Package 'blogdown'

January 9, 2019

Type Package

Title Create Blogs and Websites with R Markdown

Version 0.10

Description Write blog posts and web pages in R Markdown. This package supports the static site generator 'Hugo' (<https://gohugo.io>) best, and it also supports 'Jekyll' (<http://jekyllrb.com>) and 'Hexo' (<https://hexo.io>).

License GPL-3

Imports rmarkdown, bookdown (>= 0.9), knitr (>= 1.20), htmltools, yaml (>= 2.1.19), httpuv (>= 1.4.0), xfun (>= 0.4), servr (>= 0.11)

Suggests testit, shiny, miniUI, stringr, rstudioapi, tools, processx, later

LazyData true

URL https://github.com/rstudio/blogdown

BugReports https://github.com/rstudio/blogdown/issues

SystemRequirements Hugo (<https://gohugo.io>) and Pandoc (<https://pandoc.org>)

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author Yihui Xie [aut, cre] (<https://orcid.org/0000-0003-0645-5666>), Beilei Bian [ctb], Forest Fang [ctb], Garrick Aden-Buie [ctb], Hiroaki Yutani [ctb], Ian Lyttle [ctb], JJ Allaire [ctb], Kevin Ushey [ctb], Leonardo Collado-Torres [ctb], Xianying Tan [ctb], RStudio Inc [cph]

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2019-01-09 05:50:03 UTC

R topics documented:

blogdown 2
build_dir 2
build_site 3
dep_path 4
find_yaml 4
html_page 5
hugo_cmd 6
install_hugo 9
install_theme 10
serve_site 10
shortcode 11

Index **13**

`blogdown` *The* **blogdown** *package*

Description

The comprehensive documentation of this package is the book **blogdown: Creating Websites with R Markdown** (https://bookdown.org/yihui/blogdown/). You are expected to read at least the first chapter. If you are really busy or do not care about an introduction to **blogdown** (e.g., you are very familiar with creating websites), set your working directory to an empty directory, and run `blogdown::new_site()` to get started right away.

Examples

```
if (interactive()) blogdown::new_site()
```

`build_dir` *Build all Rmd files under a directory*

Description

List all Rmd files recursively under a directory, and compile them using `rmarkdown::render()`.

Usage

```
build_dir(dir = ".", force = FALSE, ignore = "[.]Rproj$")
```

Arguments

dir	A directory path.
force	Whether to force building all Rmd files. By default, an Rmd file is built only if it is newer than its output file(s).
ignore	A regular expression to match output filenames that should be ignored when testing if the modification time of the Rmd source file is newer than its output files.

---

build_site *Build a website*

Description

Compile all Rmd files and build the site through Hugo.

Usage

```
build_site(local = FALSE, method = c("html", "custom"), run_hugo = TRUE)
```

Arguments

local	Whether to build the website locally to be served via `serve_site()`.
method	Different methods to build a website (each with pros and cons). See Details. The value of this argument will be obtained from the global option `getOption('blogdown.method')` when it is set.
run_hugo	Whether to run `hugo_build()` after R Markdown files are compiled.

Details

You can use `serve_site()` to preview your website locally, and `build_site()` to build the site for publishing.

For the `method` argument: `method = "html"` means to render all Rmd files to HTML via `rmarkdown::render()` (which means Markdown is processed through Pandoc), and process the paths of external dependencies generated from R code chunks, including images and HTML dependencies.

For all rendering methods, a custom R script 'R/build.R' will be executed if you have provided it under the root directory of the website (e.g. you can compile Rmd to Markdown through `knitr::knit()` and build the side via `hugo_cmd()`). `method = "custom"` means it is entirely up to this R script how a website is rendered. The script is executed via command line `Rscript "R/build.R"`, which means it is executed in a separate R session. The value of the argument `local` is passed to the command line (you can retrieve the command-line arguments via `commandArgs(TRUE)`).

