Package 'tinkr'

August 19, 2022

Title Cast '(R)Markdown' Files to 'XML' and Back Again

Version 0.1.0

Description Parsing '(R)Markdown' files with numerous regular expressions can
be fraught with peril, but it does not have to be this way. Converting
'(R)Markdown' files to 'XML' using the 'commonmark' package allows
in-memory editing via of 'markdown' elements via 'XPath' through the
extensible 'R6' class called 'yarn'. These modified 'XML' representations
can be written to '(R)Markdown' documents via an 'xslt' stylesheet which
implements an extended version of 'GitHub'-flavoured 'markdown' so that you
can tinker to your hearts content.

License GPL-3

URL https://docs.ropensci.org/tinkr/,
https://github.com/ropensci/tinkr

BugReports https://github.com/ropensci/tinkr/issues

Imports commonmark (>= 1.6), glue, magrittr, purrr, R6, stringr, xml2,
xslt, yaml

Suggests knitr, rmarkdown, covr, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Author Maëlle Salmon [aut] (<https://orcid.org/0000-0002-2815-0399>),
Zhian N. Kamvar [aut, cre] (<https://orcid.org/0000-0003-1458-7108>),
Jeroen Ooms [aut],
Nick Wellnhofer [cph] (Nick Wellnhofer wrote the XSLT stylesheet.),
rOpenSci [fnd] (https://ropensci.org/),
Peter Daengeli [ctb]

Maintainer Zhian N. Kamvar <zkamvar@gmail.com>

Repository CRAN

Date/Publication 2022-08-19 12:30:02 UTC

R **topics documented:**

---

find_between *Find between a pattern*

Description

Helper function to find all nodes between a standard pattern. This is useful if you want to find unnested pandoc tags.

Usage

```
find_between(
  body,
  ns,
  pattern = "md:paragraph[md:text[starts-with(text(), ':::')]]",
  include = FALSE
)
```

Arguments

body	and XML document
ns	the namespace of the document
pattern	an XPath expression that defines characteristics of nodes between which you want to extract everything.
include	if TRUE, the tags matching pattern will be included in the output, defaults to FALSE, which only gives you the nodes in between pattern.

Value

a nodeset

Examples

```
md <- glue::glue("
 h1
 ====

 ::: section

 h2
 ----

 section *text* with [a link](https://ropensci.org/)

 :::
")
x <- xml2::read_xml(commonmark::markdown_xml(md))
ns <- xml2::xml_ns_rename(xml2::xml_ns(x), d1 = "md")
res <- find_between(x, ns)
res
xml2::xml_text(res)
xml2::xml_find_all(res, ".//descendant-or-self::md:*", ns = ns)
```

---

md_ns   *Aliased namespace prefix for commonmark*

Description

The commonmark package is used to translate markdown to XML, but it does not assign a namespace prefix, which means that xml2 will auto-assign a default prefix of d1.

Usage

```
md_ns()
```

Details

This function renames the default prefix to md, so that you can use XPath queries that are slightly more descriptive.

Value

an xml_namespace object (see xml2::xml_ns())

Examples

```
tink <- tinkr::to_xml(system.file("extdata", "example1.md", package = "tinkr"))
# with default namespace
xml2::xml_find_all(tink$body,
  ".//d1:link[starts-with(@destination, 'https://ropensci')]"
```

```
)
# with tinkr namespace
xml2::xml_find_all(tink$body,
  ".//md:link[starts-with(@destination, 'https://ropensci')]",
  tinkr::md_ns()
)
```

---

protect_math   *Protect math elements from commonmark's character escape*

Description

Protect math elements from commonmark's character escape

Usage

```
protect_math(body, ns = md_ns())
```

Arguments

body	an XML object
ns	an XML namespace object (defaults: md_ns()).

Details

Commonmark does not know what LaTeX is and will LaTeX equations as normal text. This means that content surrounded by underscores are interpreted as <emph> elements and all backslashes are escaped by default. This function protects inline and block math elements that use $ and $$ for delimiters, respectively.

Value

a copy of the modified XML object

Note

this function is also a method in the yarn object.

Examples

```
m <- tinkr::to_xml(system.file("extdata", "math-example.md", package = "tinkr"))
txt <- textConnection(tinkr::to_md(m))
cat(tail(readLines(txt)), sep = "\n") # broken math
close(txt)
m$body <- protect_math(m$body)
txt <- textConnection(tinkr::to_md(m))
cat(tail(readLines(txt)), sep = "\n") # fixed math
close(txt)
```

---

stylesheet   *The tinkr stylesheet*

Description

This function returns the path to the tinkr stylesheet

Usage

```
stylesheet()
```

Value

a single element character vector representing the path to the stylesheet used by tinkr.

