

Package ‘pkgdown’

December 7, 2018

Title Make Static HTML Documentation for a Package

Version 1.3.0

Description Generate an attractive and useful website from a source package. 'pkgdown' converts your documentation, vignettes, 'README', and more to 'HTML' making it easy to share information about your package online.

License MIT + file LICENSE

URL <https://pkgdown.r-lib.org>, <https://github.com/r-lib/pkgdown>

BugReports <https://github.com/r-lib/pkgdown/issues>

Depends R (>= 3.1.0)

Imports callr (>= 2.0.2), cli, crayon, desc, digest, evaluate, fs (>= 1.2.0), fansi, highlight, httr, magrittr, MASS, memoise, openssl, pkgload (>= 1.0.2), purrr, processx, R6, rematch2, rlang (>= 0.2.0), rmarkdown (>= 1.1.9007), roxygen2, rsconnect, rstudioapi, tibble, tools, whisker, withr, xml2 (>= 1.1.1), yaml

Suggests covr, devtools, htmlwidgets, jsonlite, knitr, leaflet, testthat, rtticles

VignetteBuilder knitr

SystemRequirements pandoc

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author Hadley Wickham [aut, cre] (<<https://orcid.org/0000-0003-4757-117X>>), Jay Hesselberth [aut] (<<https://orcid.org/0000-0002-6299-179X>>), RStudio [cph, fnd]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2018-12-07 00:50:07 UTC

R topics documented:

as_pkgdown	2
autolink_html	3
build_articles	4
build_favicon	6
build_home	7
build_news	9
build_reference	10
build_site	12
build_tutorials	17
clean_site	18
deploy_site_github	18
init_site	19
in_pkgdown	20
preview_site	20
rd2html	21
render_page	21
template_navbar	22
Index	24

as_pkgdown	<i>Generate pkgdown data structure</i>
------------	--

Description

You will generally not need to use this unless you need a custom site design and you're writing your own equivalent of `build_site()`.

Usage

```
as_pkgdown(pkg = ".", override = list())
```

Arguments

pkg	Path to package.
override	An optional named list used to temporarily override values in <code>_pkgdown.yml</code>

Description

The autolinker is built around two XPath expressions:

- `//pre[contains(@class, 'r')]`: this finds all `<div>`s with class `sourceCode` and `r`. The contents must be syntax-highlighted using `pygments`. (This is default in `rmarkdown::html_document()` when `theme = NULL`.)
- `./code[count(*) = 0]`: this finds all `<code>` that contain only text (and no other tags).

Usage

```
autolink_html(input, output = input, local_packages = character())
```

Arguments

`input`, `output` Input and output paths for HTML file

`local_packages` A named character vector providing relative paths (value) to packages (name) that can be reached with relative links from the target HTML document.

Details

Currently the following expressions are linked:

- Function calls, `foo()`
- Function calls qualified with the package name, `bar::foo()`
- Symbols qualified with the package name, `bar::baz`
- Help calls, `?foo`, `package?foo`, and `?bar::foo`
- Vignette calls, `vignette(baz)`, `vignette(baz, package = "bar")`

Calls to `library()` and `require()` are used to find the topics connected to unqualified references.

Examples

```
## Not run:
autolink_html("path/to/file.html",
  local_packages = c(
    shiny = "shiny",
    shinydashboard = "shinydashboard"
  )
)

## End(Not run)
```

build_articles *Build articles section*

Description

Each R Markdown vignette in `vignettes/` and its subdirectories is rendered and saved to `articles/`. Vignettes are rendered using a special document format that reconciles `rmarkdown::html_document()` with your pkgdown template.

