# Package 'rebook'

March 30, 2021

**Version** 1.0.0

**Date** 2020-10-19

**Title** Re-using Content in Bioconductor Books

**Description** Provides utilities to re-use content across chapters of a Bioconductor book. This is mostly based on functionality developed while writing the OSCA book, but generalized for potential use in other large books with heavy compute. Also contains some functions to assist book deployment.

**Imports** utils, methods, knitr, callr, rmarkdown, CodeDepends, BiocStyle

**Suggests** testthat, igraph, BiocManager

**License** GPL-3

**VignetteBuilder** knitr

**biocViews** Software, Infrastructure, ReportWriting

**RoxygenNote** 7.1.1

**git_url** https://git.bioconductor.org/packages/rebook

**git_branch** RELEASE_3_12

**git_last_commit** 93381c3

**git_last_commit_date** 2020-10-27

**Date/Publication** 2021-03-29

**Author** Aaron Lun [aut, cre, cph]

**Maintainer** Aaron Lun <infinite.monkeys.with.keyboards@gmail.com>

## R topics documented:

buildChapterGraph . . . . . 2
chapterPreamble . . . . . 3
compileChapter . . . . . 4
createMakefile . . . . . 5
deployCustomCSS . . . . . 6
extractCached . . . . . 7
openingDetails . . . . . 9
prettySessionInfo . . . . . 10
scrapeDependencies . . . . . 11
setupHTML . . . . . 12
updateDependencies . . . . . 12

**Index** **15**

---

`buildChapterGraph` *Build the chapter dependency graph*

---

## Description

Build the dependency graph between chapter based on their `extractCached` calls to each other.

## Usage

```
buildChapterGraph(dir, recursive = TRUE, pattern = "\\.Rmd$")
```

## Arguments

| | |
|---|---|
| `dir` | String containing the path to the directory containing Rmarkdown reports. This is searched recursively for all files ending in `".Rmd"`. |
| `recursive` | Further arguments to pass to `list.files` when searching for Rmarkdown reports. |
| `pattern` | Further arguments to pass to `list.files` when searching for Rmarkdown reports. |

## Value

A directed graph object from the **igraph** package, where each node is a chapter and is connected to its dependencies by an edge.

## Author(s)

Aaron Lun

## Examples

```
dir <- tempfile()
dir.create(dir)

tmp1 <- file.path(dir, "alpha.Rmd")
write(file=tmp1, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
rodan <- 1
```")

tmp2 <- file.path(dir, "bravo.Rmd")
write(file=tmp2, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
extractCached('alpha.Rmd')
```")
```

```
# Building the chapter graph:
g <- buildChapterGraph(dir)
plot(g)
```

---

`chapterPreamble` *Execute chapter preamble code*

---

## Description

Execute code to set up the compilation environment at the start of every chapter.

## Usage

```
chapterPreamble(cache = TRUE)
```

## Arguments

| | |
|---|---|
| cache | Logical indicating whether to cache code chunks. |

## Details

Compilation is performed with no tolerance for errors, no printing of package start-up messages, and no printing of warnings.

Numbers are printed to 4 digits of precision.

The **BiocStyle** package is automatically attached, primarily for use of `Biocpkg` and similar functions.

HTML elements are defined using `setupHTML`.

## Value

HTML is printed to standard output, see `setupHTML`.

## Author(s)

Aaron Lun

## Examples

````
tmp <- tempfile(fileext=".Rmd")
write(file=tmp, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
pi # four digits!
```

```{r}
warning('ASDASD') # warnings and messages are not saved in the HTML.
```
````
````

```
```{r, results='asis'}
prettySessionInfo()
```")

rmarkdown::render(tmp)

if (interactive()) browseURL(sub(".Rmd$", ".html", tmp))
```

---

`compileChapter` *Compile a Rmarkdown file*

---

## Description

Compile a Rmarkdown file - typically a chapter of a book - so that `extractCached` calls work correctly in other chapters.

## Usage

```
compileChapter(path, cache = TRUE)
```

## Arguments

| | |
|---|---|
| `path` | String containing a path to an Rmarkdown file. |
| `cache` | Logical scalar indicating whether the compilation should be cached. |

## Details

Compilation is performed in an isolated session using `r` from the **callr** package. This ensures that settings from one chapter do not affect the next chapter.

