textTable(ft, colheadLabels="layers")
plt2 <- plot(ttbl, title="Plotting an 'ftable'",
             subtitle="colheadLabels = 'layers'")
ttbl <- textTable(ft, colheadLabels="paste")
plt3 <- plot(ttbl, title="Plotting an 'ftable'",
             subtitle="colheadLabels = 'paste'")
print(plt1, position=c("left", "top"))
print(plt2, position=c("left", "center"), newpage=FALSE)
print(plt3, position=c("left", "bottom"), newpage=FALSE)
```

`textTable.matrix` *Create a* `texttable` *from a Matrix*

Description

Create a `textTable` object from a matrix.

Usage

```
## S3 method for class 'matrix'
textTable(x, rcnames=c(TRUE, TRUE), title=character(0),
    subtitle=character(0), foot=character(0), na="NA", ...)
```

Arguments

`x`	A matrix.
`rcnames`	A logical or character vector of length 2. The first element applies to rows, the second to columns. If FALSE, row/column names are not included. If TRUE and `x` has row/column names, they are included as a row/column header. Further, if the row/column dimension itself has a non-empty name, it is included as an additional, outer row/column header layer. A character string is treated the same as TRUE, except that the string is used as the dimension name (and thus an empty string will not create an outer header layer).
`title, subtitle, foot`	Optional character vectors providing annotation for the table. May be empty (i.e., `character(0)`, the default).
`na`	Character string used to represent missing values (NAs) in the body of the table.
`...`	Additional arguments passed to `format(x, ...)`.

Value

An object with S3 class `textTable`. The body of the table will contain the matrix values, after formatting `x` with `format(x,...)`. Row and column names may be included as headers, depending on argument `rcnames`.

Examples

```
data(iris, package="datasets")
mat <- data.matrix(subset(iris, Species == "setosa")[, 1:4])
ttbl <- textTable(cor(mat), digits=3, title="Correlations for setosa irises")
summary(ttbl)
plt <- plot(ttbl)
# Make hvrules invisible:
propsd(plt, subset=enabled) <- element_hvrule(color=NA)
print(plt)
```

`textTable.table` *Create a* `texttable` *from a* `table` *or* `xtabs` *Object*

Description

Create a `textTable` object representing a flattened multiway contingency table.

Usage

```
## S3 method for class 'table'
textTable(x, colheadLabels=c("layers", "none", "paste"), sep=": ",
    title=character(0), subtitle=character(0), foot=character(0), ...)
```

Arguments

`x`	A `table` object, as produced by R's `table` or `xtabs` functions, representing a multiway contingency table.
`colheadLabels`	Character scalar; how to display names of column header variables. "none" means to not display them. "layers" (the default) means to display them as additional column header layers (so each header variable occupies two rows instead of one). "paste" means to paste the variable name in front of each of its values, separated by `sep`.
`sep`	Character scalar; string that separates a variable name from its values when `colheadLabels` is "paste".
`title, subtitle, foot`	Optional character vectors providing annotation for the table. May be empty (i.e., `character(0)`, the default).
`...`	Additional arguments passed to `ftable`, to convert `x` to an `ftable` object.

Details

This function simply converts `x` to an `ftable` (flattened multiway contingency table), then applies the corresponding `textTable` method to that object.

It also works for `xtabs` objects since they inherit from `table`.

Value

An object with S3 class `textTable`. See the documentation for the generic for details about its structure.

See Also

`ftable`, `xtabs`

Examples

```
# UCBAdmissions is a contingency table in array form ('table' object).
data(UCBAdmissions, package="datasets")
ttbl <- textTable(UCBAdmissions)
plot(ttbl, title=c("Plotting a 'table' object:", "UCB Admissions data"))

# Method also works for 'xtabs' since they inherit from 'table' (example
# from '?xtabs'):
data(warpbreaks, package="datasets")
warpbreaks$replicate <- rep_len(1:9, 54)
xt <- xtabs(breaks ~ wool + tension + replicate, data = warpbreaks)
ttbl <- textTable(xt, title="Plotting an 'xtabs' object (warpbreaks data)")
plot(ttbl)
```

`textTable.tabular` *Create a* `texttable` *from a* `tabular` *Object*

Description

Convert a `tabular` object, representing a 2D data summary table, into a `textTable` object, which can be plotted.

Usage

