

Package ‘Rpad’

April 17, 2009

Title Workbook-style, web-based interface to R

Version 1.3.0

Author Tom Short (EPRI), Philippe Grosjean (UMH EcoNum)

Description A workbook-style user interface to R through a web browser. Provides convenient plotting, HTML GUI generation, and HTML output routines. Can be used with R in standalone mode or with a webserver to serve Rpad pages to other users.

Depends graphics, utils, grDevices

Suggests tcltk, R2HTML

LazyLoad no

Maintainer Tom Short <tshort@epri.com>

License GPL (>= 2)

URL <http://www.rpad.org/Rpad>

Repository CRAN

Date/Publication 2007-04-25 21:00:25

R topics documented:

json	2
RpadGraphing	3
RpadHTML	5
RpadLocalServer	8
RpadUtil	10
Index	12

`json`*JSON (JavaScript Object Notation) generation*

Description

Write out a JSON structure of an R object.

Usage

```
json(x, ...)
```

Arguments

<code>x</code>	an object to convert to JSON.
<code>...</code>	unused for now.

Details

JSON is Javascript Object Notation, a data-exchange format especially suited for passing to Javascript. Mapping from S to Javascript is as follows:

list (including data frames and other list objects) -> object character -> string array -> array array
with names -> object matrix -> 2-D array numeric -> number NULL -> null TRUE -> true FALSE
-> false NA -> NaN

Other types are converted to character using `as.character` and output as character arrays or objects. Handling of NA's is problematic as Javascript doesn't have NA's. Strings are escaped. Currently, dimensioning in higher dimension arrays is lost. Object attributes are also ignored.

Value

A character string of class `json`. With automatic printing, the string is sent to the output (with `cat`).

Author(s)

Tom Short, EPRI, (tshort@epri.com)

See Also

See also <http://www.json.org/>

Examples

```
json( 1:10 )
## [1,2,3,4,5,6,7,8,9,10]

json( c("apple", "line\nwith break") )
## ["apple","line\nwith break"]

json( c(a = 1, b = 5, c = 10) )
```

```
## {"a":1,"b":5,"c":10}

json( data.frame(a = 1:5, b = 6:10) )
## {"a":[1,2,3,4,5],"b":[6,7,8,9,10]}

json( list(a = 5, b = 1:10, c = data.frame(a = 1:2, b = 3:4) ) )
## {"a":5,"b":[1,2,3,4,5,6,7,8,9,10],"c":{"a":[1,2],"b":[3,4]}}
```

RpadGraphing

Rpad graphing utilities

Description

Utilities to make graphing in Rpad R scripts easier.

Usage

```
graphoptions(..., reset = FALSE, override.check = TRUE)
newgraph(name = "", extension = graphoptions()$extension,
         type = graphoptions()$type, res = graphoptions()$res,
         width = graphoptions()$width, height = graphoptions()$height,
         deviceUsesPixels = graphoptions()$deviceUsesPixels,
         pointsize = graphoptions()$pointsize, sublines = graphoptions()$sublines,
         toplines = graphoptions()$toplines, ratio = graphoptions()$ratio,
         leftlines = graphoptions()$leftlines, lwd = graphoptions()$lwd, ...)
showgraph(name = RpadPlotName(), link = FALSE, ...)
RpadPlotName()
```

Arguments

<code>reset</code> , <code>override.check</code>	logical arguments passed to <code>check.options</code> .
<code>name</code>	is the name of the graph with the extension left OFF.
<code>extension</code>	is the file extension of the device (typically '.png').
<code>type</code>	is the graphics device, either a ghostscript device as a character string, "Rpng" for R's builtin png device, or any R graphics device function. For a ghostscript device, common possibilities are "png256", "pngalpha" (the default), or "pdf".
<code>res</code>	is the resolution of the bitmap in dots per inch.
<code>width</code> , <code>height</code>	are the dimensions of the graph in inches.
<code>deviceUsesPixels</code>	is a logical specifying whether the R graphics driver uses pixels. This only applies if <code>type</code> is used to specify an R graphics driver. It defaults to TRUE.
<code>ratio</code>	specifies the ratio of the graph if either the width or height is not specified.
<code>pointsize</code>	is the font point size passed to the postscript device.