Examples

```
tinkr::stylesheet()
```

---

to_md   *Write YAML and XML back to disk as (R)Markdown*

Description

Write YAML and XML back to disk as (R)Markdown

Usage

```
to_md(yaml_xml_list, path = NULL, stylesheet_path = stylesheet())
```

Arguments

yaml_xml_list   result from a call to [to_xml()](#) and editing.

path   path of the new file. Defaults to NULL, which will not write any file, but will still
                 produce the conversion and pass the output as a character vector.

stylesheet_path
                 path to the XSL stylesheet

Details

The stylesheet you use will decide whether lists are built using "*" or "-" for instance. If you're
keen to keep your own Markdown style when using [to_md()](#) after [to_xml()](#), you can tweak the
XSL stylesheet a bit and provide the path to your XSL stylesheet as argument.

Value

the converted document, invisibly.

Examples

```
path <- system.file("extdata", "example1.md", package = "tinkr")
yaml_xml_list <- to_xml(path)
names(yaml_xml_list)
library("magrittr")
# transform level 3 headers into level 1 headers
body <- yaml_xml_list$body
body %>%
  xml2::xml_find_all(xpath = './/d1:heading',
                     xml2::xml_ns(.)) %>%
  .[xml2::xml_attr(., "level") == "3"] -> headers3
xml2::xml_set_attr(headers3, "level", 1)
yaml_xml_list$body <- body
# save back and have a look
newmd <- tempfile("newmd", fileext = ".md")
to_md(yaml_xml_list, newmd)
# file.edit("newmd.md")
file.remove(newmd)
```

---

to_xml   *Transform file to XML*

Description

Transform file to XML

Usage

```
to_xml(path, encoding = "UTF-8", sourcepos = FALSE, anchor_links = TRUE)
```

Arguments

path	Path to the file.
encoding	Encoding to be used by readLines.
sourcepos	passed to `commonmark::markdown_xml()`. If TRUE, the source position of the file will be included as a "sourcepos" attribute. Defaults to FALSE.
anchor_links	if TRUE (default), reference-style links with anchors (in the style of [key]: https://example.com/link will be preserved as best as possible. If this is FASLE, the anchors disappear and the links will appear as normal links. See `resolve_anchor_links()` for details.

Details

This function will take a (R)markdown file, split the yaml header from the body, and read in the body through `commonmark::markdown_xml()`. Any RMarkdown code fences will be parsed to expose the chunk options in XML and tickboxes (aka checkboxes) in GitHub-flavored markdown will be preserved (both modifications from the commonmark standard).

Math elements

Value

A list containing the YAML of the file (yaml) and its body (body) as XML.

Examples

```
path <- system.file("extdata", "example1.md", package = "tinkr")
post_list <- to_xml(path)
names(post_list)
path2 <- system.file("extdata", "example2.Rmd", package = "tinkr")
post_list2 <- to_xml(path2)
post_list2
```

---

yarn   *R6 class containing XML representation of Markdown*

Description

Wrapper around an XML representation of a Markdown document. It contains four publicly accessible slots: path, yaml, body, and ns.

Details

This class is a fancy wrapper around the results of `to_xml()` and has methods that make it easier to add, analyze, remove, or write elements of your markdown document.

Public fields

path [character] path to file on disk

yaml [character] text block at head of file

body [xml_document] an xml document of the (R)Markdown file.

ns [xml_document] an xml namespace object defining "md" to commonmark.

Methods

Public methods:

- yarn$new()
- yarn$reset()
- yarn$write()
- yarn$show()
- yarn$head()
- yarn$tail()
- yarn$add_md()
- yarn$protect_math()
- yarn$clone()

Method new(): Create a new yarn document

Usage:

```
yarn$new(path = NULL, encoding = "UTF-8", sourcepos = FALSE, ...)
```

Arguments:

path [character] path to a markdown episode file on disk

encoding [character] encoding passed to readLines()

sourcepos passed to commonmark::markdown_xml(). If TRUE, the source position of the file will be included as a "sourcepos" attribute. Defaults to FALSE.

... arguments passed on to to_xml().

Returns: A new yarn object containing an XML representation of a (R)Markdown file.

Examples:

```
path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
ex1
path2 <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex2 <- tinkr::yarn$new(path2)
ex2
```

Method reset(): reset a yarn document from the original file

Usage:

```
yarn$reset()
```

Examples:

```
path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
# OH NO
ex1$body
ex1$body <- xml2::xml_missing()
ex1$reset()
ex1$body
```

Method write(): Write a yarn document to Markdown/R Markdown

Usage:

```
yarn$write(path = NULL, stylesheet_path = stylesheet())
```

Arguments:

path path to the file you want to write

stylesheet_path path to the xsl stylesheet to convert XML to markdown.