```
## S3 method for class 'tabular'
textTable(x, title=character(0), subtitle=character(0),
    foot=character(0), rowheadLabels=TRUE, ...)
```

Arguments

x	An object of class tabular, representing a 2D data summary table, as produced by the tables package.
title, subtitle, foot	Optional character vectors providing annotation for the table. May be empty (i.e., character(0), the default).
rowheadLabels	Character vector or logical scalar specifying labels for the row header columns of the table. FALSE or character(0) means no labels, TRUE will attempt to extract labels from x.
...	Additional arguments passed to format.tabular.

Details

tabular objects are produced by the tabular function in package tables. This function converts them to textTable objects to enable plotting. It can also add table annotation.

Row headers and column headers are derived from the rowLabels and colLabels attributes of x, respectively. It appears that tabular objects always have "rowLabels", "colLabels" and a body with non-zero dimensions (although this is not required for textTable objects in general). In addition, runs of duplicated values in rowLabels and colLabels are replaced by NA; the NAs are changed back to the original values by this function. The dropcells attribute is a character matrix matching the table body. If not NA, the value in dropcells is used to replace the cell content after formatting.

In the returned object, components body, rowhead, and colhead will each have an attribute type. For body the attribute is a character matrix containing a string describing the type of value represented in each cell of the table body; namely, the first element of the vector returned by function class() as applied to each element of x. For rowhead and colhead, type is a character vector with one element per header variable (i.e., per column of rowhead or row of colhead). Since tabular objects do not retain the classes of the variables that define row and column dimensions of a table, type will be set to NA.

Components of the returned object will also have an attribute justification. It will be a character matrix or vector of the same size and shape as the component; a value of NA means that the type of justification is not specified. Values for the table body and row and column headers will be taken from x. Values for table annotation will be NA.

Value

An object with S3 class textTable. See the documentation for the generic function for a description of the structure of this object.

See Also

tables::tabular, tables::format.tabular, plot.textTable

Examples

```
# 'iris2_tab' is a 'tabular' object created using 'tables::tabular'.
class(iris2_tab)
# 'tables' package provides a 'print' method for such objects:
```

```
if (requireNamespace("tables", quietly=TRUE)) {
print(iris2_tab)
}
# This package provides 'textTable' and 'plot' methods for such objects:
ttbl <- textTable(iris2_tab)
plot(ttbl)
# ... or just
#plot(iris2_tab)  # same
```

`textTable.xtable` *Create a* `texttable` *from an* `xtable` *Object*

Description

Create a `textTable` from an `xtable` object produced by the `xtable` package. The `textTable` can then be styled and plotted.

Usage

```
## S3 method for class 'xtable'
textTable(x, title, subtitle=character(0), foot=character(0),
    row.names="", na="", mathExponents=TRUE, ...)
```

Arguments

`x`	An `xtable` object as produced by the `xtable` package.
`title`	Optional character vector containing title lines for the table. May be empty (`character(0)`). The default is to use the first element of the `caption` attribute of `x`, if present.
`subtitle, foot`	Optional character vectors providing additional annotation for the table. May be empty (i.e., `character(0)`, the default).
`row.names`	A logical scalar or a character string. If FALSE, the row names of `x` are not included in the table. If TRUE, the row names are included as row headers, with a row header label of "row.names". If a character string, row names are included as row headers, and if the string is not empty, it is used as the row header label.
`na`	String to be used to represent missing values in `x`. The default value is the empty string "".
`mathExponents`	Logical scalar. If TRUE, then numerical values in `x` that are formatted into scientific notation (i.e., strings like "3.14e-02", specified by values of `e` or `E` in the `display` attribute of `x`) will be plotted in math style, with the power of 10 shown as a superscript.
`...`	Additional named arguments passed to `formatC`, when converting values in `x` to character strings. They must not include `digits` or `format`, which are specified within `x` itself.

Details

This function was designed based on the structure of objects produced by version 1.8-4 of the `xtable` package.

An `xtable` object is a data frame that contains the columns of the table and attributes that specify how those columns are to be formatted. This function uses those attributes to created formatted character strings for each table entry, and assembles them into a `textTable` object, which may then be styled and plotted.

Formatting is done by `formatC` using the `digits` and `display` attributes of `x`. The `align` attribute is used to set the `justification` attributes in the returned `textTable`. (Vertical rule characters, `|`, within `align` are ignored; use an `hvruleStyle` or the `addHvrule` function to insert vertical rules into the plotted table, as shown in the examples.)

Value

An object with S3 class `textTable`. See the documentation for the generic for details about its structure.

Examples

