Package 'pkgdown'

September 10, 2020

Title Make Static HTML Documentation for a Package

Version 1.6.1

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),
crayon,
desc,
digest,
downlit,
fs (>= 1.3.0),
httr (>= 1.4.2),
magrittr,
memoise,
openssl,
purrr,
ragg,
rematch2,
rlang (>= 0.3.0),
rmarkdown (>= 1.1.9007),
tibble,
tools,
whisker,
withr,
xml2 (>= 1.3.1),
yaml

Suggests covr,
htmlwidgets,
jsonlite,
knitr,
leaflet,
pkgload (>= 1.0.2),
testthat (>= 2.1.0),

rticles,
rsconnect,
rstudioapi

VignetteBuilder knitr

SystemRequirements pandoc

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

R topics documented:

`as_pkgdown` *Generate pkgdown data structure*

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 _pkgdown.yml

`build_articles` *Build articles section*

Description

`build_articles()` renders each R Markdown file underneath vignettes/ and saves it to articles/. There are two exceptions:

- Files that start with _ (e.g., _index.Rmd) are ignored, enabling the use of child documents in bookdown
- Files in `vignettes/tutorials` are handled by `build_tutorials()`

Vignettes are rendered using a special document format that reconciles `rmarkdown::html_document()` with the pkgdown template. This means articles behave slightly differently to vignettes, particularly with respect to external files, and custom output formats. See below for more details.

Note that when you run `build_articles()` directly (outside of `build_site()`) vignettes will use the currently installed version of the package, not the current source version. This makes iteration quicker when you are primarily working on the text of an article.

Usage

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

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

build_articles_index(pkg = ".")
```

Arguments

`pkg`	Path to package.
`quiet`	Set to `FALSE` to display output of knitr and pandoc. This is useful when debugging.
`lazy`	If `TRUE`, will only re-build article if input file has been modified more recently than the output file.
`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.
`name`	Name of article to render. This should be either a path relative to vignettes/ without extension, or `index` or `README`.
`data`	Additional data to pass on to template.

Index and navbar

You can control the articles index and navbar with a `articles` section in your _pkgdown.yaml. It defines a list of sections, each of which can contain four fields:

- `title` (required): title of section, which appears as a heading on the articles index.
- `desc` (optional): An optional markdown description displayed underneath the section title.
- `navbar` (optional): A couple of words used to label this section in the navbar. If omitted, this section of vignettes will not appear in the navbar.
- `contents` (required): a list of article names to include in the section. This can either be names of individual vignettes or a call to `starts_with()`. The name of a vignette includes its path under `vignettes` without extension so that the name of the vignette found at `vignettes/pizza/slice.Rmd` is `pizza/slice`.

The title and description of individual vignettes displayed on the index comes from `title` and `description` fields of the YAML header in the Rmds.

For example, this yaml might be used for some version of dplyr:

```
articles:
- title: Main verbs
  navbar: ~
  contents:
  - one-table
  - two-table
  - rowwise
  - colwise

- title: Developer
  desc: Vignettes aimed at package developers
  contents:
  - programming
  - packages
```

Note the use of the `navbar` fields. navbar: ~ means that the "Main verbs" will appear in the navbar without a heading; the absence of the `navbar` field in the for the developer vignettes means that they will only be accessible via the articles index.

The navbar will include a link to the articles index if one or more vignettes are not available through the navbar. If some vignettes appear in the navbar drop-down list and others do not, the list will automatically include a "More ..." link at the bottom; if no vignettes appear in the the navbar, it will link directly to the articles index instead of providing a drop-down.

Get started

Note that a vignette with the same name as the package (e.g., `vignettes/pkgdown.Rmd` or `vignettes/articles/pkgdo` automatically becomes a top-level "Get started" link, and will not appear in the articles drop-down.

(If your package name includes a `.`, e.g. `pack.down`, use a `-` in the vignette name, e.g. `pack-down.Rmd`.)

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.

Embedding Shiny apps

If you would like to embed a Shiny app into an article, the app will have to be hosted independently, (e.g. https://www.shinyapps.io). Then, you can embed the app into your article using an <iframe>, e.g. <iframe src = "https://gallery.shinyapps.io/083-front-page" class="shiny-app">.

See https://github.com/r-lib/pkgdown/issues/838#issuecomment-430473856 for some hints on how to customise the appearance with CSS.

YAML header

By default, pkgdown builds all articles with rmarkdown::html_document() by 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
```