Note

This function recompiles all R Markdown files by default, even if the output files are newer than the source files. If you want to build the site without rebuilding all R Markdown files, you should use `hugo_build()` instead.

dep_path *A helper function to return a dependency path name*

Description

In most cases, **blogdown** can process images and HTML widgets automatically generated from code chunks (they will be moved to the `static/` folder by default), but it may fail to recognize dependency files generated to other paths. This function returns a path that you can use for your output files, so that **blogdown** knows that they should be be processed, too. It is designed to be used in a **knitr** code chunk.

Usage

```
dep_path(default = knitr::opts_chunk$get("fig.path"))
```

Arguments

`default`	Return this default value when this function is called outside of a **knitr** code chunk.

Value

A character string of the `default` value (outside **knitr**), or a path consisting of the **knitr** figure path appended by the current chunk label.

find_yaml *Find posts containing the specified metadata*

Description

Given a YAML field name, find the (R) Markdown files that contain this field and its value contains any of the specified values. Functions `find_tags()` and `find_categories()` are wrappers of `find_yaml()` with `field = 'tags'` and `field = 'categories'`, respectively; `count_fields()` returns the frequency tables of the specified YAML fields, such as the counts of tags and categories.

Usage

```
find_yaml(field = character(), value = character(), open = FALSE)

find_tags(value = character(), open = FALSE)

find_categories(value = character(), open = FALSE)

count_yaml(fields = c("categories", "tags"), sort_by_count = TRUE)
```

Arguments

field, fields	A character vector of YAML field names.
value	A vector of the field values to be matched.
open	Whether to open the matched files automatically.
sort_by_count	Whether to sort the frequency tables by counts.

Value

find_yaml() returns a character vector of filenames; count_yaml() returns a list of frequency tables.

Examples

```
library(blogdown)
find_tags(c("time-series", "support vector machine"))
find_categories("Statistics")

count_yaml(sort_by_count = FALSE)
```

---

html_page *An R Markdown output format for* **blogdown** *web pages*

Description

This function is a simple wrapper of bookdown::html_document2() with different default arguments, and more importantly, a special HTML template designed only for **blogdown** to render R Markdown to HTML pages that can be processed by Hugo.

Usage

```
html_page(..., number_sections = FALSE, self_contained = FALSE, highlight = NULL,
    template = NULL, pre_knit = NULL, post_processor = NULL)
```

Arguments

..., number_sections, self_contained, highlight, template	Arguments passed to bookdown::html_document2() (note the option theme is not supported and set to NULL internally, and when template = NULL, a default template in **blogdown** will be used).
pre_knit, post_processor	Passed to rmarkdown::output_format.

Details

The HTML output is not a complete HTML document, and only meaningful to **blogdown** (it will be post-processed to render valid HTML pages). The only purpose of this output format is for users to change options in YAML.

The fact that it is based on **bookdown** means most **bookdown** features are supported, such as numbering and cross-referencing figures/tables.

Note

Do not use a custom template unless you understand how the default template actually works (see the **blogdown** book).

The argument `highlight` does not support the value `"textmate"`, and the argument `template` does not support the value `"default"`.

References

See Chapter 2 of the **bookdown** book for the Markdown syntax: https://bookdown.org/yihui/bookdown. See the **blogdown** book for full details: https://bookdown.org/yihui/blogdown.

---

hugo_cmd *Run Hugo commands*

Description

Wrapper functions to run Hugo commands via `system2('hugo', ...)`.

