Package 'R.rsp'

December 6, 2016

Version 0.40.0

Depends R (>= 2.14.0)

Imports methods, stats, tools, utils, R.methodsS3 (>= 1.7.1), R.oo (>= 1.21.0), R.utils, R.cache (>= 0.12.0)

Suggests tcltk, digest (>= 0.6.10), ascii (>= 2.1), markdown (>= 0.7.7), knitr (>= 1.9), R.devices (>= 2.15.1), base64enc (>= 0.1-3)

SuggestsNote Recommended: digest, R.devices, base64enc, markdown

VignetteBuilder R.rsp

Date 2016-12-05

Title Dynamic Generation of Scientific Reports

Author Henrik Bengtsson [aut, cre, cph]

Maintainer Henrik Bengtsson <henrikb@braju.com>

Description The RSP markup language makes any text-based document come alive. RSP provides a powerful markup for controlling the content and output of LaTeX, HTML, Markdown, AsciiDoc, Sweave and knitr documents (and more), e.g. 'Today's date is <%=Sys.Date()%>'. Contrary to many other literate programming languages, with RSP it is straightforward to loop over mixtures of code and text sections, e.g. in month-by-month summaries. RSP has also several preprocessing directives for incorporating static and dynamic contents of external files (local or online) among other things. Functions rstring() and rcat() make it easy to process RSP strings, rsource() sources an RSP file as it was an R script, while rfile() compiles it (even online) into its final output format, e.g. rfile('report.tex.rsp') generates 'report.pdf' and rfile('report.md.rsp') generates 'report.html'. RSP is ideal for self-contained scientific reports and R package vignettes. It's easy to use - if you know how to write an R script, you'll be up and running within minutes.

License LGPL (>= 2.1)

URL https://github.com/HenrikBengtsson/R.rsp

BugReports https://github.com/HenrikBengtsson/R.rsp/issues

LazyLoad TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-06 07:52:32

R topics documented:

`R.rsp-package` *Package R.rsp*

Description

The RSP markup language makes any text-based document come alive. RSP provides a powerful markup for controlling the content and output of LaTeX, HTML, Markdown, AsciiDoc, Sweave and knitr documents (and more), e.g. 'Today's date is <%=Sys.Date()%>'. Contrary to many other literate programming languages, with RSP it is straightforward to loop over mixtures of code and text sections, e.g. in month-by-month summaries. RSP has also several preprocessing directives for incorporating static and dynamic contents of external files (local or online) among other things. Functions rstring() and rcat() make it easy to process RSP strings, rsource() sources an RSP file as it was an R script, while rfile() compiles it (even online) into its final output format, e.g. rfile('report.tex.rsp') generates 'report.pdf' and rfile('report.md.rsp') generates 'report.html'. RSP is ideal for self-contained scientific reports and R package vignettes. It's easy to use - if you know how to write an R script, you'll be up and running within minutes.

Installation

To install this package, call `install.packages("R.rsp")`.

To get started

We recommend that you start by reading one of the 'vignettes';

1. A 5 minute slideshow covering the basics of RSP.
2. Detailed description of the RSP markup language.
3. A one-page RSP reference card.
4. How to use RSP for package vignettes.
5. How to use plain LaTeX for package vignettes.
6. How to use static PDF or HTML package vignettes.

Then, when you're ready to try it yourself, these are commands you can start with:

1. Play with `rcat`(), which works like `cat`() but also processed RSP expressions, e.g. `rcat("A random number: <%=samp`

2. To source a RSP document as you do with R scripts, use rsource(), e.g. rsource("report.md.rsp") which will run the RSP and display the output as it appears.
3. To compile a RSP document to a final document, use rfile(), e.g. rfile("report.md.rsp") outputs Markdown file 'report.md' which is automatically compiled into a final 'report.html'.

Acknowledgments

Several of the post-processing features of this package utilize packages such as **base64enc**, **ascii**, **knitr**, and **markdown**. Not enough credit can be given to the authors and contributors of those packages. Thank you for your great work.

License

How to cite this package

Bengtsson H (2016). *R.rsp: Dynamic Generation of Scientific Reports*. R package version 0.40.0, https://github.com/HenrikBengtsson/R.rsp.

Author(s)

Henrik Bengtsson

---

rcat *Evaluates an RSP string and outputs the generated string*

Description

Evaluates an RSP string and outputs the generated string.

Usage

```
 ## Default S3 method:
rcat(..., file=NULL, path=NULL, envir=parent.frame(), args="*", output="", buffered=TRUE,
  append=FALSE, verbose=FALSE)
 ## Default S3 method:
rsource(file, path=NULL, envir=parent.frame(), output="", buffered=FALSE, ...)
```

Arguments

`...`	A `character` string with RSP markup.
`file, path`	Alternatively, a file, a URL or a `connection` from with the strings are read. If a file, the `path` is prepended to the file, iff given.
`envir`	The `environment` in which the RSP string is preprocessed and evaluated.
`args`	A named `list` of arguments assigned to the environment in which the RSP string is parsed and evaluated. See `cmdArgs`.
`output`	A `connection`, or a pathname where to direct the output. If `""`, the output is sent to the standard output.
`buffered`	If `TRUE`, and `output=""`, then the RSP output is outputted as soon as possible, if possible.
`append`	Only applied if `output` specifies a pathname; If `TRUE`, then the output is appended to the file, otherwise the files content is overwritten.
`verbose`	See `Verbose`.