Usage

```
build_articles(pkg = ".", quiet = TRUE, lazy = TRUE,
              override = list(), preview = NA)

build_article(name, pkg = ".", data = list(), lazy = FALSE,
             quiet = TRUE)
```

Arguments

<code>pkg</code>	Path to package.
<code>quiet</code>	Set to FALSE to display output of knitr and pandoc. This is useful when debugging.
<code>lazy</code>	If TRUE, will only re-build article if input file has been modified more recently than the output file.
<code>override</code>	An optional named list used to temporarily override values in <code>_pkgdown.yml</code>
<code>preview</code>	If TRUE, or <code>is.na(preview) && interactive()</code> , will preview freshly generated section in browser.
<code>name</code>	Name of article to render. This should be either a path relative to <code>vignettes/</code> without extension, or <code>index</code> or <code>README</code> .
<code>data</code>	Additional data to pass on to template.

Details

A vignette with the same name as the package (e.g., `vignettes/pkgdown.Rmd`) gets special treatment. It is rendered and linked to in the navbar under "Get started".

External files

pkgdown differs from base R in its handling of external files. When building vignettes, R assumes that vignettes are self-contained (a reasonable assumption when most vignettes were PDFs) and only copies files explicitly listed in `.install_extras`. pkgdown takes a different approach based on `rmarkdown::find_external_resources()`, and it will also copy any images that you link to. If for some reason the automatic detection doesn't work, you will need to add a `resource_files` field to the yaml metadata, e.g.:

```

---
title: My Document
resource_files:
  - data/mydata.csv
  - images/figure.png
---

```

Note that you can not use the `fig.path` to change the output directory of generated figures as its default value is a strong assumption of `rmarkdown`.

YAML config

To tweak the index page, you need a section called `articles`, which provides a list of sections containing, a title, list of contents, and optional description.

For example, this imaginary file describes some of the structure of the [R markdown articles](#):

```

articles:
- title: R Markdown
  contents:
  - starts_with("authoring")
- title: Websites
  contents:
  - rmarkdown_websites
  - rmarkdown_site_generators

```

Note that `contents` can contain either a list of vignette names (including subdirectories), or if the functions in a section share a common prefix or suffix, you can use `starts_with("prefix")` and `ends_with("suffix")` to select them all. If you don't care about position within the string, use `contains("word")`. For more complex naming schemes you can use an arbitrary regular expression with `matches("regex")`.

`pkgdown` will check that all vignettes are included in the index this page, and will generate a warning if you have missed any.

YAML header

By default, `pkgdown` builds all articles with `rmarkdown::html_document()` using setting the `template` parameter. This overrides any custom settings you have in your YAML metadata, ensuring that all articles are rendered in the same way (and receive the default site template).

If you need to override the output format, or set any options, you'll need to add a `pkgdown` field to your yaml metadata:

```

pkgdown:
  as_is: true

```

This will tell `pkgdown` to use the `output_format` (and options) that you have specified. This format must accept `template`, `theme`, and `self_contained` in order to work with `pkgdown`.

If the output format produces a PDF, you'll also need to specify the `extension` field:

```
pkgdown:  
  as_is: true  
  extension: pdf
```

Suppressing vignettes

If you want articles that are not vignettes, either put them in subdirectories or list in `.Rbuildignore`. An articles link will be automatically added to the default navbar if the vignettes directory is present; if you do not want this, you will need to customise the navbar. See `build_site()` details.

Vignette files prefixed with an underscore (e.g., `_index.Rmd`) are ignored to enable rendering of **bookdown** sites.

Tables of contents

You can control the TOC depth via the YAML configuration file:

```
toc:  
  depth: 2
```

Figures

You can control the default rendering of figures by specifying the `figures` field in `_pkgdown.yml`. The default settings are equivalent to:

```
figures:  
  dev: grDevices::png  
  dpi: 96  
  dev.args: []  
  fig.ext: png  
  fig.width: 7.2916667  
  fig.height: ~  
  fig.retina: 2  
  fig.asp: 1.618
```

build_favicon

Create favicons from package logo

Description

This function auto-detects the location of your package logo (with the name `logo.svg` (recommended format) or `logo.png`) and runs it through the <https://realfavicongenerator.net> API to build a complete set of favicons with different sizes, as needed for modern web usage.

Usage

```
build_favicon(pkg = ".")
```

Arguments

pkg Path to package.