```
# 'tli_xtab' is an 'xtable' object created using 'xtable::xtable':
class(tli_xtab)
# This package provides a 'textTable' method for such objects:
ttbl <- textTable(tli_xtab)
plot(ttbl)

if (requireNamespace("xtable", quietly=TRUE)) withAutoprint({
  data(tli, package="xtable")

  # ANOVA table.
  fm1 <- aov(tlimth ~ sex + ethnicty + grade + disadvg, data = tli)
  plt1 <- plot(textTable(fm1.table <- xtable::xtable(fm1),
                         title="xtable: ANOVA table"))

  # Table of linear regression results.
  fm2 <- lm(tlimth ~ sex*ethnicty, data = tli)
  plt2 <- plot(textTable(fm2.table <- xtable::xtable(fm2),
                         title="xtable: Linear regression"))

  # Time series table.
  temp.ts <- ts(cumsum(1 + round(rnorm(100), 0)), start = c(1954, 7),
                frequency = 12)
  plt3 <- plot(textTable(xtable::xtable(temp.ts, digits = 0),
                         title="xtable: Time series"))

  # Math style for scientific notation.
  plt4 <- plot(textTable(xtable::xtable(data.frame(text = c("foo","bar"),
                                                   googols = c(10e10,50e10),
                                                   small = c(8e-24,7e-5),
                                                   row.names = c("A","B")),
                                        display = c("s","s","g","g")),
```

```
                      mathExponents = TRUE,
                    title=c("xtable:", "Math style for scientific notation")))
  print(plt1, position=c(0.1, 0.9))
  print(plt2, position=c(0.1, 0.5), newpage=FALSE)
  print(plt3, position=c(0.1, 0.1), newpage=FALSE)
  print(plt4, position=c(0.9, 0.9), newpage=FALSE)

  # By default vertical rules specified by '|' characters in 'align' are
  # ignored.  They can be added afterward using the 'addHvrule' function
  # as follows:
  tli.table <- xtable::xtable(tli[1:10, ])
  xtable::align(tli.table) <- "|rrl|l|lr|"
  plt <- plot(textTable(tli.table,
                        title="xtable: Vertical rules derived from 'align'"))
  pipe_posn <- which(unlist(strsplit(attr(tli.table, "align"), "")) == "|")
  vrule_acol <- pipe_posn - seq_along(pipe_posn) + 0.5
  for (ac in vrule_acol)  plt <- addHvrule(plt, direction="vrule", acols=ac,
                                           arows=arow(plt, "colhead_and_body"),
                                           props=element_hvrule(linetype=1,
                                                                color="black"))
  plt
})
```

`textTable.xtableList` *Create a* `texttable` *from an* `xtablelist` *Object*

Description

Create a `textTable` from an `xtableList` object produced by the `xtable` package. Such an object represents a set of subtables that are to be stacked into a single table, with "subheadings" separating the subtables. The `textTable` that is produced by this function uses the subheadings as an additional, outer row header layer. That layer can optionally be moved inside the table by setting `rowheadInside=TRUE` when the table is plotted.

Usage

