Package 'makehtml'

June 28, 2022

Type Package

Title Provides Generic Code for Producing Tabbed HTML for Presenting Plotted Results

Version 0.0.5

Date 2021-09-10

Maintainer Malcolm Haddon <malcolm.haddon@gmail.com>

Description Contains code to include Cascading Style Sheet codes into a series of HTML files each of which constitutes a tab into a Home HTML file. Initially this is designed for use with the Abalone projects I am working on but I will will also work to generalize this for any project that could generate graphics and tables. The graphics files need to be .png files. This is still under development but can be used currently.

Depends R (>= 3.5.0)

License GPL (>= 3)

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Malcolm Haddon [aut, cre]

R topics documented:

`addplot` *addplot adds a plot's filename to the autoresult csv file*

Description

addplot is used to facilitate the production of the HTML results summary for a particular run. If the filen remains empty, then addplot does nothing. However, if filen is longer than "", then addplot adds the plot's filename and the supporting caption and category to resultTable.csv that lists all files to be included in the local website. If no category is added explicitly then the local webpage will have an 'any' tab containing these unloved results.

Usage

```
addplot(filen, rundir, category = "any", caption = "")
```

Arguments

`filen`	full path and filename for file being added
`rundir`	full path to the directory to contain the results
`category`	what HTML tab should it be added to? default="any" obviously? you would want to change this.
`caption`	the caption for the figure or table, default = "", This should not contain commas as this confuses the csv file. But if you accidentally put some in they will be removed.

Value

nothing but it does add a line to resfile

Examples

```
indir <- tempdir()
rundir <- filenametopath(indir,"result")
dirExists(rundir,verbose=FALSE)
resfile <- setuphtml(rundir=rundir)
filename <- filenametopath(rundir,"example.png") # must be a png file
png(filename=filename,width=7,height=4,units="in",res=300)
plot(runif(100),runif(100),type="p")
addplot(filen=filename,rundir=rundir,"A_category",caption="Example Figure")
dir(rundir)
```

addtable *addtable adds a table to the output*

Description

If the filen remains empty, then addtable does nothing. Otherwise, addtable saves a table as a csv file into the rundir. Then it logs the filename in the the resultTable.csv file containing the names of each file to be plotted or tabulated. This function saves having to combined the filename with the rundir, and then does the filename logging for you. If no category is added explicitly then the local webpage will have an 'any' tab containing these unloved results.

Usage

```
addtable(intable, filen, rundir, category = "any", caption = "", big = FALSE)
```

Arguments

intable	the table or data.frame to be saved and output
filen	only the filename being given to the table, no path included
rundir	full path to the directory to contain the results
category	what HTML tab should it be added to? default="any"
caption	the caption for the figure or table, default = ""
big	if FALSE (the default) the complete table is generated, if TRUE then scroll bars are added.

Value

nothing but it does add a line to resultTable.csv and saves a csv file to rundir

Examples

```
## Not run:
indir <- tempdir()
rundir <- filenametopath(indir,"result")
dirExists(rundir,verbose=FALSE)
resfile <- setuphtml(rundir,"example_only")
filen <- "example.csv"
egtable <- matrix(rnorm(25,0,1),nrow=5,ncol=5)
addtable(egtable,filen=filen,rundir=rundir,"A_category",
            caption="An example Table")
dir(rundir) # examine the resfile and the example.csv files.

## End(Not run)
```

addtext *addtext literally enables the addition of a text block to a tab*

Description

addtext provides a solution for when one wants to add a block of explanatory text to a tab in the internal website created by make_html.

Usage

```
addtext(txt, rundir, filename, category = "text")
```

Arguments

txt	the vector of character strings to be added as a text block.
rundir	full path to the directory to contain the results
filename	the name of text or CSV file to store ready for inclusion in the htmlfile defined by category. Be sure to include the filetype .txt
category	what HTML tab should it be added to? default="text"

Value

nothing but it does add text to a tab within an html file in rundir. It does not make a note in resultTable.csv

Examples

```
print("wait on suitable data")
# txt=txt;rundir=rundir; filename="test.txt";category="text"
```

cleanrundir *cleanrundir removes all html, png, and css files from rundir*

Description

cleanrundir removes all html, png, and css files from rundir. In addition, if abmse=TRUE, the default, then up to eight specific files are removed. This function is to replace the use of cleanslate in setuphtml, which was more clumsy. The special aMSE files to be removed are: "final_harvestR.csv","glb.RData","hsargs.
"HSstats.RData","popdefs.csv","propertyDD.csv","resultTable.csv", "zonebiology.csv".