If an error is encountered during compilation of any Rmarkdown file, the standard output of `render` leading up to the error is printed out before the function exists.

## Value

The specified file is (re)compiled to generate the corresponding `*_cache` directories. `NULL` is invisibly returned.

## Author(s)

Aaron Lun

## See Also

`extractCached`, which calls this function.

## Examples

```
tmp <- tempfile(fileext=".Rmd")
write(file=tmp, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
rodan <- 1
```")

compileChapter(tmp)

file.exists(sub(".Rmd$", ".html", tmp)) # output HTML exists.
file.exists(sub(".Rmd$", "_cache", tmp)) # output cache exists.
exists("rodan") # FALSE
```

---

`createMakefile` *Create a compilation Makefile*

---

## Description

Create a Makefile for compiling individual chapters, in a manner that respects the dependencies between chapters.

## Usage

```
createMakefile(dir = ".", pattern = "\\.Rmd$", ..., fname = "Makefile")
```

## Arguments

| | |
|---|---|
| `dir` | String containing the path to the directory containing Rmarkdown reports. This is searched recursively for all files ending in `".Rmd"`. |
| `pattern` | Further arguments to pass to `list.files` when searching for Rmarkdown reports. |
| `...` | Further arguments to pass to `buildChapterGraph`. |
| `fname` | String containing the name of the output Makefile. |

## Details

The main benefit of using a Makefile is that the generation of the chapter caches can be done in parallel. Then, the **bookdown** step can just serially retrieve the cache contents for rapid rendering.

The Makefile uses the markdown output file as an indicator of successful `knit`ting of a chapter. Caches are left in the current working directory after the compilation of each report. It is assumed that **bookdown**'s `render_book` is smart enough to find and use these caches.

## Value

A Makefile is created in `dir` with the name `fname` and a `NULL` is invisibly returned.

## Author(s)

Aaron Lun
```

## See Also

buildChapterGraph, to detect dependencies between chapters.

## Examples

```
dir <- tempfile()
dir.create(dir)

tmp1 <- file.path(dir, "alpha.Rmd")
write(file=tmp1, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
rodan <- 1
```")

tmp2 <- file.path(dir, "bravo.Rmd")
write(file=tmp2, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
extractCached('alpha.Rmd')
```")

# Creating the Makefile:
createMakefile(dir)
cat(readLines(file.path(dir, "Makefile")), sep="\n")
```

---

# deployCustomCSS *Deploy a custom CSS*

---

## Description

Deploy a custom CSS to change the colors of the book's section headers, mostly to add some flavor to the book.

## Usage

```
deployCustomCSS(path = "style.css", h2.col = "#87b13f", h3.col = "#1a81c2")
```

## Arguments

| | |
|---|---|
| path | String containing the path to the output CSS file. |
| h2.col | String containing the color to use for the section headers. |
| h3.col | String containing the color to use for the subsection headers. |

## Details

We quickly learned that it was unwise to be too adventurous with the colors. In particular, changing the colors of the table of contents was quite distracting. Altering the colors of the section headers provides a tasteful level of customization, with the default colors set (almost) to the Bioconductor color palette.

## Value

The CSS file is overwritten at `path`. A `NULL` is invisibly returned.

## Author(s)

Aaron Lun, based on work by Rob Amezquita and Kevin Rue-Albrecht

## Examples

```
fname <- tempfile(fileext=".css")
deployCustomCSS(fname)
cat(readLines(fname), sep="\n")
```

---

# extractCached *Extract cached objects*

---

## Description

Extract specific R objects from the knitr cache of a previously compiled Rmarkdown file (the "donor") so that it can be used in the compilation process of another Rmarkdown file (the "acceptor").

## Usage

```
extractCached(path, chunk, objects, envir = topenv(parent.frame()))
```

## Arguments

| | |
|---|---|
| `path` | String containing the path to the donor Rmarkdown file. |
| `chunk` | String containing the name of the requested chunk. |
| `objects` | Character vector containing variable names for one or more objects to be extracted. |
| `envir` | Environment where the loaded objects should be stored. |

## Details

Each R object is extracted in its state at the requested `chunk` and inserted into `envir`. Note that the object does not have to be generated or even referenced in `chunk`, provided it was generated in a previous chunk.

