Package ‘hwriterPlus’

July 2, 2014

Type Package

Title hwriterPlus: Extending the hwriter Package

Version 1.0-3

Date 2014-01-15

Author David Scott <`d.scott@auckland.ac.nz`>

Maintainer David Scott <`d.scott@auckland.ac.nz`>

Depends R (>= 2.10.0), hwriter

Imports TeachingDemos

Suggests xtable, RUnit

Description This package extends the package hwriter providing
facilities such as the inclusion of output from R, the
results of an R session, the display of mathematical
expressions using LaTeX notation and the incorporation of
SVG graphical objects. It uses MathJax to display
mathematical expressions, in contrast to R2HTML which uses ASCIIMathML.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-01-15 01:11:27

R topics documented:

`hwriterPlus-package` *hwriterPlus: Extending the hwriter Package*

Description

This extends the package hwriter providing facilities such as the inclusion of output from R, the results of an R session and the display of mathematical expressions using LaTeX notation.

Details

Package:	hwriterPlus
Type:	Package
Version:	1.0
Date:	2011-10-27
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

David Scott <d.scott@auckland.ac.nz>

See Also

See the package **hwriter**. Also the functions `as.latex` and `HTML.latex` from the package **R2HTML**.

Examples

```
### hwriter example: shows many table examples
hwriter:::showExample()

### Some examples of using hwriter to produce tags
### Text in a heading tag
hwrite("Some text", heading = 1)
```

```
### Text in a div tag
hwrite("Some text", div = TRUE)

### A link to an external website
hwrite("Auckland University", link = "http://www.auckland.ac.nz")

### A heading with a name so that linking to it is possible
hwrite(hwrite("A Level 2 Heading", name = 'a named heading'),
       heading = 2, center = FALSE, br = TRUE)

### Produce and display a simple html file
### Create a temporary file first
tmpDir <- tempdir()
fileName <- file.path(tmpDir, 'hwriterPlus.html')
### Copy some necessary files
cssDir <- file.path(system.file(package = 'hwriterPlus'), 'css')
file.copy(file.path(cssDir, "BrowserExample.css"), tmpDir)

### Set up some header information
### Open file for writing
p <- newPage(fileName,
             title = "Minimal Document",
             link.css = c("BrowserExample.css"))
hwrite("", p, br = TRUE)
hwrite(paste("<span class = 'title'>",
             "Minimal Document</span>"),
       p, center = TRUE, br = TRUE)
hwrite("<span class = 'subtitle'> Dr. David J. Scott</span>",
       p, center = TRUE, br = TRUE)
hwrite(paste("<span class = 'subtitle'>",
             format(as.Date("2011-11-16"), '%B %d, %Y'), "</span>",
             sep = ""),
       p, center = TRUE, br = TRUE)
hwrite(hwrite("Entering Text", name = "intro"), p,
       heading = 1, center = FALSE, br = TRUE)
hwrite(paste("Ordinary paragraph text can be entering using",
             "<font face = 'monospace' >hwrite</font>.",
             "Aspects of the text such as the font family,",
             "face, size and colour can be altered in line by using",
             "tags or styles."),
       p, center = FALSE, br = TRUE)
hwrite("", p, br = TRUE)
hwrite(paste("For example the font can be changed to sans-serif",
             "<font face = 'Arial, sans-serif'> like this,",
             "</font> italic <i> like this, </i> bold <b> like this,",
             "this,</b>coloured <font color = 'blue'> like this,",
             "</font> or any combination of these",
             "<b><i><font color = 'blue'>like this.</font></i></b>"),
      p, br = TRUE)
hwrite("", p, br = TRUE)

close(p)
```

```
### Open a web browser and examine the resulting file
if (interactive()) try(browseURL(fileName))
```

`hwriteLatex` *Insert LaTeX Code into an HTML File*

Description

This makes use of the `MathJax` JavaScript functions. Standard LaTeX input will be turned into MathML and displayed in any modern browser.