Details

You only need to run the function once. The favicon set will be stored in pkgdown/favicon and copied by `init_site()` to the relevant location when the website is rebuilt.

build_home	<i>Build home section</i>
------------	---------------------------

Description

This function is responsible for building `.md` (or `.Rmd`) files typically found in the root directory of the package. It will generate the home page from `index.Rmd` or `README.Rmd`, then `index.md` or `README.md`. If none of these are found, it falls back to the description field in `DESCRIPTION`. It also builds any files found in `.github/`.

Usage

```
build_home(pkg = ".", override = list(), preview = NA,
           quiet = TRUE)
```

Arguments

pkg Path to package.

override An optional named list used to temporarily override values in `_pkgdown.yml`

preview If `TRUE`, or `is.na(preview) && interactive()`, will preview freshly generated section in browser.

quiet Set to `FALSE` to display output of knitr and pandoc. This is useful when debugging.

Images and figures

If you want to include images in your `README.md`, they must be stored in somewhere in the package so that they can be displayed on the CRAN website. The best place to put them is `man/figures`. If you are generating figures with R Markdown, make sure you set up `fig.path` as followed:

```
```{r, include = FALSE}
knitr::opts_chunk$set(
 fig.path = "man/figures/"
)
```
```

Package logo

If you have a package logo, you can include it at the top of your README in a level-one heading:

```
# pkgdown 
```

`init_site()` will also automatically create a favicon set from your package logo.

YAML config - home

To tweak the home page, you need a section called `home`.

The sidebar links are automatically generated by inspecting the `URL` and `BugReports` fields of the `DESCRIPTION`. You can add additional links with a subsection called `links`, which should contain a list of `text + href` elements:

```
home:
  links:
    - text: Link text
      href: http://website.com
```

You can remove the first heading with:

```
home:
  strip_header: true
```

YAML config - authors

The "developers" list is populated by the maintainer ("`cre`"), authors ("`aut`"), and funder ("`fnd`") from the `DESCRIPTION`. You can modify their display on the home page by adding a subsection for authors. The author name should match the value in `DESCRIPTION`:

```
authors:
  name:
    href: "http://name-website.com"
    html: "<img src='name-picture.png' height=24>"
```

ORCID

ORCID identification numbers are added to author info if they are defined in the `DESCRIPTION`:

```
person("Jay", "Hesselberth", role = "aut",
  comment = c(ORCID = "0000-0002-6299-179X")
),
```

Badges

Status badges are displayed in the sidebar under the section "Dev status". This section is automatically populated if the first paragraph of the homepage consists solely of status badges as linked images.

| | |
|------------|---------------------------|
| build_news | <i>Build news section</i> |
|------------|---------------------------|

Description

Your NEWS.md is parsed in to sections based on your use of headings.

Usage

```
build_news(pkg = ".", override = list(), preview = NA)
```

Arguments

| | |
|----------|---|
| pkg | Path to package. |
| override | An optional named list used to temporarily override values in <code>_pkgdown.yml</code> |
| preview | If TRUE, or <code>is.na(preview) && interactive()</code> , will preview freshly generated section in browser. |

Details

The NEWS.md file should be formatted with level one headings (#) containing the package name and version number, level two headings (##) with topic headings and lists of news bullets. Commonly used level two headings include 'Major changes', 'Bug fixes', or 'Minor changes'.

```
# pkgdown 0.1.0.9000
```

```
## Major changes
```

- Fresh approach based on the staticdocs package. Site configuration now based on YAML files.

If the package is available on CRAN, release dates will be added to versions in level-one headings, and "Unreleased" will be added versions that are not on CRAN.

