Package ‘htmlTable’

March 4, 2015

Version 1.2
Date 2015-03-03
Title Advanced Tables for Markdown/HTML
Author Max Gordon <max@gforge.se>
Maintainer Max Gordon <max@gforge.se>
Description Tables with state-of-the-art layout elements such as
row spanners, column spanners, table spanners, zebra striping,
and more. While allowing advanced layout, the underlying CSS structure is
simple in order to maximize compatibility with wordprocessors such as
MS Word or LibreOffice.
The package also contains a few text formatting functions that help outputting
text compatible with HTML/LaTeX.
License GPL (>= 3)
URL http://gforge.se/packages/
BugReports https://github.com/gforge/htmlTable/issues
Biarch yes
Imports stringr, knitr (>= 1.6), magrittr (>= 1.5), methods
Suggests testthat, XML, xtable, ztable, Hmisc, reshape
Encoding UTF-8
NeedsCompilation no
VignetteBuilder knitr
Repository CRAN
Date/Publication 2015-03-04 00:24:39

R topics documented:

htmlTable .................................................. 2
SCB .......................................................... 7
tblNoLast ................................................... 8
tblNoNext ................................................... 9
**htmlTable**

### Description

This is a function for outputting a more advanced table than what `xtable`, `ztable`, or `knitr`'s `kable()` allows. It's aim is to provide the `Hmisc latex()` colgroup and rowgroup functions in HTML. The html-output is designed for maximum compatibility with LibreOffice/OpenOffice.

### Usage

```
htmlTable(x, ...)  
```

```
## Default S3 method:  
htmlTable(x, header, rnames, rowlabel, caption, tfoot, label,  
  rgroup, n.rgroup, cgroup, n.cgroup, tspanner, n.tspanner, total,  
  align = paste(rep("c", ncol(x)), collapse = ""),  
  align.header = paste(rep("c", ncol(x)), collapse = ""), align.cgroup,  
  css.rgroup = "font-weight: 900;", css.rgroup.sep = ",",  
  css.tspanner = "font-weight: 900; text-align: left;",  
  css.tspanner.sep = "border-top: 1px solid #BEBEBE;",  
  css.total = "border-top: 1px solid #BEBEBE; font-weight: 900;",  
  css.cell = ",", css.cgroup = ",", css.class = "gmisc_table",  
  pos.rowlabel = "bottom", pos.captions = "top", col.rgroup = "none",  
  col.columns = "none", padding.rgroup = "&nbsp;&nbsp;",  
  padding.tspanner = "", ctable = TRUE, compatibility = "LibreOffice",  
  cspan.rgroup = "all", ...)  
```

```
## S3 method for class 'htmlTable'  
n knit_print(x, ...)  
```

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

### Arguments

- **x**
  The matrix/data.frame with the data. For the `print` and `knit_print` it takes a string of the class `htmlTable` as `x` argument.

- **...**
  Passed on to `print.htmlTable` function and any argument except the `useViewer` will be passed on to the `cat` functions arguments.

- **header**
  A vector of character strings specifying column header, defaulting to `colnames(x)`
rnames
Default rownames are generated from `rownames(x)`. If you provide `FALSE` then it will skip the rownames. *Note:* For `data.frames` if you do `rownames(my_dataframe) <- NULL` it still has rownames. Thus you need to use `FALSE` if you want to suppress rownames for `data.frames`.

rowlabel
If the table has rownames or rnames, rowlabel is a character string containing the column heading for the rnames.

caption
Adds a table caption.

tfoot
Adds a table footer (uses the `<tfoot>` html element). The output is run through `txtMergeLines` simplifying the generation of multiple lines.

label
A text string representing a symbolic label for the table for referencing as an anchor. All you need to do is to reference the table, for instance `<a href="#anchor_name">see table 2</a>`. This is known as the element’s id attribute, i.e. table id, in HTML linguo, and should be unique id for an HTML element in contrast to the `css.class` element attribute.

