

Package: simpleSingleCell (via r-universe)

October 4, 2026

Title A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor

Version 1.36.0

Date 2020-08-23

Description Once a proud workflow package, this is now a shell of its former self. Almost all of its content has been cannibalized for use in the ``Orchestrating Single-Cell Analyses with Bioconductor'' book at <https://osca.bioconductor.org>. Most vignettes here are retained as reminders of the glory that once was, also providing redirection for existing external links to the relevant OSCA book chapters.

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

Suggests readxl, R.utils, SingleCellExperiment, scater, scan, limma, BiocFileCache, org.Mm.eg.db

License Artistic-2.0

Encoding UTF-8

LazyData true

VignetteBuilder knitr

biocViews ImmunoOncologyWorkflow, Workflow, SingleCellWorkflow

Workflow True

NeedsCompilation no

URL <https://www.bioconductor.org/help/workflows/simpleSingleCell/>

RoxygenNote 7.1.0

Config/pak/sysreqs cmake make libuv1-dev libxml2-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:43:17 UTC

RemoteUrl <https://github.com/bioc/simpleSingleCell>

RemoteRef RELEASE_3_23

RemoteSha 7f14a9ffbb028f4071ef1165cd1b4236d630176b

Contents

chapterPreamble	2
compileChapter	3
extractCached	4
prettySessionInfo	6
scrapeDependencies	7
setupHTML	8
Index	10

chapterPreamble	<i>Execute chapter preamble code</i>
-----------------	--------------------------------------

Description

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

Usage

chapterPreamble(cache = FALSE)

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.

HTML elements are defined using [setupHTML](#).

Value

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**

All files are (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
```

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 = toparent(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}
destructorah <- 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))

```

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()
```

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	<i>Scrape dependencies</i>
--------------------	----------------------------

---

**Description**

Scrape Rmarkdown reports in the book for all required dependencies.

**Usage**

```
scrapeDependencies(dir)
```

**Arguments**

dir	String containing the path to the directory containing Rmarkdown reports. This is searched recursively for all files ending in ".Rmd".
-----	----------------------------------------------------------------------------------------------------------------------------------------

**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 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 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 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**

```
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))
```

Index

chapterPreamble, [2](#), [5](#), [7](#), [8](#)

compileChapter, [3](#), [5](#)

extractCached, [3](#), [4](#), [4](#), [8](#)

prettySessionInfo, [6](#), [8](#)

r, [3](#)

render, [3](#)

scrapeDependencies, [7](#)

setupHTML, [2](#), [5](#), [7](#), [8](#)