Issues and contributors mentioned in news items are automatically linked to github if a URL entry linking to github.com is provided in the package DESCRIPTION.

```
## Major changes
```

- Lots of bug fixes (@hadley, #100)

YAML config

To automatically link to release announcements, include a releases section.

```
news:
  releases:
    - text: "usethis 1.3.0"
      href: https://www.tidyverse.org/articles/2018/02/usethis-1-3-0/
    - text: "usethis 1.0.0 (and 1.1.0)"
      href: https://www.tidyverse.org/articles/2017/11/usethis-1.0.0/
```

Control whether news is present on one page or multiple pages with the `one_page` field. The default is true.

```
news:
- one_page: false
```

See Also

[Tidyverse style for News](#)

| | |
|-----------------|--------------------------------|
| build_reference | <i>Build reference section</i> |
|-----------------|--------------------------------|

Description

By default, pkgdown will generate an index that lists all functions in alphabetical order. To override this, provide a reference section in your `_pkgdown.yml` as described below.

Usage

```
build_reference(pkg = ".", lazy = TRUE, document = FALSE,
  examples = TRUE, run_dont_run = FALSE, seed = 1014,
  override = list(), preview = NA)
```

```
build_reference_index(pkg = ".")
```

Arguments

| | |
|---------------------------|---|
| <code>pkg</code> | Path to package. |
| <code>lazy</code> | If TRUE, only rebuild pages where the <code>.Rd</code> is more recent than the <code>.html</code> . This makes it much easier to rapidly prototype. It is set to FALSE by <code>build_site()</code> . |
| <code>document</code> | If TRUE, will run <code>devtools::document()</code> before updating the site. |
| <code>examples</code> | Run examples? |
| <code>run_dont_run</code> | Run examples that are surrounded in <code>\dontrun?</code> |
| <code>seed</code> | Seed used to initialize so that random examples are reproducible. |
| <code>override</code> | An optional named list used to temporarily override values in <code>_pkgdown.yml</code> |
| <code>preview</code> | If TRUE, or <code>is.na(preview) && interactive()</code> , will preview freshly generated section in browser. |

YAML config

To tweak the index page, add a section called `reference` to `_pkgdown.yml` which provides a list of sections containing, a title, list of contents, and an optional description.

For example, the following code breaks up the functions in `pkgdown` into two groups:

```
reference:
- title: Render components
  desc: Build each component of the site.
  contents:
  - starts_with("build_")
  - init_site
- title: Templates
  contents:
  - render_page
```

Note that `contents` can contain either a list of function names, or if the functions in a section share a common prefix or suffix, you can use `starts_with("prefix")` and `ends_with("suffix")` to select them all. For more complex naming schemes you can use an arbitrary regular expression with `matches("regexp")`. You can also use a leading `-` to exclude matches from a section. By default, these functions that match multiple topics will exclude topics with the Rd keyword "internal". To include these, use `starts_with("build_", internal = TRUE)`.

You can also select topics that contain specified Rd concepts with `has_concept("blah")`.

You can provide long descriptions for groups of functions using the `YAML >` notation:

```
desc: >
  This is a very long and overly flowery description of a
  single simple function.
```

If you have functions with odd names (e.g. that start with a plus symbol `+`), you can include them by double-escaping. This YAML entry adds the `+.gg` function to the `ggplot2` documentation:

```
- "`+.gg`"
```

`pkgdown` will check that all non-internal topics are included on this page, and will generate a warning if you have missed any.

Figures

You can control the default rendering of figures by specifying the `figures` field in `_pkgdown.yml`. The default settings are equivalent to:

```
figures:
  dev: grDevices::png
  dpi: 96
  dev.args: []
  fig.ext: png
  fig.width: 7.2916667
  fig.height: ~
  fig.retina: 2
  fig.asp: 1.618
```

Icons

You can optionally supply an icon for each help topic. To do so, you'll need a top-level icons directory. This should contain .png files that are either 30x30 (for regular display) or 60x60 (if you want retina display). Icons are matched to topics by aliases.

build_site

Build a complete pkgdown website

Description

build_site() is a convenient wrapper around six functions:

- [init_site\(\)](#)
- [build_home\(\)](#)
- [build_reference\(\)](#)
- [build_articles\(\)](#)
- [build_tutorials\(\)](#)
- [build_news\(\)](#)

See the documentation for the each function to learn how to control that aspect of the site.

Note if names of generated files were changed, you will need to use [clean_site](#) first to clean up orphan files.