```
## S3 method for class 'xtableList'
textTable(x, title, subtitle=character(0), foot, ...)
```

Arguments

`x`	An `xtableList` object as produced by the `xtable` package. All the subtables in `x` must have the same column names.
`title`	Optional character vector containing title lines for the table. May be empty (`character(0)`). The default is to use the first element of the `caption` attribute of `x`, if present.
`subtitle, foot`	Optional character vectors providing additional annotation for the table. May be empty (i.e., `character(0)`). The default for `foot` is the `message` attribute of `x`.

... Additional arguments passed to `textTable.xtable` for each of the subtables in `x`. See the documentation for that function.

Details

This function was designed based on the structure of objects produced by version 1.8-4 of the `xtable` package.

If `x` has a `message` attribute, it is used as the default value of the `foot` component of the returned `textTable`.

If components of `x` have no `subheading` attribute, then the subtables are simply stacked, with no additional row header layer to separate or distinguish them. It is an error if only some of the components have a `subheading` attribute.

Value

An object with S3 class `textTable`. See the documentation for the generic for details about its structure.

Examples

```
# 'mtcars_xtab' is an 'xtableList', following an example in the
# "listOfTablesGallery" vignette of the 'xtable' package.  (See '?mtcars_xtab'
# for the code to create it.)
plot(textTable(mtcars_xtab), rowheadInside=TRUE,
     title="Example of plotting an 'xtableList'",
     subtitle="(With 'rowheadInside=TRUE')")
```

---

tli_xtab *Table of Test Scores and Demographics for 20 Students*

Description

This is a table of the first 20 observations from the data frame `tli` from package **xtable**. The observations include demographic data and math scores, from the Texas Assessment of Academic Skills, for 20 students.

Usage

```
tli_xtab
```

Format

An `xtable` object as produced by version 1.8-4 of the **xtable** package (`https://CRAN.R-project.org/package=xtable`). The table was produced with the following code:

```
data("tli", package="xtable")
  tli_xtab <- xtable::xtable(tli[1:20, ])
  xtable::display(tli_xtab)[c(2,6)] <- "f"
  xtable::digits(tli_xtab) <- matrix(0:4, nrow = 20, ncol = ncol(tli)+1)
```

Source

- Texas Education Agency, <URL: http://www.tea.state.tx.us>

Examples

```
str(tli_xtab)
```

`update.pltdTable` *Update a* `pltdtable` *(Plotted Table) Object*

Description

Update a `pltdTable` (plotted table) object with new styles or scaling.

Usage

```
## S3 method for class 'pltdTable'
update(object, entryStyle=NULL, blockStyle=NULL, hvruleStyle=NULL,
    scale=NULL, plot.margin=attr(object, "plot.margin"),
    sizeAdjust=attr(object, "sizeAdjust"), ...)
```

Arguments

`object`	A `pltdTable` object, containing a plotted table.
`entryStyle, blockStyle, hvruleStyle`	Optional `styleObj` objects, specifying new styles for assigning graphical properties to table entries, blocks, or hvrules. The default value of NULL leaves the corresponding style of `object` unchanged.
`scale`	Optional numeric multiplier used to increase or decrease the displayed size of table elements, relative to the natural size implied by their (possibly updated) styles. If it has length two, the first element applies to entries and blocks, and the second to hvrules. The default is to use the existing scale value(s) in `object`.
`plot.margin, sizeAdjust`	See the documentation for `plot.textTable`. The default is to use the same values that were used to create `object`.
`...`	Ignored, with a warning. (Included for compatibility with the generic.)

Details

Updating a plotted table is limited to changing its style or scale–changes that do not affect the augmented row-column grid of the table. (See `adim` for a description of that grid.) For other changes, start with a `textTable` object, and edit it and/or replot it using different arguments (e.g., `rowheadInside`, `rowgroupSize`, `mergeRuns`, or annotation).

Updating does not change the `enabled` field for any entries, blocks, or existing hvrules.

When argument `hvruleStyle` is provided, hvrules are regenerated by applying the style to the blocks component of `object`. These new hvrules replace any existing hvrules with the same ID. However existing hvrules with other ID's are left unchanged.

Value

An object of S3 class `pltdTable`, inheriting from `ggplot`. See `plot.textTable` for details about this object.

See Also

`plot.textTable`, `styleObj`

Examples