Usage

```
as.latex(x, label = NULL,
         inline = ifelse(is.null(label), TRUE, FALSE),
         count = ifelse(is.null(label), FALSE, TRUE))
hwriteLatex(ltx, page = NULL,
            table.attributes = NULL,
            tr.attributes = "",
            td.attributes = NULL, ...)
```

Arguments

`x`	Character. A string containing a mathematical expression in LaTeX notation.
`label`	Character. A text string giving the label to be displayed before the equation.
`inline`	Logical. Set to `TRUE` for a mathematical expression to be included inline. Set to `FALSE` for a displayed mathematical expression.
`count`	Logical. Set to `TRUE` for a numbered equation, `FALSE` otherwise.
`ltx`	Either a string containing a mathematical expression in LaTeX notation or an object of class `latex` as produced by `as.latex`.
`page`	Character. An optional connection. A character string naming the file to write to or a page object returned by `newPage` or openPage.
`table.attributes`	Character. A string specifying attributes of the table for a displayed equation. See details.
`tr.attributes`	Character. A string specifying attributes of the table row for a displayed equation. Default is "" meaning no special attributes.
`td.attributes`	Character. A string specifying attributes of the table data entries for a displayed equation. See details.
`...`	Possible attributes to be attached to the mathematical expression.

Details

Mathematical expressions in LaTeX format will be translated into MathML by the `MathJax` JavaScript program.

Mathematical expressions can be inserted *inline* (within text) or on a single line as a displayed expression, corresponding to the use of $...$ and $$...$$ in normal LaTeX.

One problem with writing LaTeX expressions is that all backslashes must be repeated. See the examples.

See the examples for an example of arguments to be supplied to `newPage` to ensure mathematical expressions are displayed.

If `ltx` is a string containing a mathematical expression in LaTeX notation, or if it is an object of class `latex` with `inline = TRUE` then any values assigned to `table.attributes` `tr.attributes` and `td.attributes` are ignored. Otherwise the default value of `table.attributes` when this argument is `NULL` depends on whether the mathematical expression has an equation number or not, which is determined by whether `count` in the `latex` object is `TRUE` or `FALSE`. If `count = FALSE` then `table.attributes` is assigned the value `"border = '0'"`. If `count = FALSE` then `table.attributes` is assigned the value `"border = '0'   width = '90%'"`. Similarly the default value of `td.attributes` depends on whether the mathematical expression has an equation number or not. If `count = FALSE` then `td.attributes` is ignored. If `count = FALSE` then `table.attributes` is assigned the value `c("width = '50'","align = 'center'",   "align = 'right' width = '50'")`. The explanation for these assignments is that if a displayed mathematical expression has no equation number it is constructed as a table comprised of a single row with a single entry in that row. If the expression has an equation number, it is constructed as a table comprised of a single row with three entries: the first is a non-breaking space, the second is the mathematical expression itself and the third is the equation number.

Note that these functions are modified versions of `as.latex` and `HTML.latex` from the package **R2HTML**. The most notable changes are that `HTML.latex` does not allow for inclusion of table and table data attributes in mathematical expressions and the numbering of equations uses R code rather than JavaScript. The numbering system for equations adds an entry to the global vector `hwriterEquationList` whenever a numbered equation is created. If a label is specified via the `label` argument, by for example, `label = "anequationlabel"`, then an anchor is created as part of the table data entry containing the equation number, using an HTML `a` element. The actual anchor name is `eq:anequationlabel`, which copies the label categorizing approach used in LaTeX. If no label is specified, but the equation is numbered, the name used for the anchor is `eq:equationxx` where `xx` is the equation number. Then the number of an equation corresponding to a given label may be retrieved using the function `eqRef` (intended to be reminiscent of the LaTeX `\ref` command).

To see what attributes are valid for tables, table rows and table data, go to `http://www.w3schools.com/tags/` and click on the desired tag.

Value

`as.latex` returns a list of class `latex` with components:

`alt`	A character string containing the A string containing a mathematical expression in LaTeX notation.
`inline`	A logical value.

`count`	A logical value.
`label`	If not `NULL` a character string.