Usage

```
build_site(pkg = ".", examples = TRUE, document = TRUE,
  run_dont_run = FALSE, seed = 1014, lazy = FALSE,
  override = list(), preview = NA, new_process = TRUE)
```

Arguments

| | |
|--------------|---|
| pkg | Path to package. |
| examples | Run examples? |
| document | If TRUE, will run <code>devtools::document()</code> before updating the site. |
| run_dont_run | Run examples that are surrounded in <code>\dontrun</code> ? |
| seed | Seed used to initialize so that random examples are reproducible. |
| lazy | If TRUE, will only rebuild articles and reference pages if the source is newer than the destination. |
| override | An optional named list used to temporarily override values in <code>_pkgdown.yml</code> |
| preview | If TRUE, or <code>is.na(preview) && interactive()</code> , will preview freshly generated section in browser. |
| new_process | If TRUE, will run <code>build_site()</code> in a separate process. This enhances reproducibility by ensuring nothing that you have loaded in the current process affects the build process. |

YAML config

There are four top-level YAML settings that affect the entire site: `destination`, `url`, `title`, `template`, and `navbar`.

`destination` controls where the site will be generated. It defaults to `docs/` (for GitHub pages), but you can override if desired. Relative paths will be taken relative to the package root.

`url` optionally specifies the url where the site will be published. Supplying this will:

- Allow other pkgdown sites will link to your site when needed, rather than using generic links to <https://rdocumentation.org>.
- Generate a `sitemap.xml`, increasing the searchability of your site.
- Automatically generate a CNAME when [deploying to github](#).

`url`: `http://pkgdown.r-lib.org`

`title` overrides the default site title, which is the package name. It's used in the page title and default navbar.

You can also provided information to override the default display of the authors. Provided a list named with the name of each author, including `href` to add a link, or `html` to override the text:

`authors`:

Hadley Wickham:

`href`: `http://hadley.nz`

RStudio:

`href`: `https://www.rstudio.com`

`html`: ``

Development mode

The development mode of a site controls four main things:

- Where the site is built.
- The colour of the package version in the navbar.
- The optional tooltip associated with the version.
- The indexing of the site by search engines.

There are currently three possible development modes:

- **release**: site written to `docs/`, the version gets the default colouring, and no message.
- **development**: written to `docs/dev/`, the version gets a danger label, and message stating these are docs for an in-development version of the package. The `noindex` meta tag is used to ensure that these packages are not indexed by search engines.
- **unreleased**: the package is written to `docs/`, the version gets a "danger" label, and the message indicates the package is not yet on CRAN.

The default development mode is "release". You can override it by adding a new development field to `_pkgdown.yml`, e.g.

```
development:
  mode: development
```

You can also have pkgdown automatically detect the mode with:

```
development:
  mode: auto
```

The mode will be automatically determined based on the version number:

- 0.0.0.9000: unreleased
- four version components: development
- everything else -> release

There are three other options that you can control:

```
development:
  destination: dev
  version_label: danger
  version_tooltip: "Custom message here"
```

`destination` allows you to override the default subdirectory used for the development site; it defaults to `dev/`. `version_label` allows you to override the style used for development (and unreleased) versions of the package. It defaults to "danger", but you can set to "default", "info", or "warning" instead. (The precise colours are determined by your bootstrap theme, but become progressively more eye catching as you go from default to danger). Finally, you can choose to override the default tooltip with `version_tooltip`.

YAML config - navbar

`navbar` controls the navbar at the top of the page. It has two primary components: `structure` and `components`. These components interact in a somewhat complicated way, but the complexity allows you to make minor tweaks to part of the navbar while relying on pkgdown to automatically generate the rest.