Usage

```
hugo_cmd(...)

hugo_version()

hugo_build(local = FALSE)

new_site(dir = ".", install_hugo = TRUE, format = "toml", sample = TRUE,
    theme = "yihui/hugo-lithium", hostname = "github.com", theme_example = TRUE,
    empty_dirs = FALSE, to_yaml = TRUE, serve = interactive())

new_content(path, kind = "", open = interactive())

new_post(title, kind = "", open = interactive(), author = getOption("blogdown.author"),
    categories = NULL, tags = NULL, date = Sys.Date(), file = NULL, slug = NULL,
   title_case = getOption("blogdown.title_case"), subdir = getOption("blogdown.subdir",
        "post"), ext = getOption("blogdown.ext", ".md"))

hugo_convert(to = c("YAML", "TOML", "JSON"), unsafe = FALSE, ...)

hugo_server(host, port)
```

Arguments

...	Arguments to be passed to `system2('hugo', ...)`, e.g. `new_content(path)` is basically `hugo_cmd(c('new', path))` (i.e. run the command `hugo new path`).
local	Whether to build the site for local preview (if `TRUE`, all drafts and future posts will also be built, and the site configuration `baseurl` will be set to `/` temporarily).
dir	The directory of the new site. It should be empty or only contain hidden files, RStudio project ('`*.Rproj`') files, '`LICENSE`', and/or '`README`'/'`README.md`'.
install_hugo	Whether to install Hugo automatically if it is not found.
format	The format of the configuration file. Note that the frontmatter of the new (R) Markdown file created by `new_content()` always uses YAML instead of TOML.
sample	Whether to add sample content. Hugo creates an empty site by default, but this function adds sample content by default.
theme	A Hugo theme on Github (a chararacter string of the form `user/repo`, and you can optionally specify a GIT branch or tag name after `@`, i.e. `theme` can be of the form `user/repo@branch`). You can also specify a full URL to the zip file of the theme. If `theme = NA`, no themes will be installed, and you have to manually install a theme.
hostname	Where to find the theme. Defaults to `github.com`; specify if you wish to use an instance of GitHub Enterprise. You can also specify the full URL of the zip file in `theme`, in which case this argument is ignored.
theme_example	Whether to copy the example in the '`exampleSite`' directory if it exists in the theme. Not all themes provide example sites.
empty_dirs	Whether to keep the empty directories generated by Hugo.
to_yaml	Whether to convert the metadata of all posts to YAML.
serve	Whether to start a local server to serve the site.
path	The path to the new file under the '`content`' directory.
kind	The content type to create.
open	Whether to open the new file after creating it. By default, it is opened in an interactive R session.
title	The title of the post.
author	The author of the post.
categories	A character vector of category names.
tags	A character vector of tag names.
date	The date of the post.
file	The filename of the post. By default, the filename will be automatically generated from the title by replacing non-alphanumeric characters with dashes, e.g. `title = 'Hello World'` may create a file '`content/post/2016-12-28-hello-world.md`'. The date of the form `YYYY-mm-dd` will be prepended if the filename does not start with a date.

slug	The slug of the post. By default (NULL), the slug is generated from the filename by removing the date and filename extension, e.g., if file = 'post/2015-07-23-hi-there.md', slug will be hi-there. Set slug = '' if you do not want it.
title_case	A function to convert the title to title case. If TRUE, the function is tools::toTitleCase()). This argument is not limited to title case conversion. You can provide an arbitrary R function to convert the title.
subdir	If specified (not NULL), the post will be generated under a subdirectory under 'content/'. It can be a nested subdirectory like 'post/joe/'.
ext	The filename extension (e.g., '.md', '.Rmd', or '.Rmarkdown'). Ignored if file has been specified.
to	A format to convert to.
unsafe	Whether to enable unsafe operations, such as overwriting Markdown source documents. If you have backed up the website, or the website is under version control, you may try unsafe = TRUE.
host, port	The host IP address and port; see servr::server_config().

Functions

- hugo_cmd: Run an arbitrary Hugo command.
- hugo_version: Return the version number of Hugo if possible, which is extracted from the output of hugo_cmd('version').
- hugo_build: Build a plain Hugo website. Note that the function build_site() first compiles Rmd files, and then calls Hugo via hugo_build() to build the site.
- new_site: Create a new site (skeleton) via hugo new site. The directory of the new site should be empty,
- new_content: Create a new (R) Markdown file via hugo new (e.g. a post or a page).
- new_post: A wrapper function to create a new post under the 'content/post/' directory via new_content(). If your post will use R code chunks, you can set ext = '.Rmd' or the global option options(blogdown.ext = '.Rmd') in your '~/.Rprofile'. Similarly, you can set options(blogdown.author = 'Your Name') so that the author field is automatically filled out when creating a new post.
- hugo_convert: A wrapper function to convert source content to different formats via hugo convert.
- hugo_server: Start a Hugo server.