```
# Plot using 'factory-fresh' entry style:
plt <- plot(textTable(iris2_tab), entryStyle=styles_pkg$entryStyle_pkg_1)
# Change to a generic style that uses the same graphical properties for
# all entries:
plt2 <- update(plt, entryStyle=styles_pkg$entryStyle_pkg_base)
plt2
# Also make the plot smaller:
plt3 <- update(plt2, scale=0.8)
plt3
```

---

update.textTable *Update a* texttable *Object*

Description

Update a `textTable` object with new annotation or rowheadLabels.

Usage

```
## S3 method for class 'textTable'
update(object, title=NULL, subtitle=NULL, foot=NULL,
    rowheadLabels=NULL, ...)
```

Arguments

`object`	A `textTable` object, representing a 2D table with all cells formatted as character strings.
`title, subtitle, foot`	Optional character vectors of annotation for the table. NULL means to leave the current annotation unchanged (the default); `character(0)` means to omit/remove it.
`rowheadLabels`	Optional character vector or 1-row matrix specifying labels for the row header columns of the table. NULL means to leave the current value unchanged (the default); `character(0)` means to omit/remove it.
`...`	Ignored, with a warning. (Present for compatibility with the generic.)

Details

To indicate that a string in `title`, `subtitle`, `foot`, or `rowheadLabels` is to be interpreted as a `plotmath` expression, prefix it with `MATH_`. To indicate that it contains markdown or HTML tags, prefix it with `MKDN_`.

Value

A `textTable` object with annotation set or changed based on the provided arguments.

See Also

`textTable`

Examples

```
ttbl <- textTable(iris2_tab, title="The iris data",
                  foot="sd = standard deviation")
# Change annotation:
ttbl <- update(ttbl, title=c("The iris data", "Summary statistics by species"),
               foot=character(0))
plot(ttbl)
# Change row header labels:
ttbl <- update(ttbl, rowheadLabels=c("Species", "Summary\nstatistic"))
plot(ttbl)
```

`[.textTable` *Extract a Subset of a* `texttable` *Object*

Description

Extract a subset of a `textTable` object, creating a new table with fewer and/or rearranged rows and columns.

Usage

```
## S3 method for class 'textTable'
x[i, j, drop=FALSE]
```

Arguments

x	An object of S3 class textTable, representing a 2D table.
i, j	Logical or numeric indexing arguments used as subscripts with respect to the augmented row-column grid of the table. See DETAILS.
drop	Ignored (always treated as FALSE).

Details

This function extracts, deletes, or rearranges subsets of the rows and columns of a table. It is similar to subsetting an ordinary matrix, but with restrictions required to ensure that the resulting object is still a valid textTable:

1. Indexing is with respect to the augmented row-column grid of the table, in which all parts of the table (body, headers, and annotation) are included. See ?textTable for a description of table parts, and ?adim for a description of the augmented grid. The summary method for a textTable shows the dimensions of each part.

2. The first index argument, i, cannot itself be a matrix.

3. Indexing cannot be used to move rows or columns between different parts of the table (e.g. between body and headers, or between headers and annotation).

Helper functions arow and acol can be used to get the augemented row and column numbers spanned by different table parts. See the examples.

Value

An object of S3 class textTable.

See Also

textTable, adim, arow, acol

Examples

```
ttbl <- textTable(iris2_tab)
plot(ttbl)

# Remove the first column header row ("Flower part"), and reverse the
# order of the "Sepal" and "Petal" sets of columns:
subttbl <- ttbl[-1, c(1,2,5,6,3,4)]
plot(subttbl)

# Use helper functions 'arow', 'acol' to specify indices based on
# table structure:
i <- arow(ttbl, "colhead")[1]  # row number of first column header row
j1 <- acol(ttbl, "rowhead")    # column numbers for row header
```

```
j2 <- acol(ttbl, "colhead")    # column numbers for column header
subttbl2 <- ttbl[-i, c(j1, j2[c(3,4,1,2)])]
identical(subttbl, subttbl2)
```

Index