rgroup
A vector of character strings containing headings for row groups. `n.rgroup` must be present when `rgroup` is given. The first `n.rgroup[1]` rows are sectioned off and `rgroup[1]` is used as a bold heading for them. The usual row dimnames (which must be present if `rgroup` is) are indented. The next `n.rgroup[2]` rows are treated likewise, etc. If you don’t want a row to be part of a row group then you just put "" for that row, remember to add the corresponding number of rows in `n.rgroup`.

n.rgroup
An integer vector giving the number of rows in each grouping. If `rgroup` is not specified, `n.rgroup` is just used to divide off blocks of rows by horizontal lines. If `rgroup` is given but `n.rgroup` is omitted, `n.rgroup` will default so that each row group contains the same number of rows.

cgroup
A vector or a matrix of character strings defining major column header. The default is to have none. These elements are also known as *column spanners*. If you want a column *not* to have a spanner then put that column as "". If you pass `cgroup` and `n.cgroup` as matrices you can have column spanners for several rows. See `cgroup` section below for details.

n.cgroup
An integer vector or matrix containing the number of columns for which each element in `cgroup` is a heading. For example, specify `cgroup=c("Major_1","Major_2"), n.cgroup=c(3,3)` if "Major_1" is to span columns 1-3 and "Major_2" is to span columns 4-6. `rowlabel` does not count in the column numbers. You can omit `n.cgroup` if all groups have the same number of columns.

tspanner
The table spanner is somewhat of a table header that you can use when you want to join different tables with the same columns.

n.tspanner
An integer vector with the number of rows in the original matrix that the table spanner should span.

total
The last row is sometimes a row total with a border on top and bold fonts. Set this to `TRUE` if you are interested in such a row. If you want a total row at the end of each table spanner you can set this to "tspanner".

align
A character strings specifying column alignments, defaulting to `paste(rep('c', ncol(x)), collapse='') to center. Valid alignments are l = left, c = center and r = right. You can also specify align\(\)='c\(\)c\(\)' and other LaTeX tabular formatting. If you want to set the
alignment of the rownames this string needst to be ncol(x) + 1, otherwise it automatically pads the string with a left alignment for the rownames.

align.header A character strings specifying alignment for column header, defaulting to centered, i.e. \texttt{paste(rep('c', ncol(x)), collapse='')}.

align.cgroup The justification of the cgroups

css.rgroup CSS style for the rgroup, if different styles are wanted for each of the rgroups you can just specify a vector with the number of elements

css.rgroup.sep The line between different rgroups. The line is set to the TR element of the lower rgroup, i.e. you have to set the border-top/padding-top etc to a line with the expected function. This is only used for rgroups that are printed. You can specify different separators if you give a vector of rgroup - 1 length (this is since the first rgroup doesn’t have a separator).

css.tspanner The CSS style for the table spanner

css.tspanner.sep The line between different spanners

css.total The css of the total row

css.cell The css.cell element allows you to add any possible CSS style to your table cells. If you provide a vector the vector it is assumed that the styles should be repeated throughout the columns. If you provide a matrix of the same size as your x argument. If have ncol(x) + 1 the first row will correspond to the rowname style. Correspondingly if the size is nrow(x) + 1 it is assummed that the first row is the header row.

css.cgroup The same as css.class but for cgroup formatting.

css.class The html CSS class for the table. This allows directing html formatting through \texttt{CSS} directly at all instances of that class. \textit{Note:} unfortunately the CSS is frequently ignored by word processors. This option is mostly inteded for web-presentations.

pos.rowlabel Where the rowlabel should be positioned. This value can be "top", "bottom", "header", or a integer between 1 and nrow(cgroup) + 1. The options "bottom" and "header" are the same, where the row label is presented at the same level as the header.

pos.caption Set to "bottom" to position a caption below the table instead of the default of "top".

col.rgroup Alternating colors (zebra striping/banded rows) for each rgroup; one or two colors is recommended and will be recycled.

col.columns Alternating colors for each column.

padding.rgroup Generally two non-breakings spaces, i.e. &nbsp;&nbsp;, but some journals only have a bold face for the rgroup and leaves the subelements unindented.

padding.tspanner The table spanner is usually without padding but you may specify padding similar to padding.rgroup and it will be added to all elements, including the rgroup elements. This allows for a 3-level hierarchy if needed.