If you want to set an output format for all your articles, you can do that by adding a vignettes/_site.yml, much like you would for an rmarkdown website. For example, you can backport some bookdown features such as cross-references to all your articles by using the bookdown::html_document2 format.

```
output:
  bookdown::html_document2:
  number_sections: false
```

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.

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: ragg::agg_png
  dpi: 96
  dev.args: []
  fig.ext: png
  fig.width: 7.2916667
  fig.height: ~
  fig.retina: 2
  fig.asp: 1.618
  bg: NA
```

---

build_favicons *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_favicons(pkg = ".", overwrite = FALSE)

build_favicon(pkg, overwrite)
```

Arguments

`pkg`	Path to package.
`overwrite`	If `TRUE`, re-create favicons from package logo.

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.

Once complete, you should add pkgdown/ to `.Rbuildignore` to avoid a NOTE during package checking.

build_home *Build home section*

Description

This function generates the home page, converts .md files found in the package root (and in .github/), and builds an authors page from DESCRIPTION and inst/CITATION (if present).

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.

Home page

The home page (index.html) is generated from _pkgdown/index.md, index.md, or README.md, in that order. Most packages will use README.md because that's also displayed by GitHub and CRAN. Use index.md if you want your package website to look different to your README, and use _pkgdown/index.md if you don't want that file to live in your package root directory.

If you use index.Rmd or README.Rmd it's your responsibility to knit the document to create the corresponding .md. pkgdown does not do this for you because it only touches files in the doc/ directory.

Sidebar

The sidebar is automatically populated with:

- Development status badges found in README.md/index.md. pkgdown identifies badges in three ways:
  - Any image-containing links between <!-- badges: start --> and <!-- badges: end -->, as e.g. created by usethis::use_readme_md() or usethis::use_readme_rmd().
  - Any image-containing links within .
  - Within the first paragraph, if it only contains image-containing links.
- A link for bug reports is added if the BugReports field in DESCRIPTION contains a link. You can use usethis::use_github_links() to populate this field.
- Licensing information if LICENSE/LICENCE or LICENSE.md/LICENCE.md files are present.
- Community information is linked in the side bar using the .github/CONTRIBUTING.md and .github/CODE_OF_CONDUCT.md files, if present.
- Extra markdown files in the base directory or in .github/ are copied to docs/ and converted to HTML.

- Citation information from a `inst/CITATION` file is linked in the side bar to the authors page.
- Author ORCID identification numbers in the `DESCRIPTION` are linked under "Developers" using the ORCID logo:

```
Authors@R: c(
    person("Hadley", "Wickham", , "hadley@rstudio.com", role = c("aut", "cre"),
      comment = c(ORCID = "0000-0003-4757-117X")
    ),
    person("Jay", "Hesselberth", role = "aut",
      comment = c(ORCID = "0000-0002-6299-179X")
    )
  )
```

Images and figures

If you want to include images in your `README.md`, they must be stored 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:

```
knitr::opts_chunk$set(
  fig.path = "man/figures/"
)
```

This should usually go in a block with `include = FALSE`.

Package logo

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

```
# pkgdown <img src="man/figures/logo.png" align="right" />
```

`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`.