<code>sublines</code> , <code>toplines</code> , <code>leftlines</code>	specify the dimension of the graph along with the outside border. It defaults to fairly tight outside dimensions.
<code>lwd</code>	is the line width set with <code>par</code> .
<code>link</code>	is a logical specifying whether to show a link to allow the user to download an EPS file of the graph (not available when using R's png driver).
<code>...</code>	in <code>fspdf</code> and <code>newgraph</code> , arguments are passed to <code>postscript</code> . In <code>graphoptions</code> , the arguments are assigned as defaults for <code>newgraph</code> .

Details

The `graphoptions`, `newgraph`, and `showgraph` set of functions allows quick setup and display of web-friendly graphics. The user can normally just use any of the plot commands followed by `showgraph`. `newgraph` sets up the graphics device, and it's called when the Rpad package starts. `showgraph` generates the HTML to show the graph and runs `newgraph` to advance to the next graphics file. The user only runs `newgraph` to change parameters from their defaults. Graphics files are by default named `Rpad_plot1`, `Rpad_plot2`, and so on. Named graphs can also be used, but there's more of a chance that if the user has caching set wrong (or the server's caching is set wrong) that graphs won't update properly in the user's browser. With the default sequential numbering of files, caching problems are less likely. `graphoptions` is also available to change the defaults for subsequent graphs.

Internally, `newgraph` uses the `postscript` device and `ghostscript` to generate the bitmap for the browser. The `pngalpha` (`type="pngalpha"`) device of `ghostscript` does anti-aliasing for smoother-looking PNG output. Also, this approach has the side-effect of creating an EPS file for each graph, so it's easy to add a link to allow the user to download the EPS file. A second approach is to use R's `png` device directly by specifying `type="Rpng"` (but on linux, this requires an X server, and results are not antialiased). A third approach allows arbitrary R graphic's devices to be specified. Simon Urbanek's `GDD` or `Cairo` packages can be used to generate `png` (for both packages, see <http://www.rosuda.org/R/>). This allows high-quality `png` generation with anti-aliasing, and you don't need `ghostscript` or `X11`.

Value

`RpadPlotName()` returns the name of the currently active plot. None of the other routines return a value: all are run for their side effects.

Author(s)

Tom Short, EPRI, ((tshort@epri.com))

See Also

See also `bitmap`, `png`, and `pdf`.

Examples

```
# make some graphs (a default graphics device is already available)
x <- 1:10
y <- x^2
```

```

y2 <- x^3
if (capabilities("png")) graphoptions(type="Rpng")
newgraph()
plot(x, y) # does the plot
plot(x, y2) # does the second plot
HTMLon() # sets Rpad to HTML output
showgraph() # closes the device, outputs the HTML for the both
# images, and creates the next device
plot(x, y2)
showgraph()

# graphs with named files:
newgraph("graph_A")
plot(x, y)
showgraph("graph_A")
newgraph("graph_B", width = 4, height = 6) # also adjust the width and height
plot(x, y2)
showgraph("graph_B")

# an example with Simon Urbanek's GDD device:
if (require(GDD)) {
  graphoptions(type = GDD, width = 4, height = 3) # note the use of inches
  newgraph()
  plot(x, y)
  plot(x, y2)
  showgraph()
}

```

RpadHTML

Rpad utilities

Description

Rpad utilities to generate HTML output.