The parser in this function is rather limited, so the donor Rmarkdown file is subject to several constraints:

- All chunks involved in generating the requested objects (indirectly or otherwise) should be named.
- All named chunks should be executed; `eval=FALSE` is not respected.
- All relevant code occurs within triple backticks, i.e., any inline code should be read-only.

Unnamed chunks are allowed but cannot be referenced and will not be shown in the output of this function. This should not be used for code that might affect variables in the named chunks, i.e., code in unnamed chunks should be "read-only" with respect to variables in the named chunks. Chunks with names starting with `unref-` are considered to be the same as unnamed chunks and will be ignored; this is useful for figure-generating chunks that need to be referenced inside the donor report.

Obviously, this entire process assumes that donor report has already been compiled with `cache=TRUE`. If not, `extractCached` will compile it (and thus generate the cache) using `compileChapter`.

## Value

Variables with names `objects` are created in `envir`. A markdown chunk (wrapped in a collapsible element) is printed that contains all commands needed to generate those objects, based on the code in the named chunks of the donor Rmarkdown file.

## Author(s)

Aaron Lun

## See Also

`setupHTML` and `chapterPreamble`, to set up the code for the collapsible element.

`compileChapter`, to compile a Rmarkdown report to generate the cache.

## Examples

````
# Mocking up an Rmarkdown report.
donor <- tempfile(fileext=".Rmd")
write(file=donor, "```{r some-monsters}
destoroyah <- 1
mecha.king.ghidorah <- 2
```

```{r more-monsters}
space.godzilla <- 3
```

```{r}
msg <- 'I am not referenced.'
```

```{r unref-figure}
plot(1, 1, main='I am also not referenced.')
```

```{r even-more-monsters}
megalon <- 4
```")
````
````

```
# Extracting stuff from it in another report.
acceptor <- tempfile(fileext=".Rmd")
dpb <- deparse(basename(donor))
write(file=acceptor, sprintf("```{r, echo=FALSE, results='asis'}
chapterPreamble()
```

```{r, results='asis', echo=FALSE}
extractCached(%s, chunk='more-monsters',
   objects=c('space.godzilla', 'destoroyah'))
```

```{r}
space.godzilla * destoroyah
```

```{r, results='asis', echo=FALSE}
extractCached(%s, chunk='even-more-monsters',
   objects=c('megalon', 'mecha.king.ghidorah'))
```

```{r}
mecha.king.ghidorah * megalon
```", dpb, dpb))

rmarkdown::render(acceptor)
if (interactive()) browseURL(sub(".Rmd$", ".html", acceptor))
```

---

`openingDetails` *Report opening details about the book*

---

## Description

Report opening details about the book, to be executed as an R expression in the `Date:` field.

## Usage

```
openingDetails(...)
```

## Arguments

| | |
|---|---|
| `...` | Further named strings to be included in the opening details. |

## Details

It is usually sufficient to set something like

```
date: "`r rebook::openingDetails()`"
```

in the YAML header of the book, thereby ensuring that the book details are printed after the title but before any contents. This assumes that none of the details have problematic characters, particularly double quotes.

Details are extracted from a `DESCRIPTION` file in the current or any parent directory. This assumes that authors are formatted as `Authors@R` and the `License` and `Date` fields are specified.
```

## Value

A string containing the formatted details for inclusion into a YAML header.

## Author(s)

Aaron Lun

## Examples

```
wd <- getwd()
setwd(file.path(R.home(), 'library', 'rebook'))
cat(openingDetails(), '\n')
setwd(wd)
```

---

`prettySessionInfo` *Pretty session info*

---

## Description

Wraps the session information output chunk in a collapsible HTML element so that it doesn't dominate the compiled chapter.

## Usage

```
prettySessionInfo()
```

## Value

Prints a HTML block containing a collapsible section with session information.

## Author(s)

Aaron Lun

## See Also

`setupHTML` and `chapterPreamble`, to set up the code for the collapsible element.