References

The full list of Hugo commands: https://gohugo.io/commands, and themes: http://themes.gohugo.io.

Examples

```
library(blogdown)
if (interactive()) new_site()
```

install_hugo *Install Hugo*

Description

Download the appropriate Hugo executable for your platform from Github and try to copy it to a system directory so **blogdown** can run the hugo command to build a site. `update_hugo()` is a wrapper of `install_hugo(force = TRUE)`.

Usage

```
install_hugo(version = "latest", use_brew = Sys.which("brew") != "", force = FALSE)

update_hugo()
```

Arguments

`version`	The Hugo version number, e.g., `0.26`; the special value `latest` means the latest version (fetched from Github releases). Alternatively, this argument can take a file path of the zip archive or tarball of the Hugo installer that has already been downloaded from Github, in which case it will not be downloaded again.
`use_brew`	Whether to use Homebrew (`https://brew.sh`) on macOS to install Hugo (recommended if you have already installed Homebrew). Note Homebrew will be automatically installed if it has not been installed when `use_brew = TRUE`.
`force`	Whether to install Hugo even if it has already been installed. This may be useful when upgrading Hugo (if you use Homebrew, run the command `brew update && brew upgrade` instead).

Details

This function tries to install Hugo to `Sys.getenv('APPDATA')` on Windows, '`~/Library/Application Support`' on macOS, and '`~/bin/`' on other platforms (such as Linux). If these directories are not writable, the package directory '`Hugo`' of **blogdown** will be used. If it still fails, you have to install Hugo by yourself and make sure it can be found via the environment variable `PATH`.

This is just a helper function and may fail to choose the correct Hugo executable for your operating system, especially if you are not on Windows or Mac or a major Linux distribution. When in doubt, read the Hugo documentation and install it by yourself: `https://gohugo.io`.

If you want to install Hugo to a custom path, you can set the global option `blogdown.hugo.dir` to a directory to store the Hugo executable before you call `install_hugo()`, e.g., `options(blogdown.hugo.dir = '~/Downloads/hugo_0.20.1/')`. This may be useful for you to use a specific version of Hugo for a specific website. You can set this option per project. See Section 1.4 Global options for details, or store a copy of Hugo on a USB Flash drive along with your website.

install_theme *Install a Hugo theme from Github*

Description

Download the specified theme from Github and install to the 'themes' directory. Available themes are listed at http://themes.gohugo.io.

Usage

```
install_theme(theme, hostname = "github.com", theme_example = FALSE,
    update_config = TRUE, force = FALSE)
```

Arguments

theme	A Hugo theme on Github (a chararacter string of the form user/repo, and you can optionally specify a GIT branch or tag name after @, i.e. theme can be of the form user/repo@branch). You can also specify a full URL to the zip file of the theme. If theme = NA, no themes will be installed, and you have to manually install a theme.
hostname	Where to find the theme. Defaults to github.com; specify if you wish to use an instance of GitHub Enterprise. You can also specify the full URL of the zip file in theme, in which case this argument is ignored.
theme_example	Whether to copy the example in the 'exampleSite' directory if it exists in the theme. Not all themes provide example sites.
update_config	Whether to update the theme option in the site configurations.
force	Whether to override the existing theme of the same name. If you have made changes to this existing theme, your changes will be lost when force = TRUE! Please consider backing up the theme by renaming it before you try force = TRUE.

serve_site *Live preview a site*

Description

The function serve_site() calls servr::httw() to start a web server, watch for changes in the site, rebuild the site if necessary, and refresh the web page automatically by default; stop_server() stops the web server.