Usage

```
cleanrundir(rundir, verbose = TRUE, abmse = TRUE)
```

Arguments

rundir	a specific directory used for an analytical run
verbose	default=TRUE. This assumes no action unless an explicit Y or y is entered at the prompt. If set to TRUE it will delete the files with no explicit prompt.
abmse	should a specific set of files be deleted (see description)

Value

invisibly the remaining contents of rundir as a vector

Examples

```
## Not run:
  print("wait on an example file")

## End(Not run)
```

`dirExists` *dirExists: Checks for the existence of a directory*

Description

dirExists: does a directory exist? It uses dir.exists and reports existence if already present and uses dir.create it it does not exist, but avoids the warning message is one already exists. The option of not creating a new directory is also present. This uses 'recursive=TRUE' inside the dir.create, so one should be able to create directories as deeply down a path as wished.

Usage

```
dirExists(indir, make = TRUE, verbose = TRUE)
```

Arguments

`indir`	a character string containing the name of the directory whose existence is to be checked before it is created if it does not already exist.
`make`	if the directory does NOT exist should it be created. default = TRUE; if make=FALSE and a directory does not exist a warning will be given to the console.
`verbose`	default=TRUE, prints directory status to the console, If make is set to FALSE and a directory does not exist a warning will always be given.

Value

a message to the screen if the directory exists or is created; if make is TRUE then it also creates the directory as listed in 'indir'.

Examples

```
indirect <- getwd()
dirExists(indirect)
```

endmakehtml *endmakehtml write the syntax of the website generation to console*

Description

endmakehtml writes to the console the syntax of the final set of commands needed to generate the internal website of results. The idea is that one would then copy those lines to the end of one's own code to complete the output of results. Its functions is simply to facilitate remembering the syntax.

Usage

```
endmakehtml()
```

Value

nothing but it does write some code syntax to the console

Examples

```
endmakehtml()
```

filenametopath *filenametopath safely add a filename to a path*

Description

filenametopath add a filename to a path safely, using pathtype to get the seperator and then checks the end character. If the separator is nothing or a '/' or a '//' then it adds to the path appropriately. Without this one can unwittingly include extra separators or none at all.

Usage

```
filenametopath(inpath, infile)
```

Arguments

inpath	the path to be analysed
infile	the filename to be added to the path 'inpath'

Value

the completed filename or extended path

Examples

```
indir <- tempdir()
infile <- "control.csv"
filenametopath(indir,infile)
```

getextension *getextension gets the 3 letter file extension from a filename*

Description

getextension is a utility function that is used with logfilename to determine whether one is dealing with a csv or a png file. depending on which it will return either a 'table' or 'plot' value. Any other extension type will throw an error and stop execution.

Usage

```
getextension(filename)
```

Arguments

filename	the filename, with path attached, whose extension needs to be determined.

Value

a character string of either 'table' or 'plot'

Examples

```
filen <- "not_a_real_file.png"
getextension(filen)
filen <- "another_unreal_file.csv"
getextension(filen)
```

htmltable *htmltable generates the html to print out a table*

Description

htmltable generates the required html code to add a table to the website of the results. This requires the logfilename function to include both a category biology, productivity, etc, and a type, which is currently limited to plot or table.

Usage

```
htmltable(inmat, filename, caption, basename, big = FALSE)
```

Arguments

inmat	the 2D matrix or data.frame to be printed
filename	the filename to which to add the html, defined by makehtml
caption	the caption text placed at the top of the table
basename	the name of the csv file being tabulated.
big	if FALSE (the default) the complete table is generated, if TRUE then scroll bars are added.

Value

nothing but it does add some html to the input filename

`logfilename` *logfilename adds a filename to the autoresult csv file*

Description

logfilename is used to facilitate the produciton of the HTML results summary for a particular run. This depends upon a csv file containing the names of each file to be plotted or tabulated. This function adds a filename and the supporting caption and category, without one needing to remember the syntax. If no category is added explicitly then the local webpage will have an 'any' tab containing these unloved results. logfilename would not usually be called independently of addtable or addplot but it is exported for completeness.

Usage

```
logfilename(filename, resfile, category = "any", caption = "", type = "")
```

Arguments

`filename`	the full path and filename for the file being added
`resfile`	the file to be added to, which is defined by setuphtml found in aMSE_utils
`category`	what HTML tab should it be added to? default="any"
`caption`	the caption for the figure or table, default = "", This should not contain commas as this confuses the csv file. But if you accidentally put some in they will be removed.
`type`	allows one to override the plot and table options. Currently the alternative is bigtable

Value

nothing but it does add a line to resfile

Examples

```
indir <- tempdir()
rundir <- filenametopath(indir,"result")
dirExists(rundir,verbose=FALSE)
resfile <- setuphtml(rundir)
filename <- filenametopath(rundir,"example.png")
png(filename=filename,width=7,height=4,units="in",res=300)
plot(runif(100),runif(100),type="p")
dev.off()
logfilename(filename=filename,resfile=resfile,"A_category",
            caption="Example Figure")
dir(rundir)
```

`makehtml` *makehtml functions to generate tabbed HTML output files*

Description

The makehtml package is designed to assist with the presentation of the results of analyses where multiple types of output files are expected. For example, in an Abalone MSE, one would expect to summarize the data used to define the operating model for each run, to summarize the time-series outputs for the various populations and trends of interest, to summarinze the outcomes in terms of the performance measured used, the outputs from teh harvest control rules trialed, and so on, across an array of categories of output. makehtml assists by sequentilly plotting these into seperate named tabs in a web-site based output.