ctable If the table should have a double top border or a single a’ la LaTeX ctable style
compatibility is default set to LibreOffice as some settings need to be in old html format as Libre Office can’t handle some commands such as the css caption-alignment. Note: this option is not yet fully implemented for all details, in the future I aim to generate a html-correct table and one that is aimed at Libre Office compatibility. Word-compatibility is difficult as Word ignores most settings and destroys all layout attempts (at least that is how my 2010 version behaves).

cspan.rgroup is the number of columns that an rgroup should span. It spans by default all columns but you may want to limit this if you have column colors that you want to retain.

useViewer If you are using RStudio there is a viewer that can render the table within that is invoked if in interactive mode. Set this to FALSE if you want to remove that functionality. You can also force the function to call a specific viewer by setting this to a viewer function, e.g. useViewer = utils::browseURL if you want to override the default RStudio viewer. Another option that does the same is to set the options(viewer = utils::browseURL) and it will default to that particular viewer (this is how RStudio decides on a viewer). Note: If you want to force all output to go through the cat() the set options(htmlTable.cat = TRUE).

Value

string Returns a string of class htmlTable

Multiple rows of column spanners cgroup

If you want to have a column spanner in multiple levels you can set the cgroup and n.cgroup arguments to matrices. If the different levels have different number of elements you need to set the ones that lack elements to NA. For instance cgroup = rbind(c("first", "second", NA), c("a", "b", "c")) and the corresponding n.cgroup would be n.cgroup = rbind(c(1, 2, NA), c(2, 1, 2)). for a table consisting of 5 columns. The "first" spans the first two columns, the "second" spans the last three columns, "a" spans the first two, "b" the middle column, and "c" the last two columns.

The add attribute to rgroup

You can now have an additional element at the rgroup level by specifying the att(rgroup, 'add'). The value can either be a vector or a list of the same length as the rgroup or a list/vector with names corresponding to integers within the rgroup span.

Important knitr-note

This function will only work with knitr outputting html, i.e. markdown mode. As the function returns raw html-code the compatibility with non-html formatting is limited, even with pandoc.

Thanks to the the knit_print and the asis_output the results='asis' is no longer needed except within for-loops. If you have a knitr-chunk with a for loop and use print() to produce raw html you must set the chunk option results='asis'. Note: the print-function relies on the interactive() function for determining if the output should be sent to a browser or to the terminal. In vignettes and other directly knitted documents you may need to either set useViewer = FALSE alternatively set options(htmlTable.cat = TRUE).
Table counter

If you set the option `table_counter` you will get a Table 1, 2, 3 etc before each table, just set `options(table_counter=TRUE)`. If you set it to a number then that number will correspond to the start of the table_counter. The `table_counter` option will also contain the number of the last table, this can be useful when referencing it in text. By setting the option `options(table_counter_str = "<b>Table %s:</b> ")` you can manipulate the counter table text that is added prior to the actual caption. Note, you should use the `sprintf` %s instead of %d as the software converts all numbers to characters for compatibility reasons. If you set `options(table_counter_roman = TRUE)` then the table counter will use Roman numerals instead of Arabic.

Possible issues

Note that when using complex cgroup alignments with multiple levels not every browser is able to handle this. For instance the RStudio webkit browser seems to have issues with this and a bug has been filed.

As the table uses html for rendering you need to be aware of that headers, rownames, and cell values should try respect this for optimal display. Browsers try to compensate and frequently the tables still turn out fine but it is not advised. Most importantly you should try to use `<tl; instead of < and &gt; instead of >. You can find a complete list of html characters here.