By default, the page title and description are extracted automatically from the `Title` and `Description` fields `DESCRIPTION` (stripping single quotes off quoted words). CRAN ensures that these fields don't contain phrases like "R package" because that's obvious on CRAN. To make your package more findable with google, it's good practice to override the `title` and `description`, thinking about what people might search for:

```
home:
  title: An R package for pool-noodle discovery
  description: >
    Do you love R? Do you love pool-noodles? If so, you might enjoy
    using this package to automatically discover and add pool-noodles
    to your growing collection.
```

(Note the use of YAML's `>`; this is a convenient way of writing paragraphs of text.)

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
```

READMEs usually start with an <h1> containing the package name. If that feels duplicative with the package name in the navbar you can remove it with `strip_header: true`:

```
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`. Each entry in `authors` should be named with the author's name (matching `DESCRIPTION`) and can contain `href` and/or `html` fields:

- If `href` is provided, the author's name will be linked to this url.
- If `html` is provided, it will be shown instead of the author's name. This is particularly useful if you want to display the logo of a corporate sponsor.

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

`build_news` *Build news section*

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 _pkgdown.yml
`preview`	If `TRUE`, or `is.na(preview) && interactive()`, 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
```

Suppress the default addition of CRAN release dates with:

```
news:
  cran_dates: false
```

See Also

Tidyverse style for News

build_reference *Build reference section*

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,
  examples = TRUE,
  run_dont_run = FALSE,
  seed = 1014,
  override = list(),
  preview = NA,
  devel = TRUE,
  document = "DEPRECATED",
  topics = NULL
)

build_reference_index(pkg = ".")
```

Arguments

`pkg`	Path to package.
`lazy`	If `TRUE`, only rebuild pages where the `.Rd` is more recent than the `.html`. This makes it much easier to rapidly prototype. It is set to `FALSE` by `build_site()`.
`examples`	Run examples?
`run_dont_run`	Run examples that are surrounded in \dontrun?
`seed`	Seed used to initialize so that random examples are reproducible.
`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.
`devel`	Determines how code is loaded in order to run examples. If `TRUE` (the default), assumes you are in a live development environment, and loads source package with `pkgload::load_all()`. If `FALSE`, uses the installed version of the package.
`document`	**Deprecated** Use `devel` instead.
`topics`	Build only specified topics. If supplied, sets lazy`` and previewtoFALSE‘.

Reference index

To tweak the index page, add a section called `reference` to _pkgdown.yml. It can contain three different types of element:

- A **title** (`title` + `desc`), which generates an row containing an <h2> with optional paragraph description.

- A **subtitle** (`subtitle + desc`), which generates an row containing an <h3> with optional paragraph description.
- A **list of topics** (`contents`), which generates one row for each topic, with a list of aliases for the topic on the left, and the topic title on the right.

(For historical reasons you can include `contents` with a title or subtitle, but this is no longer recommended).

Most packages will only need to use `title` and `contents` components. For example, here's a snippet from the YAML that pkgdown uses to generate its own reference index:

```
reference:
- title: Build
  desc:  Build a complete site or its individual section components.
- contents:
  - starts_with("build_")
- title: Templates
- contents:
  - template_navbar
  - render_page
```

Bigger packages, e.g. ggplot2, may need an additional layer of structure in order to clearly organise large number of functions:

```
reference:
- title: Layers
- subtitle: Geoms
  desc: Geom is short for geometric element
- contents:
  - starts_with("geom")
- subtitle: Stats
  desc: Statistical transformations transform data before display.
  contents:
  - starts_with("stat")
```

`desc` can use markdown, and if you have a long description it's a good idea to take advantage of the YAML > notation:

```
desc: >
  This is a very _long_ and **overly** flowery description of a
  single simple function. By using `>`, it's easy to write a description
  that runs over multiple lines.
```

Topic matching:

`contents` can contain:

- Individual function/topic names.
- Weirdly named functions with doubled quoting, once for YAML and once for R, e.g. `"`+.gg`"`.
- `starts_with("prefix")` to select all functions with common prefix.
- `ends_with("suffix")` to select all functions with common suffix.
- `matches("regexp")` for more complex regular expressions.
- `has_keyword("x")` to select all topics with keyword "x"; `has_keyword("datasets")` selects all data documentation.