Examples:

```
path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
ex1
tmp <- tempfile()
try(readLines(tmp)) # nothing in the file
ex1$write(tmp)
head(readLines(tmp)) # now a markdown file
unlink(tmp)
```

Method show(): show the markdown contents on the screen

Usage:

```
yarn$show(stylesheet_path = stylesheet())
```

Arguments:

stylesheet_path path to the xsl stylesheet to convert XML to markdown.

Returns: a character vector with one line for each line in the output

Examples:

```
path <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex2 <- tinkr::yarn$new(path)
ex2$head(5)
ex2$tail(5)
ex2$show()
```

Method head(): show the head of the markdown contents on the screen

Usage:

```
yarn$head(n = 6L, stylesheet_path = stylesheet())
```

Arguments:

n the number of elements to show from the top. Negative numbers

stylesheet_path path to the xsl stylesheet to convert XML to markdown. exclude lines from
    the bottom

Returns: a character vector with n elements

Method tail(): show the tail of the markdown contents on the screen

Usage:

```
yarn$tail(n = 6L, stylesheet_path = stylesheet())
```

Arguments:

n the number of elements to show from the bottom. Negative numbers

stylesheet_path path to the xsl stylesheet to convert XML to markdown. exclude lines from
    the top

Returns: a character vector with n elements

Method add_md()**:** add an arbitrary Markdown element to the document

Usage:

```
yarn$add_md(md, where = 0L)
```

Arguments:

md a string of markdown formatted text.

where the location in the document to add your markdown text. This is passed on to xml2::xml_add_child().
    Defaults to 0, which indicates the very top of the document.

Examples:

```
path <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex <- tinkr::yarn$new(path)
# two headings, no lists
xml2::xml_find_all(ex$body, "md:heading", ex$ns)
xml2::xml_find_all(ex$body, "md:list", ex$ns)
ex$add_md(
  "# Hello\n\nThis is *new* formatted text from `{tinkr}`!",
  where = 1L
)$add_md(
  " - This\n - is\n - a new list",
  where = 2L
)
# three headings
xml2::xml_find_all(ex$body, "md:heading", ex$ns)
xml2::xml_find_all(ex$body, "md:list", ex$ns)
tmp <- tempfile()
ex$write(tmp)
readLines(tmp, n = 20)
```

Method protect_math()**:** Protect math blocks from being escaped

Usage:

```
yarn$protect_math()
```

Examples:

```
path <- system.file("extdata", "math-example.md", package = "tinkr")
ex <- tinkr::yarn$new(path)
ex$tail() # math blocks are escaped :(
ex$protect_math()$tail() # math blocks are no longer escaped :)
```

Method clone()**:** The objects of this class are cloneable with this method.

Usage:

```
yarn$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `yarn$new`
## ------------------------------------------------

path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
ex1
path2 <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex2 <- tinkr::yarn$new(path2)
ex2


## ------------------------------------------------
## Method `yarn$reset`
## ------------------------------------------------


path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
# OH NO
ex1$body
ex1$body <- xml2::xml_missing()
ex1$reset()
ex1$body


## ------------------------------------------------
## Method `yarn$write`
## ------------------------------------------------

path <- system.file("extdata", "example1.md", package = "tinkr")
ex1 <- tinkr::yarn$new(path)
ex1
tmp <- tempfile()
try(readLines(tmp)) # nothing in the file
ex1$write(tmp)
head(readLines(tmp)) # now a markdown file
unlink(tmp)


## ------------------------------------------------
## Method `yarn$show`
## ------------------------------------------------

path <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex2 <- tinkr::yarn$new(path)
ex2$head(5)
ex2$tail(5)
ex2$show()


## ------------------------------------------------
## Method `yarn$add_md`
## ------------------------------------------------
```

```
path <- system.file("extdata", "example2.Rmd", package = "tinkr")
ex <- tinkr::yarn$new(path)
# two headings, no lists
xml2::xml_find_all(ex$body, "md:heading", ex$ns)
xml2::xml_find_all(ex$body, "md:list", ex$ns)
ex$add_md(
  "# Hello\n\nThis is *new* formatted text from `{tinkr}`!",
  where = 1L
)$add_md(
  " - This\n - is\n - a new list",
  where = 2L
)
# three headings
xml2::xml_find_all(ex$body, "md:heading", ex$ns)
xml2::xml_find_all(ex$body, "md:list", ex$ns)
tmp <- tempfile()
ex$write(tmp)
readLines(tmp, n = 20)


## ------------------------------------------------
## Method `yarn$protect_math`
## ------------------------------------------------

path <- system.file("extdata", "math-example.md", package = "tinkr")
ex <- tinkr::yarn$new(path)
ex$tail() # math blocks are escaped :(
ex$protect_math()$tail() # math blocks are no longer escaped :)
```

Index