The `structure` defines the layout of the navbar, i.e. the order of the components, and whether they're right aligned or left aligned. You can use this component to change the order of the default components, and to add your own components.

```
navbar:
  structure:
    left: [home, intro, reference, articles, tutorials, news]
    right: [github]
```

The `components` describes the appearance of each element in the navbar. It uses the same syntax as [RMarkdown](#). The following YAML snippet illustrates some of the most important features.

```
components:
  home: ~
  articles:
```

```

text: Articles
menu:
- text: Category A
- text: Title A1
  href: articles/a1.html
- text: Title A2
  href: articles/a2.html
- text: -----
- text: "Category B"
- text: Title B1
  href: articles/b1.html
twitter:
  icon: fa-lg fa-twitter
  href: http://twitter.com/hadleywickham

```

Components can contain sub-menus with headings (indicated by missing href) and separators (indicated by a bunch of -). You can use icons from fontawesome: see a full list <https://fontawesome.com/icons>.

This yaml would override the default "articles" component, eliminate the "home" component, and add a new "twitter" component. Unless you explicitly mention new components in the structure they'll be added to the far right of the left menu.

YAML config - search

You can use [docsearch](#) by algolia to add search to your site.

```

template:
  params:
    docsearch:
      api_key: API_KEY
      index_name: INDEX_NAME

```

You also need to add a url: field, see above.

YAML config - template

You can get complete control over the appearance of the site using the template component. There are two components to the template: the HTML templates used to layout each page, and the css/js assets used to render the page in the browser.

The easiest way to tweak the default style is to use a bootswatch template, by passing on the bootswatch template parameter to the built-in template:

```

template:
  params:
    bootswatch: cerulean

```

See a complete list of themes and preview how they look at <https://gallery.shinyapps.io/117-shinythemes/>:

Optionally provide the ganalytics template parameter to enable [Google Analytics](#). It should correspond to your [tracking id](#).

```
template:
  params:
    ganalytics: UA-000000-01
```

Suppress indexing of your pages by web robots by setting `noindex: true`:

```
template:
  params:
    noindex: true
```

You can also override the default templates and provide additional assets. You can do so by either storing in a package with directories `inst/pkgdown/assets` and `inst/pkgdown/templates`, or by supplying `path` and `asset_path`. To suppress inclusion of the default assets, set `default_assets` to `false`.

```
template:
  package: mycustompackage
```

OR:

```
template:
  path: path/to/templates
  assets: path/to/assets
  default_assets: false
```

These settings are currently recommended for advanced users only. There is little documentation, and you'll need to read the existing source for pkgdown templates to ensure that you use the correct components.

Internet

Users with limited internet connectivity can disable CRAN checks by setting `options(pkgdown.internet = FALSE)`. This will also disable some features from pkgdown that requires an internet connectivity. However, if it is used to build docs for a package that requires internet connectivity in examples or vignettes, this connection is required as this option won't apply on them.

Examples

```
## Not run:
build_site()

build_site(override = list(destination = tempdir()))

## End(Not run)
```

| | |
|-----------------|--------------------------------|
| build_tutorials | <i>Build tutorials section</i> |
|-----------------|--------------------------------|

Description

learnr tutorials must be hosted elsewhere as they require an R execution engine. Currently, pkgdown will not build or publish tutorials for you, but makes it easy to embed (using `<iframe>`s) published tutorials. Tutorials are automatically discovered from published tutorials in `inst/tutorials` and `vignettes/tutorials`. Alternatively, you can list in `_pkgdown.yml` as described below.

Usage

```
build_tutorials(pkg = ".", override = list(), preview = NA)
```

Arguments

| | |
|----------|---|
| pkg | Path to package. |
| override | An optional named list used to temporarily override values in <code>_pkgdown.yml</code> |
| preview | If TRUE, or <code>is.na(preview) && interactive()</code> , will preview freshly generated section in browser. |

YAML config

To override the default discovery process, you can provide a `tutorials` section. This should be a list where each element specifies:

- `name`: used for the generated file name
- `title`: used in page heading and in navbar
- `url`: which will be embedded in an `iframe`
- `source`: optional, but if present will be linked to

```
tutorials:
- name: 00-setup
  title: Setting up R
  url: https://jjallaire.shinyapps.io/learnr-tutorial-00-setup/
- name: 01-data-basics
  title: Data basics
  url: https://jjallaire.shinyapps.io/learnr-tutorial-01-data-basics/
```

| | |
|------------|-------------------|
| clean_site | <i>Clean site</i> |
|------------|-------------------|

Description

Delete all files in docs/ (except for CNAME).