- `has_concept("blah")` to select all topics with concept "blah". If you are using roxygen2, `has_concept()` also matches family tags, because roxygen2 converts them to concept tags.
- `lacks_concepts(c("concept1","concept2"))` to select all topics without those concepts. This is useful to capture topics not otherwise captured by `has_concepts()`.

All functions (except for `has_keywords()`) automatically exclude internal topics (i.e. those with \keyword{internal}). You can choose to include with (e.g.) `starts_with("build_",internal = TRUE)`.

Use a leading `-` to remove topics from a section, e.g. `-topic_name`, `-starts_with("foo")`.

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

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.

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: ragg::agg_png
  dpi: 96
  dev.args: []
  fig.ext: png
  fig.width: 7.2916667
  fig.height: ~
  fig.retina: 2
  fig.asp: 1.618
  bg: NA
```

`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,
  run_dont_run = FALSE,
  seed = 1014,
  lazy = FALSE,
  override = list(),
  preview = NA,
  devel = FALSE,
  new_process = !devel,
  install = !devel,
  document = "DEPRECATED"
)
```

Arguments

pkg	Path to package.
examples	Run examples?
run_dont_run	Run examples that are surrounded in \dontrun?
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 _pkgdown.yml
preview	If TRUE, or `is.na(preview) && interactive()`, will preview freshly generated section in browser.
devel	Use development or deployment process? If TRUE, uses lighter-weight process suitable for rapid iteration; it will run examples and vignettes in the current process, and will load code with `pkgload::load_all()`. If FALSE, will first install the package to a temporary library, and will run all examples and vignettes in a new process. `build_site()` defaults to `devel = FALSE` so that you get high fidelity outputs when you building the complete site; `build_reference()`, `build_home()` and friends default to `devel = TRUE` so that you can rapidly iterate during development.
new_process	If TRUE, will run `build_site()` in a separate process. This enhances reproducibility by ensuring nothing that you have loaded in the current process affects the build process.
install	If TRUE, will install the package in a temporary library so it is available for vignettes.
document	**Deprecated** Use `devel` instead.

YAML config

There are five 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 to link to your site when needed, rather than using generic links to https://rdrr.io. See `vignette("linking")` for more information.
- Generate a `sitemap.xml`, increasing the searchability of your site.
- Automatically generate a `CNAME` when deploying to github.

```
url: https://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: <img src="https://www.tidyverse.org/rstudio-logo.svg" height="24" />
```

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: devel
```

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 (0.0.0.*): 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

By default, the top navigation bar (the "navbar") will contain links to:

- The home page, with a "home" icon.
- "Get Started", if you have an article with the same name as the package (e.g., `vignettes/pkgdown.Rmd`).
- Reference
- Articles (i.e., vignettes, if present).
- News (if present).
- A "github" icon with a link to your github repo (if listed in the `DESCRIPTION` url field).

You can override these defaults with the `navbar` field. 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.

```
navbar:
  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
      menu:
      - text "Sub-category B11"
        href: articles/b11.html
    twitter:
      icon: "fab fa-twitter fa-lg"
      href: https://twitter.com/hadleywickham
```

Components can contain sub-menus with headings (indicated by missing `href`) and separators (indicated by a bunch of `-`). You can also use `icons` from fontawesome.

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.

When enabling Google Analytics, be aware of the type and amount of user information that you are collecting. You may wish to limit the extent of data collection or to add a privacy disclosure to your site, in keeping with current laws and regulations.

```
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.

YAML config - repo

pkgdown automatically generates links to the source repository in a few places

- Articles and documentation topics are linked back to the underlying source file.
- The NEWS automatically links issue numbers and user names.
- The homepage provides a link to "Browse source code"

pkgdown automatically figures out the necessary URLs if you link to a GitHub or GitLab repo in your `BugReports` or `URL` field. Otherwise, you can supply your own in the `repo` component:

```
repo:
  url:
    home: https://github.com/r-lib/pkgdown/
    source: https://github.com/r-lib/pkgdown/blob/master/
    issue: https://github.com/r-lib/pkgdown/issues/
    user: https://github.com/
```

- `home`: path to package home on source code repository.
- source:: path to source of individual file in master branch.
- `issue`: path to individual issue.
- `user`: path to user.