See Also

txtMergeLines, latex

Other table functions: tblNoLast; tblNoNext

Examples

# A simple output
output <- matrix(1:4, 
ncol=2, 
row.names = list("Row 1", "Row 2"), 
list("Column 1", "Column 2"))
htmlTable(output)

# An advanced output
output <-
matrix(ncol=6, nrow=8)
for (nr in 1:nrow(output)){
  for (nc in 1:ncol(output)){
    output[nr, nc] <-
    paste0(nr, ":", nc)
  }
}
htmlTable(output, align="r", 
header = paste(c("1st", "2nd", 
"3rd", "4th", 
"5th", "6th"), 
"hdr"),
rnames = paste(c("1st", "2nd", "3rd", paste0(4:8, "th")), "row"),
rgroup = paste("Group", LETTERS[1:3]),
n.rgroup = c(2,4, nrow(output) - 6),
cgroup = rbind(c("", "Column spanners", NA), c("", "Cgroup 1", "Cgroup 2\&dagger;"),
c(2,2,2)),
caption="Basic table with both column spanners (groups) and row groups",
tfoot="\&dagger; A table footer comment",
cspan.rgroup = 2,
col.columns = c(rep("none", 2), rep("#F5FBFF", 4)),
col.rgroup = c("none", "#F5FBFF"),
css.cell = "padding-left: .5em; padding-right: .2em;"

# See vignette("tables", package = "htmlTable")
# for more examples

---

**SCB**

**Average age in Sweden**

**Description**

For the vignettes there is a dataset downloaded by using the `get_pxweb_data()` call. The data is from SCB (Statistics Sweden) and downloaded using:

**Author(s)**

Max Gordon <max@gforge.se>

**References**

http://scb.se

**Examples**

```r
# Not run:
# The data was generated through downloading via the API
library(pxweb)

# Get the last 15 years of data (the data always lags 1 year)
current_year <- as.integer(format(Sys.Date(), "%Y")) - 1
SCB <- get_pxweb_data(
dims = list(Region = c('00', '01', '03', '25'),
            Kon = c('1', '2'),
            ContentsCode = c('BE0101G9'))
```

# Some cleaning was needed before use
SCB$region <- factor(substring(as.character(SCB$region), 4))
Swe_ltrs <- c("Å" = "&aring;",
              "Å" = "&Aring;",
              "ä" = "&auml;",
              "Ä" = "&Auml;",
              "ö" = "&ouml;",
              "Ö" = "&Ouml;")
for (i in 1:length(Swe_ltrs)){
  levels(SCB$region) <- gsub(names(Swe_ltrs)[i],
                           Swe_ltrs[i],
                           levels(SCB$region))
}

save(SCB, file = "data/SCB.rda")

## End(Not run)

---

### tblNoLast

**Gets the last table number**

#### Description

The function relies on `options("table_counter")` in order to keep track of the last number.

#### Usage

```r
tblNoLast(roman = getOption("table_counter_roman", FALSE))
```

#### Arguments

- **roman**
  
  Whether or not to use roman numbers instead of arabic. Can also be set through `options(table_caption_no_roman = TRUE)`

#### See Also

Other table functions: `htmlTable`, `htmlTable.default`, `knit_print.htmlTable`, `print.htmlTable`; `tblNoNext`

#### Examples

```r
org_opts <- options(table_counter=1)
tblNoLast()
options(org_opts)
```
**tblNoNext**

*Gets the next table number*

**Description**

The function relies on `options("table_counter")` in order to keep track of the last number.

**Usage**

```r
tblNoNext(roman = getOption("table_counter_roman", FALSE))
```

**Arguments**

- `roman` : Whether or not to use roman numbers instead of arabic. Can also be set through `options(table_caption_no_roman = TRUE)`

**See Also**

Other table functions: `htmlTable`, `htmlTable.default`, `knit_print.htmlTable`, `print.htmlTable`; `tblNoLast`

**Examples**

```r
org_opts <- options(table_counter=1)
tblNoNext()
options(org_opts)
```

---

**txtInt**

*SI or English formatting of an integer*

**Description**

English uses ',' between every 3 numbers while the SI format recommends a ' ' if \( x > 10^4 \). The scientific form 10e+? is furthermore avoided.