Usage

```

ROutputFormat (Format)
HTMLon ()
HTMLoff ()
HtmlTree (tagName, ...)
H (tagName, ...)
## Default S3 method:
HtmlTree (tagName, ..., standaloneTag = FALSE, collapseContents = TRUE)
Html (x, ...)
BR
HTMLh1 (text)
HTMLh2 (text)
HTMLh3 (text)

```

```

HTMLh4(text)
HTMLh5(text)
HTMLlargs(x)
HTMLtag(tagName, ...)
HTMLetag(tagName)
HTMLradio(variableName, commonName = "radio", text = "", ...)
HTMLcheckbox(name, text = "", ...)
HTMLselect(name, text, default = 1, size = 1, id = name, contenteditablewrapper = TRUE)
HTMLinput(name, value = "", rpadType = "Rvariable", contenteditablewrapper = TRUE)
HTMLlink(url, text, ...)
HTMLimg(filename = RpadPlotName(), ...)
HTMLembed(filename, width = 600, height = 600, ...)
HfromHTML(x)
print.condition(x, ...)

```

Arguments

Format	a string specifying the desired output format, like "html", "text", or "none"
text	specifies the text to be displayed in the HTML output.
tagName	is a string specifying the HTML nodeName ("H1" or "DIV" for example).
standaloneTag	is a logical that specifies whether the HTML tag is a standalone tag, meaning it has no ending tag (for example).
collapseContents	is a logical that specifies whether vector tag contents are collapsed (merged).
variableName, commonName	specify attributes of the radio element. commonName specifies the common radio elements that are grouped together (the name attribute of the radio element)—this gets translated into the R variable with the same name. variableName is the string result assigned to commonName when this radio item is selected (the value attribute of the radio element).
name	is the name attribute in HTML that is translated into the R variable with the same name.
value	is the initial value of the input box.
url	is the URL for the <A> element.
contenteditablewrapper	is a logical that specifies whether a wrapper is placed around the output to disable editing. Internet Explorer needs this to make links and input elements active.
default, size, id, optionvalue	specify parameters of a select box. default specifies which of the options is selected initially.
filename, width, height	are parameters of an IMG or EMBED object.
rpadType	is the "rpadType" attribute of an INPUT element. Normal values are either "Rvariable" or "Rstring" but other values of "rpadType" should also be possible.

```
x          == To define ==
...       arguments are passed on as HTML arguments for the given tag.
```

Details

`HTMLon` and `HTMLoff` specify how `Rpad` interprets results (either HTML or text). `Rpad` output sections are normally rendered as plain text with a fixed-width font, so text script outputs are formatted properly. The output blocks can also contain HTML codes for displaying images or for formatting blocks. To change to HTML formatting, use `HTMLon()` (which sends '`<htmlon/>`' to the output). To turn HTML mode off, use `HTMLoff()` (which sends '`<htmloff/>`'). `HTMLon` and `HTMLoff` only apply to the existing `Rpad` input section; they don't carry over into the next.

`HtmlTree` and its shortcut `H` generates HTML (or other XML-like syntax) for an arbitrary tag with the specified name and arguments. Unnamed parameters to `H` are content, and named parameters are tag attributes. By using nesting, it's a good way to generate HTML (or any XML) with properly formed tags. If `tagName` is `NULL`, then the contents are created without a surrounding tag. For vectors, each vector element becomes surrounded by tags, so `H("div", c(1, 2, 3))` results in `<div>1</div><div>2</div><div>3</div>`.

`Html` converts an R object to class `HtmlTree`. Methods are defined for matrices and data frames. For data frames, all ... arguments are passed to `format` before conversion to an HTML representation. Unnamed parameters to `H` that have a method defined for `Html` are automatically converted. For example, `H("div", data.frame(a=1:2, b=1:2))` is the HTML representation of the data frame wrapped by a `div` tag.

For conversion of a wider selection of R objects to HTML, use `HTML` from the `R2HTML` package. `HTML` differs from `Html` in that `HTML` uses `cat` to directly send output whereas `Html` returns a character string. `HfromHTML` captures the output of `HTML` and returns it as a character string of class `HtmlTree`. This is useful for nesting HTML inside of `H` trees.

`ROutputMode` changes how R behaves with automatic printing. Possible values are: "text" (the default), "html", or "none". "text" is like at the command line: values returned in the script are automatically printed (without an explicit `print` statement) in standard text format. With "html", values returned are automatically printed, but HTML output is generated by using the `HTML` method from `R2HTML` (if available) instead of the `print` method. `ROutputMode` applies to all subsequent `Rpad` input section, including a rollover back to the beginning when a page is run several times.