## Examples

```
tmp <- tempfile(fileext=".Rmd")
write(file=tmp, "```{r, echo=FALSE, results='asis'}
rebook::setupHTML()
```

```{r, results='asis'}
prettySessionInfo()
```")

rmarkdown::render(tmp)

if (interactive()) browseURL(sub(".Rmd$", ".html", tmp))
```
```

`scrapeDependencies` *Scrape dependencies*

## Description

Scrape Rmarkdown reports in the book for all required dependencies.

## Usage

```
scrapeDependencies(dir, recursive = TRUE, pattern = "\\.Rmd$")
```

## Arguments

| | |
|---|---|
| `dir` | String containing the path to the directory containing Rmarkdown reports. This is searched recursively for all files ending in `".Rmd"`. |
| `recursive, pattern` | Further arguments to pass to `list.files` when searching for Rmarkdown reports. |

## Details

The output of this should be added to the `Suggests` field of the book's `DESCRIPTION`, to make it easier to simply install all of its required dependencies.

Note that dependencies in inline code sections are not detected, so these should be explicitly mentioned in a standalone code chunk to be captured.

## Value

Character vector of required packages.

## Author(s)

Aaron Lun

## Examples

```
tmp <- tempfile(fileext=".Rmd")
write(file=tmp, "```{r}
A::a()
```

```{r}
library(B)
require(C)
```")

scrapeDependencies(tempdir())
```
```

## setupHTML *Set up HTML elements*

### Description

Set up Javascript and CSS elements for each chapter, primarily for the custom collapsible class.

### Usage

```
setupHTML()
```

### Details

The custom collapsible class allows us to hide details until requested by the user. This improves readability by reducing the clutter in the compiled chapter.

### Value

Prints HTML to standard output set up JS and CSS elements.

### Author(s)

Aaron Lun

### See Also

chapterPreamble, which calls this function.

extractCached and prettySessionInfo, which use the custom collapsible class.

### Examples

```
setupHTML()
```

## updateDependencies *Update the dependencies*

### Description

Update the book's DESCRIPTION file with the latest dependencies.

### Usage

```
updateDependencies(
  dir = ".",
  extra = NULL,
  indent = 4,
  field = "Suggests",
  ...
)
```

## Arguments

| | |
|---|---|
| dir | String containing the path to the directory containing the book DESCRIPTION file. |
| extra | Character vector of extra packages to be added to imports, usually from packages that are in Suggests and thus not caught directly by scrapeDependencies. |
| indent | Integer scalar specifying the size of the indent to use when listing packages. |
| field | String specifying the dependency field to store the packages in. Defaults to "Suggests" by convention. |
| ... | Further arguments to pass to scrapeDependencies. |

## Details

The book DESCRIPTION is useful for quick installation of all packages required across all chapters. For example, it is used by https://github.com/LTLA/TrojanBookBuilder to populate a trojan package's dependencies, ensuring that all packages are available when the book itself is compiled.

## Value

The specified field in the DESCRIPTION file in dir is updated. NULL is invisibly returned.

## Author(s)

Aaron Lun

## Examples

```
dir <- tempfile()
dir.create(dir)

write(file=file.path(dir, "DESCRIPTION"),
"Package: son.of.godzilla
Version: 0.0.1
Description: Like godzilla, but smaller.")

tmp <- file.path(dir, "alpha.Rmd")
write(file=tmp, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
A::func
library(C)
```")

tmp <- file.path(dir, "bravo.Rmd")
write(file=tmp, "```{r, echo=FALSE, results='asis'}
rebook::chapterPreamble()
```

```{r}
require(D)
B::more
```")
```

```
updateDependencies(dir)
cat(readLines(file.path(dir, "DESCRIPTION")), sep="\n")
```

# Index

Biocpkg, *3*
buildChapterGraph, 2, *5, 6*

chapterPreamble, 3, *8*, *10*, *12*
compileChapter, 4, *8*
createMakefile, 5

deployCustomCSS, 6

extractCached, *2*, *4*, 7, *12*

graph, *2*

knit, *5*

list.files, *2*, *5*, *11*

openingDetails, 9

prettySessionInfo, 10, *12*

r, *4*
render, *4*

scrapeDependencies, 11, *13*
setupHTML, *3*, *8*, *10*, 12

updateDependencies, 12