Usage

```
serve_site(...)

stop_server()
```

Arguments

...	Arguments passed to servr::httw() (arguments dir, site.dir, baseurl, and handler have been provided, hence you cannot customize these arguments).

Details

Alternatively, you can set the global option options(blogdown.generator.server = TRUE), and serve_site() will use the web server provided by the static site generator, such as hugo_server(). This requires additional packages **processx** and **later**. You may use this option when you primarily work on plain Markdown posts instead of R Markdown posts, because it can be faster to preview Markdown posts using the web server of the static site generator. The web server will always be stopped when the R session is ended, so you may consider restarting your R session if stop_server fails to stop the server for some reason.

---

shortcode *Helper functions to write Hugo shortcodes using the R syntax*

Description

These functions return Hugo shortcodes with the shortcode name and arguments you specify. The closing shortcode will be added only if the inner content is not empty. The function shortcode_html() is essentially shortcode(.type = 'html'). The function shortcodes() is a vectorized version of shortcode().

Usage

```
shortcode(.name, ..., .content = NULL, .type = "markdown")

shortcode_html(...)

shortcodes(..., .sep = "\n")
```

Arguments

.name	The name of the shortcode.
...	All arguments of the shortcode (either all named, or all unnamed). The ... argument of shortcode_html() is passed to shortcode().
.content	The inner content for the shortcode.
.type	The type of the shortcode: markdown or html.
.sep	The separator between two shortcodes (by default, a newline).

Details

These functions can be used in either **knitr** inline R expressions or code chunks. The returned character string is wrapped in `htmltools::HTML()`, so **rmarkdown** will protect it from the Pandoc conversion. You cannot simply write `{{< shortcode >}}` in R Markdown, because Pandoc is not aware of Hugo shortcodes, and may convert special characters so that Hugo can no longer recognize the shortcodes (e.g. < will be converted to `<`).

If your document is pure Markdown, you can use the Hugo syntax to write shortcodes, and there is no need to call these R functions.

Value

A character string wrapped in `htmltools::HTML()`; `shortcode()` returns a string of the form `{{% name args %}}`, and `shortcode_html()` returns `{{< name args >}}`.

References

https://gohugo.io/extras/shortcodes/

Examples

```
library(blogdown)

shortcode("tweet", "1234567")
shortcodes("tweet", as.character(1:5))  # multiple tweets
shortcode("figure", src = "/images/foo.png", alt = "A nice figure")
shortcode("highlight", "bash", .content = "echo hello world;")

shortcode_html("myshortcode", .content = "My <strong>shortcode</strong>.")
```

Index

blogdown, 2
blogdown-package *(*blogdown*)*, 2
build_dir, 2
build_site, 3, *8*

commandArgs, *3*
count_yaml *(*find_yaml*)*, 4

dep_path, 4

find_categories *(*find_yaml*)*, 4
find_tags *(*find_yaml*)*, 4
find_yaml, 4

HTML, *12*
html_document2, *5*
html_page, 5
httw, *10*
hugo_build, *3*
hugo_build *(*hugo_cmd*)*, 6
hugo_cmd, *3*, 6
hugo_convert *(*hugo_cmd*)*, 6
hugo_server, *11*
hugo_server *(*hugo_cmd*)*, 6
hugo_version *(*hugo_cmd*)*, 6

install_hugo, 9
install_theme, 10

knit, *3*

new_content *(*hugo_cmd*)*, 6
new_post *(*hugo_cmd*)*, 6
new_site, *2*
new_site *(*hugo_cmd*)*, 6

options, *11*
output_format, *5*

render, *2*, *3*

serve_site, *3*, 10
server_config, *8*
shortcode, 11
shortcode_html *(*shortcode*)*, 11
shortcodes *(*shortcode*)*, 11
stop_server *(*serve_site*)*, 10
system2, *6*

toTitleCase, *8*

update_hugo *(*install_hugo*)*, 9