Usage

```
clean_site(pkg = ".")
```

Arguments

| | |
|-----|------------------|
| pkg | Path to package. |
|-----|------------------|

| | |
|--------------------|---|
| deploy_site_github | <i>Deploy a pkgdown site on Travis-CI to Github Pages</i> |
|--------------------|---|

Description

deploy_site_github() sets up your SSH keys for deployment, builds the site with [build_site\(\)](#), commits the site to the gh-pages branch and then pushes the results back to GitHub. deploy_site_github() is meant only to be used by the CI system on Travis, it should not be called locally. See 'Setup' for details on setting up your repository to use this.

Usage

```
deploy_site_github(pkg = ".", tarball = Sys.getenv("PKG_TARBALL", ""),
  ssh_id = Sys.getenv("id_rsa", ""),
  repo_slug = Sys.getenv("TRAVIS_REPO_SLUG", ""),
  commit_message = construct_commit_message(pkg), verbose = FALSE, ...)
```

Arguments

| | |
|----------------|--|
| pkg | Path to package. |
| tarball | The location of the built package tarball. The default Travis configuration for R packages sets PKG_TARBALL to this path. |
| ssh_id | The private id to use, a base64 encoded content of the private pem file. This should <i>not</i> be your personal private key. Instead create a new keypair specifically for deploying the site. The easiest way is to use <code>travis::use_travis_deploy()</code> . |
| repo_slug | The user/repo slug for the repository. |
| commit_message | The commit message to be used for the commit. |
| verbose | Print verbose output |
| ... | Additional arguments passed to build_site() . |

Setup

Add the following to your `.travis.yml` file.

```
before_deploy: Rscript -e 'remotes::install_cran("pkgdown")'
deploy:
  provider: script
  script: Rscript -e 'pkgdown::deploy_site_github()'
  skip_cleanup: true
```

Then you will need to setup your deployment keys. The easiest way is to call `travis::use_travis_deploy()`. This will generate and push the necessary keys to your GitHub and Travis accounts. See the [travis package website](#) for more details.

Next, make sure that a `gh-pages` branch exists. The simplest way to do so is to run the following git commands locally:

```
git checkout --orphan gh-pages
git rm -rf .
git commit --allow-empty -m 'Initial gh-pages commit'
git push origin gh-pages
git checkout master
```

We recommend doing this outside of RStudio (with the project closed) as from RStudio's perspective you end up deleting all the files and then re-creating them.

If you're using a custom CNAME, make sure you have set the `url` in `_pkgdown.yml`:

```
url: http://pkgdown.r-lib.org
```

| | |
|-----------|---------------------------------------|
| init_site | <i>Initialise site infrastructure</i> |
|-----------|---------------------------------------|

Description

This creates the output directory (`docs/`), a machine readable description of the site, and copies CSS/JS assets and extra files.

Usage

```
init_site(pkg = ".")
```

Arguments

`pkg` Path to package.

Build-ignored files

We recommend using `usethis::use_pkgdown()` to build-ignore docs/ and _pkgdown.yml. If use another directory, or create the site manually, you'll need to add them to `.Rbuildignore` yourself. A NOTE about an unexpected file during R CMD CHECK is an indication you have not correctly ignored these files.

Custom CSS/JS

If you want to do minor customisation of your pkgdown site, the easiest way is to add `pkgdown/extra.css` and `pkgdown/extra.js`. These will be automatically copied to docs/ and inserted into the <HEAD> after the default pkgdown CSS and JS.

Favicon

You should manually run `build_favicon()` once to generate the favicon set from your logo. The result is stored in `pkgdown/favicon` and will automatically be copied to the relevant location when you run `init_site()`.

| | |
|-------------------------|---|
| <code>in_pkgdown</code> | <i>Determine if code is executed by pkgdown</i> |
|-------------------------|---|

Description

This is occasionally useful when you need different behaviour by pkgdown and regular documentation.