`hwriteLatex` produces a character vector containing the output HTML code. If page is specified this text is appended to the specified file.

Author(s)

David Scott <`d.scott@auckland.ac.nz`>

References

MathJax: `http://www.mathjax.org/`

See Also

`newPage`

Examples

```
as.latex("See what is added by as.latex", label = "a label",
          inline = FALSE, count = TRUE)
### Run this example to see what is written to the file
## Not run:
### An inline expression will be written to standard output as is
### Note that the backslashes remain duplicated in this case
### I think this is due to a line of code in hwriter::hwriteString
###   if (is.null(page)) txt
### which writes the txt literally rather than using cat
hwrite("Here is an inline mathematical expression:
        `\int_{-\infty}^{1}f(x)dx`")
### The same expression but displayed
hwriteLatex(as.latex("\int_{-\infty}^{1}f(x)dx",
                     inline = FALSE, count = FALSE),
            table.attributes = "border = '1'",
            tr.attributes = "bgcolor = 'white'")


### The same example written to a file
### Create a temporary file first
tmpDir <- tempdir()
fileName <- file.path(tmpDir, 'hwriteLatex.html')
### Copy some necessary files
cssDir <- file.path(system.file(package = 'hwriterPlus'), 'css')
file.copy(file.path(cssDir, "BrowserExample.css"), tmpDir)

### Open file for writing
p <- newPage(fileName,
             title = "Example of a Document for Display in a Browser",
             link.css = c("BrowserExample.css"))
hwrite("Here is an inline mathematical expression:
        `\int_{-\infty}^{1}f(x)dx`", p, br)
```

```
### The same expression but displayed
hwrite("Here is a displayed version of the same expression enclosed in a
       box", p, br)
hwriteLatex(as.latex("\int_{-\infty}^{1}f(x)dx",
                     inline = FALSE, count = FALSE),
            p,
            table.attributes = "border = '1'",
            tr.attributes = "bgcolor = 'white'")
### We can also have equation numbers and labels when writing to a file
hwrite("Here is a different expression which has an equation number and
assigned a label", p, br)
hwriteLatex(as.latex("\{ 26.119 < \sum_{i=1}^n(X_i-\bar{X})^2\}
\bigcup\ \{ 5.629 > \sum_{i=1}^n (X_i-\bar{X})^2 \}.",
                     inline = FALSE, label = "anequationlabel"),
            page = p,
            tr.attributes = "bgcolor = 'gray'",
            td.attributes = c("width = '50'",
                              "align = 'center' bgcolor = 'yellow'",
                              "align = 'right' width = '50'"))
closePage(p)
### Open a web browser and examine the resulting file
if (interactive()) try(browseURL(fileName))

## End(Not run)
```

`hwriteOutput` *Write R Output or an R Script in HTML Format*

Description

`hwriteOutput` is a wrapper function for `hwrite` which formats output from R captured by `capture.output` so it appears correctly when `hwrite` is called.

`hwriteScript` is a very similar wrapper function for `hwrite` which formats an R session captured by `script` so it appears correctly when `hwrite` is called. `hwriteScript` also optionally trims lines added at the beginning and end of the session by `script`.

Usage

```
hwriteOutput(output, page = NULL, fontSize = "10pt", ...,
             link = NULL, name = NULL, br = NULL, div = NULL)
hwriteScript(scriptFile, page = NULL, fontSize = "10pt", trim = TRUE, ...,
             link = NULL, name = NULL, br = NULL, div = NULL)
```

Arguments

`output`	Text. Output produced by R, captured by `capture.output`
`page`	An optional connection. A character string naming the file to write to or a page object returned by `newPage` or `openPage`.