Value

Returns (invisibly) the outputted `RspStringProduct`.

Processing RSP strings from the command line

Using `Rscript` and `rcat()`, it is possible to process an RSP string and output the result from the command line. For example,

```
Rscript -e "R.rsp::rcat('A random integer in [1,<%=K%>]: <%=sample(1:K, size=1)%>')" --args --K=50
```

parses and evaluates the RSP string and outputs the result to standard output.

rsource()

The `rsource(file, ...)` is a convenient wrapper for `rcat(file=file, ..., output="", buffered=FALSE)`. As an analogue, `rsource()` is to an RSP file what `source()` is to an R script file.

Author(s)

Henrik Bengtsson

See Also

To store the output in a string (instead of displaying it), see `rstring`(). For evaluating and postprocessing an RSP document and writing the output to a file, see `rfile`().

Examples

```
rcat("A random integer in [1,100]: <%=sample(1:100, size=1)%>\n")

# Passing arguments
rcat("A random integer in [1,<%=K%>]: <%=sample(1:K, size=1)%>\n", args=list(K=50))
```

rfile *Evaluates and postprocesses an RSP document and outputs the final RSP document file*

Description

Evaluates and postprocesses an RSP document and outputs the final RSP document file.

Usage

```
## Default S3 method:
rfile(file, path=NULL, output=NULL, workdir=NULL, type=NA, envir=parent.frame(),
  args="*", postprocess=TRUE, ..., verbose=FALSE)
```

Arguments

file, path	Specifies the RSP file to processed, which can be a file, a URL or a connection. If a file, the path is prepended to the file, iff given.
output	A character string or a connection specifying where output should be directed. The default is a file with a filename where the file extension (typically ".rsp") has been dropped from file in the directory given by the workdir argument.
workdir	The working directory to use after parsing and preprocessing, but while *evaluating* and *postprocessing* the RSP document. If argument output specifies an absolute pathname, then the directory of output is used, otherwise the current directory is used.
type	The default content type of the RSP document. By default, it is inferred from the output filename extension, iff possible.
envir	The environment in which the RSP document is preprocessed and evaluated.
args	A named list of arguments assigned to the environment in which the RSP string is parsed and evaluated. See cmdArgs.
postprocess	If TRUE, and a postprocessing method exists for the generated RSP product, it is postprocessed as well.
...	Additional arguments passed to the RSP engine.
verbose	See Verbose.

Value

Returns an RspProduct. If argument output specifies a file, then this is an RspFileProduct.

Processing RSP files from the command line

Using Rscript and rfile(), it is possible to process an RSP file from the command line. For example,

```
Rscript -e "R.rsp::rfile(file='RSP_refcard.tex.rsp', path=system.file('doc', package='R.rsp'))"
```

parses and evaluates 'RSP_refcard.tex.rsp' and output 'RSP_refcard.pdf' in the current directory.

Author(s)

Henrik Bengtsson

See Also

rstring() and rcat().

Examples

```
path <- system.file("exData", package="R.rsp")
pathname <- rfile("random.txt.rsp", path=path)
print(pathname)

lines <- readLines(pathname, warn=FALSE)
cat(lines, collapse="\n")


# Passing arguments
path <- system.file("exData", package="R.rsp")
pathname <- rfile("random-args.txt.rsp", path=path, args=list(K=50))
print(pathname)

lines <- readLines(pathname, warn=FALSE)
cat(lines, collapse="\n")


## Not run:
# Compile and display the main vignette (requires LaTeX)
if (isCapableOf(R.rsp, "latex")) {
  path <- system.file("doc", package="R.rsp")
  pdf <- rfile("Dynamic_document_creation_using_RSP.tex.rsp", path=path)
  cat("Created document: ", pdf, "\n", sep="")
  if (interactive()) browseURL(pdf)
}

## End(Not run)
```

rstring *Evaluates an RSP string and returns the generated string*

Description

Evaluates an RSP string and returns the generated string.

Usage

```
## Default S3 method:
rstring(..., file=NULL, path=NULL, envir=parent.frame(), args="*", verbose=FALSE)
```

Arguments

`...`	A `character` string with RSP markup.
`file, path`	Alternatively, a file, a URL or a `connection` from with the strings are read. If a file, the `path` is prepended to the file, iff given.
`envir`	The `environment` in which the RSP string is preprocessed and evaluated.
`args`	A named `list` of arguments assigned to the environment in which the RSP string is parsed and evaluated. See `cmdArgs`.
`verbose`	See `Verbose`.

Value

Returns an `RspStringProduct`.

Author(s)

Henrik Bengtsson

See Also

To display the output (instead of returning a string), see `rcat`(). For evaluating and postprocessing an RSP document and writing the output to a file, see `rfile`().

Examples

```
x <- rstring("A random integer in [1,100]: <%=sample(1:100, size=1)%>\n")
cat(x)

# Passing arguments
x <- rstring("A random integer in [1,<%=K%>]: <%=sample(1:K, size=1)%>\n", args=list(K=50))
cat(x)
```

Index