CSS functions

- addcss literally prints the CSS codes to each HTML sheet generated for each category of output

HTML functions

- make_html takes the input directory, looks for the filelist in plotFileTable_<name>.csv and then generates a Home sheet, and a separate sheet for each category of output file

`make_html` *make_html create HTML files to view results in a browser.*

Description

make_html writes a set of HTML files with tabbed navigation, with each tab containing the results relating to a given set of results or diagnostics from a particular aMSE run. This code was borrowed from Ian Taylor's r4ss, but has been extensively modified to improve both the css (see write_css) and the HTML. By default, this function will look in the results directory where PNG and CSV files were created for a resultTable.csv file. HTML files are written to link to these plots and put in the same directory. Output now includes the control, data, and HS files names.

Usage

```
make_html(
  replist = NULL,
  rundir = NULL,
  datadir = NULL,
  controlfile = NULL,
  datafile = NULL,
  hsfile = NULL,
  width = 500,
  openfile = TRUE,
  runnotes = NULL,
  verbose = TRUE,
  packagename = "aMSE",
  htmlname = "aMSE"
)
```

Arguments

replist	Object created by a run, can be NULL
rundir	Directory where a particular run's files, including any results, as tables and plots, and any other files, are all held. Cannot be NULL.
datadir	full path to the data directory, if one is used
controlfile	the character name of the control file used.
datafile	the character name of the saudata data file.
hsfile	the character name of the harvest strategy file, default=NULL
width	Width of plots (in pixels). Default = 500
openfile	Automatically open index.html in default browser?
runnotes	Add additional notes to home page.
verbose	Display more info while running this function?
packagename	name of the main package being used in the analysis. default='aMSE'. This is described on the home page
htmlname	first name of the html files generated, default='aMSE'

Author(s)

Originally Ian Taylor, modified by Malcolm Haddon

---

pathend *pathend determines what character is at the end of a path*

Description

pathend determines what character is at the end of a path uses pathtype to get the seperator and then checks the end character

Usage

```
pathend(inpath)
```

Arguments

inpath	the path to be analysed

Value

the end character of the path; either NA, '/', or "\"

Examples

```
indir <- "C:/Users/Malcolm/Dropbox/rcode2/aMSE/data-raw"
pathend(indir)
```

pathtype *pathtype finds the type of separator used in a path*

Description

pathtype finds the type of separator used in a path, this is either a '/' or a '\'

Usage

```
pathtype(inpath)
```

Arguments

inpath	- the path to be analysed

Value

the type of path divider, either a 0 = '\' or a 1 = '/'

Examples

```
indir <- "C:/Users/Malcolm/Dropbox/rcode2/aMSE/data-raw"
pathtype(indir)
```

setuphtml *setuphtml initiates csv files listing results to be included*

Description

setuphtml always initiates the resultTable.csv file used to contain the filenames, captions, and categories of the plots and tables to be included in the html results. The format of the csv file is to have column names of file, caption, category, and timestamp. Then, each plot and table is included with an entry for each column.

Usage

```
setuphtml(rundir, cleanslate = FALSE)
```

Arguments

rundir	full path to the directory to contain the results
cleanslate	this is now deprecated and has been replaced by the function cleanrundir, which is safer and clearer.

Value

invisibly, the full path to the resfile, after creating the file in rundir and potentially deleting all previous html, png, and .css files contained in rundir

Examples

```
indir <- tempdir()
rundir <- filenametopath(indir,"results")
dirExists(rundir,verbose=FALSE)
resfile <- setuphtml(rundir)
dir(rundir)
```

`write_css` *write_css generates a CSS file used by all html files*

Description

write_css generates a cascading style sheet that will be used by each separate html files made for each category of results. The origin came from Ian Taylor although I have modified the styles and ensure correct Version 3 CSS

Usage

```
write_css(rundir, htmlname)
```

Arguments

`rundir`	the directory within the run directory that contains all the plot results
`htmlname`	name of the css files generated; input to make_html

Value

nothing but it does generate a .css file in the rundir

`write_head` *write_head adds the <head> tag to each html file in results*

Description

write_head adds the head tag to each html file. It links to the css file rather than writing the complete css code to every file. There were missing elements which are now in place, so the code is now valid HTML5.

Usage

```
write_head(htmlfile, htmlname)
```

Arguments

`htmlfile`	the particular html file being worked on. This is defined within the make_html function.
`htmlname`	name of the css files generated; input to make_html

Value

nothing but it does add the <head> tag to each html file

Index