Value

All of the HTML code generation routines return a character string of class `HtmlTree`. With automatic printing, the string is sent to the output (with `cat`). This has the effect that the HTML is displayed in `Rpad`. Note that if a HTML generation function is used inside of a loop or other scenario where the function results are not automatically printed, then you need to enclose the function with `cat` or `print`.

`HTMLargs` returns a string with the arguments as "a='arg1' b='arg2'", and so on. Note the use of single quotes. This will affect how quoted strings should be passed as elements.

Author(s)

Tom Short, EPRI, ((tshort@epri.com))

See Also

R2HTML for other useful HTML routines that can be used in Rpad.

Examples

```
a <- data.frame(x = 1:10, y = (1:10)^3)

# generate some GUI elements
# - normally done in an Rpad input section with rpadRun="init"
HTMLon() # switch to html mode
data(state)
HTMLselect("favoriteAmericanState", state.name) # generate the select box

H("div", "Hello world") # a simple div: "<div>Hello world</div>"
H("div", class="helloStyle", "Hello world") # a div with a class
# --> "<div class='helloStyle'>Hello world</div>"

# you can nest them:
H("div", class = "myClass", dojoType = "tree",
  "This is some text in the div ",
  H("em", "emphasized"), "plus some more",
  "and more",
  H("div", class = "anotherClass",
    "text in the div",
    c(1,5,8)))

HTMLoff() # switch to text mode
summary(a)

# fancy HTML output with R2HTML
if (require(R2HTML)) {
  HTMLon()
  .HTML.file = "" # not needed in normal use
  # kludge to pass R CMD check
  HTML(summary(a))
  HTMLoff()
}
```

RpadLocalServer *Rpad local server*

Description

Functions to implement Rpad locally.

Rpad is an interactive, web-based analysis program. Rpad pages are interactive workbook-type sheets based on R, an open-source implementation of the S language. Rpad is an analysis package, a web-page designer, and a gui designer all wrapped in one. Rpad makes it easy to develop powerful data analysis applications that can be shared with others.

Rpad is available in two versions: a local version and an intranet/internet version. The local version works through the user's local installation of R with through the user's web browser. The intranet/internet version works in client-server fashion with the user accessing a remote server through a standard web browser.

Usage

```
Rpad(file = "", defaultfile = "LocalDefault.Rpad", port = 8079)
startRpadServer(defaultfile = "LocalDefault.Rpad", port = 8079)
stopRpadServer()
```

Arguments

<code>file</code>	the file to load into the browser.
<code>defaultfile</code>	the default filename for the minihttpd server to serve if the URL does not specify one.
<code>port</code>	the TCP port of the server (8079 by default).

Details

`Rpad()` starts the local Rpad server and launches the default browser with a default startup page. Use the `file` argument to specify a different starting page. The default startup page allows the user to select any of the Rpad html files in R's current working directory. You can also use `startRpadServer()` to start the server (without launching another browser window) and `stopRpadServer()` to stop the server.

The Rpad local server implements a mini-httpd, a minimal web-page server. This mini server is implemented in Tcl/Tk, using the powerful 'socket' command. Since it runs in the separate tcltk event loop, it is not blocking R, and it runs in the background; the user can still enter commands at the R prompt. The user can use Rpad along side of other user interfaces, including Sciviews-R, Rgui, ESS, and/or Rcmdr.

The mini-httpd server first looks for files relative to R's current working directory. If it can't find them there, it looks for files relative to the "basehtml" directory in the Rpad package directory. This way, the user can store Rpad html files wherever he wants and not have to worry about carrying around the javascript, CSS, and other html-related files.

The original implementation of Rpad uses a classical web server like Apache with perl scripts. It is working as a client-server through Intra/Internet (look at <http://www.Rpad.org>) for a live example.