Usage

```
in_pkgdown()
```

Examples

```
in_pkgdown()
```

| | |
|---------------------------|-----------------------------|
| <code>preview_site</code> | <i>Open site in browser</i> |
|---------------------------|-----------------------------|

Description

Open site in browser

Usage

```
preview_site(pkg = ".", path = ".", preview = NA)
```

Arguments

| | |
|---------|---|
| pkg | Path to package. |
| path | Path relative to destination |
| preview | If TRUE, or is.na(preview) && interactive(), will preview freshly generated section in browser. |

| | |
|---------|--|
| rd2html | <i>Translate an Rd string to its HTML output</i> |
|---------|--|

Description

Translate an Rd string to its HTML output

Usage

```
rd2html(x, fragment = TRUE, ...)
```

Arguments

| | |
|----------|---|
| x | Rd string. Backslashes must be double-escaped (""). |
| fragment | logical indicating whether this represents a complete Rd file |
| ... | additional arguments for as_html |

Examples

```
rd2html("a\n%b\nc")
rd2html("a & b")
rd2html("\\strong{\\emph{x}}")
```

| | |
|-------------|----------------------------------|
| render_page | <i>Render page with template</i> |
|-------------|----------------------------------|

Description

Each page is composed of four templates: "head", "header", "content", and "footer". Each of these templates is rendered using the data, and then assembled into an overall page using the "layout" template.

Usage

```
render_page(pkg = ".", name, data, path = "", depth = NULL,
            quiet = FALSE)
```

```
data_template(pkg = ".", depth = 0L)
```

Arguments

| | |
|-------|---|
| pkg | Path to package to document. |
| name | Name of the template (e.g. "home", "vignette", "news") |
| data | Data for the template.
This is automatically supplemented with three lists: <ul style="list-style-type: none"> • site: title and path to root. • yaml: the template key from <code>_pkgdown.yml</code>. • package: package metadata including name and version. See the full contents by running <code>data_template()</code> . |
| path | Location to create file; relative to destination directory. If "" (the default), prints to standard out. |
| depth | Depth of path relative to base directory. |
| quiet | If quiet, will suppress output messages |

| | |
|-----------------|--------------------------------|
| template_navbar | <i>Generate YAML templates</i> |
|-----------------|--------------------------------|

Description

Use these function to generate the default YAML that pkgdown uses for the different parts of `_pkgdown.yml`. This are useful starting points if you want to customise your site.

Usage

```
template_navbar(path = ".")
```

```
template_reference(path = ".")
```

```
template_articles(path = ".")
```

Arguments

| | |
|------|----------------------|
| path | Path to package root |
|------|----------------------|

Examples

```
## Not run:  
pkgdown::template_navbar()  
  
## End(Not run)  
  
## Not run:  
pkgdown::template_reference()  
  
## End(Not run)  
  
## Not run:  
pkgdown::template_articles()  
  
## End(Not run)
```

Index

as_pkgdown, 2
autolink_html, 3

build_article (build_articles), 4
build_articles, 4
build_articles(), 12
build_favicon, 6
build_favicon(), 20
build_home, 7
build_home(), 12
build_news, 9
build_news(), 12
build_reference, 10
build_reference(), 12
build_reference_index
 (build_reference), 10
build_site, 12
build_site(), 2, 6, 10, 18
build_tutorials, 17
build_tutorials(), 12

clean_site, 12, 18

data_template (render_page), 21
data_template(), 22
deploy_site_github, 18
deploying to github, 13
devtools::document(), 10, 12

in_pkgdown, 20
init_site, 19
init_site(), 7, 8, 12, 20

preview_site, 20

rd2html, 21
render_page, 21
rmarkdown::find_external_resources(),
 4
rmarkdown::html_document(), 3–5

template_articles (template_navbar), 22
template_navbar, 22
template_reference (template_navbar), 22

usethis::use_pkgdown(), 20