`fontSize`	Character string. The font size in points in which the output will be displayed. A positive number followed by the characters `"pt"` without a space between. Default is `"10pt"`.
`scriptFile`	File. Contains text from an R session captured by `script`.
`trim`	A logical. If `TRUE`, deletes the first and last two lines of the lines of text returned by `script`. Default is `TRUE`.
`...`	Optional arguments. See Details section of `hwrite` documentation.
`link`	A character vector containing the URLs the HTML element will point to. This argument is the equivalent of the attribute `href` of the HTML tag <a>.
`name`	A character string naming the HTML element for further reference. This is the equivalent of the attribute `name` of the HTML tag<a>.
`br`	A logical specifying if a breakline (carriage return) should be appended at the end of `x`. Default is `FALSE`.
`div`	A logical. If `TRUE`, places the HTML element into a HTML section, using the <div> HTML tag. This is helpful for styling a section. Default is `FALSE`.

Details

The output obtained from R using `capture.output` is modified by these functions so that it will appear verbatim in the resulting HTML file, as preformatted text.

The output is first modified by adding characters `"\n"` to the end of each output line. Then preformatting tags are added to surround the output. The opening preformatting tag includes a style definition which sets the font size to that specified by `fontSize`.

In addition `hwriteScript` can trim off the first line and the last two lines of the text returned by a call to `script`. The first and last lines record the start and end times of the call to `script`. The second last line typically consists of the text `> q()` comprised of the R prompt and the call to `q()` which terminates the recording of the session by `script`.

Value

A character vector containing the output HTML code.

Author(s)

David Scott <`d.scott@auckland.ac.nz`>

See Also

`hwrite`

Examples

```
### Example results written to standard output stream
### Annette Dobson (1990) "An Introduction to Generalized Linear Models".
### Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
```

```
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
plants <- data.frame(weight = weight, group = group)
strOutput <- capture.output(str(plants))
lm.D9 <- lm(weight ~ group, data = plants)
anovaOutput <- capture.output(anova(lm.D9))
hwrite("Analysis of Weight Data", heading = 1,
       center = FALSE, br = TRUE)
hwriteOutput(strOutput, fontSize = "8pt")
hwriteOutput(anovaOutput)

### Same example but written to a file
tmpDir <- tempdir()
fileName <- file.path(tmpDir, 'hwriterPlus.html')
p <- newPage(fileName)
hwrite("Analysis of Weight Data", p, heading = 1,
       center = FALSE, br = TRUE)
hwriteOutput(strOutput, p, fontSize = "8pt")
hwriteOutput(anovaOutput, p)
closePage(p)
### Open a web browser and examine the resulting file
if (interactive()) try(browseURL(fileName))


### Record of session captured by script written to temporary file
## Not run: tmpDir <- tempdir()
tmpFile <- file.path(tmpDir, 'script.txt')
script(tmpFile)
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
plants <- data.frame(weight = weight, group = group)
lm.D9 <- lm(weight ~ group, data = plants)
q()
sessionOut <- readLines(tmpFile)
sessionOut
hwriteScript(tmpFile, br = TRUE)
hwriteScript(tmpFile, trim = FALSE, fontSize = "8pt", br = TRUE)

## End(Not run)
```

hwriteSVG *Write HTML Code to Include an SVG Image*

Description

Write code to include an SVG image in an HTML document which should work with most modern browsers.

Usage

```
hwriteSVG(image.url, page = NULL, width = 500, height = 500,
          id = "", attributes = "border = '0'", ...)
```

Arguments

`image.url`	A character vector or matrix containing the URL or the file path of the SVG image.
`page`	An optional connection. A character string naming the file to write to or a page object returned by `newPage` or `openPage`.
`width, height`	An HTML length unit (in pixels) specifiying the width (respectively height) at which the image should be rendered. Must be present if browser being used is Internet Explorer. If missing, Firefox will use the default image width(resp. height).
`id`	Character. A string providing an identifier for the SVG image.
`attributes`	Attributes of the SVG image. See details.
`...`	Optional arguments that will be dispatched to the underlying `hwrite` call.