A number of R utility functions are provided for accessing directories and URL's that should keep compatibility between the local version and the client/server version (`RpadURL`, `RpadBaseURL`, and `RpadBaseFile`).

Note

For security reasons, the server can only run for a local client. However, it is very easy to eliminate this limitation by hacking the `startSocketServer()` function in the **svSocket** package (SciViews bundle).

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org) and Tom Short, EPRI Solutions, Inc., (tshort@eprisolutions.com)

See Also

See `link{RpadServer}` for information on the client/server version of Rpad. For utility functions for compatibility between the local and client/server versions of Rpad, see `RpadURL`, `RpadBaseURL`, and `RpadBaseFile`.

RpadUtil

Rpad utilities

Description

Rpad utilities to generate filenames or URL's

Usage

```
RpadURL(filename = "")
RpadBaseURL(filename = "")
RpadBaseFile(filename = "")
RpadIsLocal()
```

Arguments

`filename` the name of a file.

Value

`RpadURL` returns the URL for the given filename: `"/filename"` for the local version of Rpad and `"/Rpad/server/dd??????/filename"` for the server version. Use this to output HTML links for the user.

`RpadBaseURL` returns the base URL: `"filename"` for the local version and `"/Rpad/filename"` for the client-server version. Use this to point the user to data files or other links on the server that is somewhere permanent. (The current R working directory is not permanent in the client-server version.)

`RpadBaseFile` returns the file name relative to the base R directory: `"filename"` for the local version and `"../filename"` for the client-server version. Use this in R to read in data files or save data files somewhere permanent.

Author(s)

Tom Short, EPRI Solutions, Inc., (tshort@eprisolutions.com)

See Also

Rpad, RpadHTML

Examples

```
## Not run:
# make some data
x <- 1:10
y2 <- x^3
save(x, y2, file = RpadBaseFile("testdata.RData"))
# output a link to the user:
HTMLon()
cat("<a href='", RpadBaseURL("testdata.RData"), sep=""")
cat(">Click</a> to download the test data.")
unlink(RpadBaseFile("testdata.RData"))
## End(Not run)
```

Index

*Topic **IO**

RpadLocalServer, 8

*Topic **math**

json, 2

RpadGraphing, 3

RpadHTML, 5

RpadUtil, 10

BR (*RpadHTML*), 5

graphoptions (*RpadGraphing*), 3

H (*RpadHTML*), 5

HfromHTML (*RpadHTML*), 5

Html (*RpadHTML*), 5

HTMLargs (*RpadHTML*), 5

HTMLcheckbox (*RpadHTML*), 5

HTMLembed (*RpadHTML*), 5

HTMLetag (*RpadHTML*), 5

HTMLh1 (*RpadHTML*), 5

HTMLh2 (*RpadHTML*), 5

HTMLh3 (*RpadHTML*), 5

HTMLh4 (*RpadHTML*), 5

HTMLh5 (*RpadHTML*), 5

HTMLimg (*RpadHTML*), 5

HTMLinput (*RpadHTML*), 5

HTMLlink (*RpadHTML*), 5

HTMLoff (*RpadHTML*), 5

HTMLon (*RpadHTML*), 5

HTMLradio (*RpadHTML*), 5

HTMLselect (*RpadHTML*), 5

HTMLtag (*RpadHTML*), 5

HtmlTree (*RpadHTML*), 5

json, 2

newgraph (*RpadGraphing*), 3

print.condition (*RpadHTML*), 5

ROutputFormat (*RpadHTML*), 5

Rpad (*RpadLocalServer*), 8

RpadBaseFile (*RpadUtil*), 10

RpadBaseURL (*RpadUtil*), 10

RpadGraphing, 3

RpadHTML, 5

RpadIsLocal (*RpadUtil*), 10

RpadLocalServer, 8

RpadPlotName (*RpadGraphing*), 3

RpadURL (*RpadUtil*), 10

RpadUtil, 10

showgraph (*RpadGraphing*), 3

startRpadServer

(*RpadLocalServer*), 8

stopRpadServer (*RpadLocalServer*),

8