**Usage**

```r
txtInt(x, language = "en", html = TRUE, ...)
```

**Arguments**

- `x` : The integer variable
- `language` : The ISO-639-1 two-letter code for the language of interest. Currently only english is distinguished from the ISO format using a ' ´ as the separator.
- `html` : If the format is used in html context then the space should be a non-breaking space, &nbsp;
- `...` : Passed to `format`
Value

string

Examples

txtInt(123)
txtInt(1234)
txtInt(12345)
txtInt(123456)

txtMergeLines

A merges lines while preserving the line break for html/LaTeX

Description

This function helps you to do a multiline table header in both html and in LaTeX. In html this isn’t that tricky, you just use the <br /> command but in LaTeX I often find myself writing vbox/hbox stuff and therefore I’ve created this simple helper function

Usage

txtMergeLines(..., html = 5)

Arguments

... The lines that you want to be joined
html If HTML compatible output should be used. If FALSE it outputs LaTeX formatting. Note if you set this to 5 then the html5 version of br will be used: <br> otherwise it uses the <br /> that is compatible with the xhtml-formating.

Value

string

See Also

Other text formatters: txtpval; txtRound

Examples

txtMergeLines("hello", "world")
txtMergeLines("hello", "world", html=FALSE)
txtMergeLines("hello", "world", list("A list", "is OK"))
**txtPval**

*Formats the p-values*

---

**Description**

Gets formatted p-values. For instance you often want 0.1234 to be 0.12 while also having two values up until a limit, i.e. 0.01234 should be 0.012 while 0.001234 should be 0.001. Furthermore you want to have < 0.001 as it becomes ridiculous to report anything below that value.

**Usage**

```r
txtPval(pvalues, lim2dec = 10^-2, lim.sig = 10^-4, html = TRUE, ...)
```

**Arguments**

- `pvalues`: The p-values
- `lim2dec`: The limit for showing two decimals. E.g. the p-value may be 0.056 and we may want to keep the two decimals in order to emphasize the proximity to the all-mighty 0.05 p-value and set this to $10^{-2}$. This allows that a value of 0.0056 is rounded to 0.006 and this makes intuitive sense as the 0.0056 level as this is well below the 0.05 value and thus not as interesting to know the exact proximity to 0.05. Disclaimer: The 0.05-limit is really silly and debated, unfortunately it remains a standard and this package tries to adapt to the current standards in order to limit publication associated issues.
- `lim.sig`: The significance limit for the less than sign, i.e. the '<'
- `html`: If the less than sign should be < or &lt; as needed for html output.
- `...`: Currently only used for generating warnings of deprecated call parameters.

**Value**

`vector`

**See Also**

Other text formatters: `txtMergeLines`; `txtRound`

**Examples**

```r
txtPval(c(0.10234, 0.010234, 0.0010234, 0.00010234))
```
A convenient rounding function

Usage

```r
txtRound(x, digits, excl.cols, excl.rows, txt.NA = "", dec = ".")
```

Arguments

- `x`: The data.frame/matrix to be rounded
- `digits`: The number of digits to round each element to. If you provide a vector each element for corresponding columns.
- `excl.cols`: Rows to exclude from the rounding procedure. This can be either a number or regular expression.
- `excl.rows`: Columns to exclude from the rounding procedure. This can be either a number or regular expression.
- `txt.NA`: The string to exchange NA with
- `dec`: The decimal marker. If the text is in non-english decimal and string formatted you need to change this to the appropriate decimal indicator.

Value

`matrix/data.frame`

See Also

Other text formatters: `txtMergeLines; txtPval`

Examples

```r
mx <- matrix(c(1, 1.11, 1.25,
               2.50, 2.55, 2.45,
               3.2313, 3, pi),
             ncol = 3, byrow=TRUE)
txtRound(mx, 1)
```
Index

*Topic data
   SCB, 7

asis_output, 5

cat, 2, 5
colnames, 2

format, 9

get_pxweb_data, 7

htmlTable, 2, 8, 9
htmlTable.default, 8, 9

interactive, 5

kable, 2
knit_print, 5
knit_print.htmlTable, 8, 9
knit_print.htmlTable(htmlTable), 2

latex, 2, 6

options, 5

paste, 3, 4
print.htmlTable, 8, 9
print.htmlTable(htmlTable), 2

rownames, 3

SCB, 7
sprintf, 6

tblNoLast, 6, 8, 9
tblNoNext, 6, 8, 9
txtInt, 9
txtMergeLines, 3, 6, 10, 11, 12
txtPval, 10, 11, 12
txtRound, 10, 11, 12