Details

Inserting workable SVG code into an HTML document is difficult because the attributes of the `<object>` tag required depend on the browser used to view the document. `hwriteSVG` uses syntax taken from support documentation for the SVG Web project. The example given in the sample `helloworld.html` is

```
<!--[if !IE]>-->
  <object data="../svg-files/helloworld.svg" type="image/svg+xml"
          width="200" height="200" id="mySVGObject"> <!--<![endif]-->
<!--[if lt IE 9]>
  <object src="../svg-files/helloworld.svg" classid="image/svg+xml"
          width="200" height="200" id="mySVGObject"> <![endif]-->
<!--[if gte IE 9]>
  <object data="../svg-files/helloworld.svg" type="image/svg+xml"
          width="200" height="200" id="mySVGObject"> <![endif]-->
  </object>
```

Note that recent versions of Firefox should display SVG images without problems, and should not require the specification of height and width. Internet Explorer is far less foregiving and both height and width values must be given or no image will be displayed. Experimentation with appropriate values for the arguments may be necessary.

For attributes which may be included in an `<object>` tag, see the reference for this tag on the website `http://www.w3schools.com`.

Value

A character vector containing the output HTML code.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

http://www.w3schools.com/tags/tag_object.asp http://codinginparadise.org/projects/svgweb/docs/QuickStart.html

See Also

hwriteImage

Examples

```
examplesDir <- file.path(system.file(package = 'hwriterPlus'),
                         'examples')
catsSVG <- file.path(examplesDir, "cats.svg")
### Raw html produced
hwriteSVG(catsSVG, page = NULL, height = 600, width = 600,
          center = FALSE, br = TRUE)
### Formatted  a little more nicely
svgHTML <- hwriteSVG(catsSVG, page = NULL, height = 600, width = 600,
          center = FALSE, br = TRUE)
unlist(strsplit(svgHTML, "\n"))
```

---

newPage *newPage*

Description

Opens an HTML page/document, allowing a sequential building of an HTML page.

Usage

```
newPage(filename, dirname = NULL, title = filename,
        doctype = "<!DOCTYPE html>\n",
        link.javascript = NULL, link.css = NULL, css = NULL,
        head = NULL, head.attributes = NULL,
        body.attributes = NULL)
```

Arguments

filename	A character string containing the filename or the path of the HTML file to be created.
dirname	An optional character string containing the path of the directory where the file should be written.
title	An optional character string containing the title of the HTML page.

`doctype`	A character string containing a `DOCTYPE` statement which declares the document type and level of HTML syntax.
`link.javascript`	An optional character vector containing the URL of JavaScripts to be associated with the page.
`link.css`	An optional character vector containing the URL of CSS stylesheets to be associated with the page.
`css`	An optional character vector containing inline CSS stylesheets to be associated with the page.
`head`	An optional character string containing an HTML fragment to be added in the `<head>` section of the page.
`head.attributes`	An optional named list of character strings, containing the `<head>` attributes.
`body.attributes`	An optional named list of character strings, containing the `<body>` attributes.

Details

`newPage` opens a new file for writing and returns a page handle which is used by `hwrite` to append HTML elements in a current page. Any previous existing file will be overwritten.

The `doctype` argument is used to specify the document type and level of HTML syntax. Valid forms may be found at `http://www.w3.org/QA/2002/04/valid-dtd-list.html`.

`link.javascript` produces A `<script>` tag specifying the language as JavaScript and stating the location of the code. **hwriterPlus** uses MathJax from `http://www.mathjax.org/`. The JavaScript may either be loaded from the MathJax site, or from a local file. The former option requires an active internet connection, while the latter requires that MathJax must have been installed locally. The form of the text for this argument is `"path-to-MathJax/MathJax.js"`. See the examples for the appropriate text when loading from MathJax directly. Details of how to ensure MathJax is loaded, for Version 2 of MathJax may be found at `http://www.mathjax.org/docs/2.0/configuration.html`.

The argument `head` is useful to add extra HTML code in the `<head>` header code.

`head.attributes` is useful to add extra attributes in the `<head>` head code. A description of possible head attributes may be found at `http://www.w3schools.com/tags/tag_body.asp`.

`body.attributes` is useful to add extra attributes in the `<body>` body code. A description of possible body attributes may be found at `http://www.w3schools.com/tags/tag_body.asp`.

To allow the numbering of displayed equations `newPage` creates a global variable called `hwriterEquation` and assigns it the value zero. This variable will be updated when a numbered equation is written to the HTML file so that it always records the number of the last numbered equation. It also creates a a global vector named `hwriterEquationList`, which is initially empty, but will contain the labels of any numbered equations.

Value

A connection which is a handle to the current HTML page.

Note

newPage is a modification of openPage from **hwriter** which includes a doctype argument but eliminates the charset and lang arguments provided by openPage.

Author(s)

David Scott <d.scott@auckland.ac.nz>

References

An invaluable source of information concerning valid HTML code and options is the website http://www.w3schools.com/.

See Also

openPage

Examples

```
### The same as the example from openPage, with newPage replacing
### openPage

### Creates a new web page 'test.html' in the R temporary directory
tmpdir <- tempdir()
p <- newPage('test.html', dirname = tmpdir,
              link.css = 'http://www.ebi.ac.uk/~gpau/hwriter/hwriter.css')
hwrite('Iris example', p, center = TRUE, heading = 1)
hwrite('This famous (Fisher\'s or Anderson\'s) iris data set gives the
        measurements in centimeters of the variables sepal length and
        width and petal length and width, respectively, for 50 flowers
        from each of 3 species of iris.', p, class = 'king')
hwrite(iris, p, row.bgcolor = '#ffffaa')
closePage(p)

### Opens a web browser on the web page
if (interactive()) try(browseURL(file.path(tmpdir, 'test.html')))
```

---

ref *Convenience Functions for Retreiving Reference Numbers*

Description

Given a label as a character string, the vector of labels is searched and the row number of the matching vector is returned. This is the number that has been assigned to the object with the specified label.

Usage

```
eqRef(label)
```

Arguments

`label`	Character. The label of the object whose number is sought.

Details

The function first adds the required prefix (e.g. `eq:` for an equation) to the supplied label, then uses `which` to find the row number in the vector of labels matching that label. Names of vectors storing the labels are based on LaTeX names. For equations, the vector is called `hwriterEquationList` (even though it is a vector not a list).

Value

The number in the list of labels for that type of object which matches the label supplied.

Author(s)

David Scott <`d.scott@auckland.ac.nz`>

See Also

`which`, `hwriteLatex`

Examples

```
## Not run:
eqRef{"anequationlabel"}

## End(Not run)
```

`script` *Record an R Session*

Description

Records an R session or part session and saves to a file.

It is an analog of the Unix script(1) command.

Usage

```
script(file = "transcript.txt")
```

Arguments

`file`	Character. The name of the text file where the record of the session is to be saved.

Details

To start recording (part of) an R session use script(\file{filename}). To finish recording use `q()`. See the Examples section.

Value

No value is returned. The function exists for the side effect it produces, which is the details of an R session recorded in a file.

Note

This is best regarded as an exercise in getting familar with R's condition system and a demonstration of how to write an interpreted REPL (Read-Eval-Print-Loop).

FIXME: Nested calls to "script" are not a good idea. The boxing-glove UI element should probably be used to prevent this.

BUG/Feature: When a parse is interrupted, nothing appears in the transcript. When a command is interrupted, no output appears. This could be fixed with a calling handler, or by shifting the interrupt catcher inside repl().

The function `txtStart` from **TeachingDemos** fullfills a similar function, but fails to preserve line-breaks on input of R commands.

Author(s)

Ross Ihaka `<r.ihaka@auckland.ac.nz>`

Examples

```
## Not run:
### Record of session captured by script written to temporary file
tmpDir <- tempdir()
tmpFile <- file.path(tmpDir, 'script.txt')
script(tmpFile)
### Annette Dobson (1990) "An Introduction to Generalized Linear Models".
### Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
plants <- data.frame(weight = weight, group = group)
lm.D9 <- lm(weight ~ group, data = plants)
q()
sessionOut <- readLines(tmpFile)
sessionOut

## End(Not run)
```

Index