The varying components (e.g. path, issue number, user name) are pasted on the end of these URLs so they should have trailing `/`s.

pkgdown defaults to using the "master" branch for source file URLs. This can be configured to use a specific branch when linking to source files by specifying a branch name:

```
repo:
  branch: main
```

YAML config - deploy

`deploy` currently offers a single parameter:

- `install_metadata` allows you to install package index metadata into the package itself. Normally this metadata is made available on the published site; installing it into your package means that it's available for autolinking even if your website is not reachable at build time (e.g. because it's only behind the firewall or requires auth).

  ```
  deploy:
    install_metadata: true
  ```

Options

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.

Users can set a timeout for `build_site(new_process = TRUE)` with `options(pkgdown.timeout = Inf)`, which is useful to prevent stalled builds from hanging in cron jobs.

Examples

```
## Not run:
build_site()

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

## End(Not run)
```

`build_tutorials` *Build tutorials section*

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 _pkgdown.yml
`preview`	If `TRUE`, or `is.na(preview) && interactive()`, 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` *Clean site*

Description

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

Usage

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

Arguments

`pkg`	Path to package.

`deploy_site_github` *Deploy a pkgdown site on Travis-CI to Github Pages*

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. `deploy_to_branch()` can be used to deploy a site directly to GitHub Pages locally. See 'Setup' for details on setting up your repository to use this.

Usage

```
deploy_site_github(
  pkg = ".",
  install = TRUE,
  tarball = Sys.getenv("PKG_TARBALL", ""),
  ssh_id = Sys.getenv("id_rsa", ""),
  commit_message = construct_commit_message(pkg),
  clean = FALSE,
  verbose = FALSE,
  host = "github.com",
  ...,
  repo_slug = Sys.getenv("TRAVIS_REPO_SLUG", "")
)
```

Arguments

pkg	Path to package.
install	Optionally, opt-out of automatic installation. This is necessary if the package you're documenting is a dependency of pkgdown
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 *not* be your personal private key. Instead create a new keypair specifically for deploying the site. The easiest way is to use `travis::use_travis_deploy()`.
commit_message	The commit message to be used for the commit.
clean	Clean all files from old site.
verbose	Print verbose output
host	The GitHub host url.
...	Additional arguments passed to `build_site()`.
repo_slug	The user/repo slug for the repository.

Setup

For a quick setup, you can use `usethis::use_pkgdown_travis()`. It will help you with the following detailed steps.

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

```
before_cache: 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.yaml:

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

`deploy_to_branch` *Build and deploy a site locally*

Description

Assumes that you're in a git clone of the project, and the package is already installed.

Usage

```
deploy_to_branch(
  pkg = ".",
  commit_message = construct_commit_message(pkg),
  clean = FALSE,
  branch = "gh-pages",
  remote = "origin",
  github_pages = (branch == "gh-pages"),
  ...
)
```

Arguments

`pkg`	Path to package.
`commit_message`	The commit message to be used for the commit.
`clean`	Clean all files from old site.
`branch`	The git branch to deploy to
`remote`	The git remote to deploy to
`github_pages`	Is this a GitHub pages deploy. If `TRUE`, adds a `CNAME` file for custom domain name support, and a `.nojekyll` file to suppress jekyll rendering.
`...`	Additional arguments passed to `build_site()`.

init_site *Initialise site infrastructure*

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

Favicons are built automatically from a logo PNG or SVG by `init_site()` and copied to `pkgdown/favicon`.

404

pkgdown creates a default 404 page (404.html). You can customize 404 page content using .github/404.md.

in_pkgdown *Determine if code is executed by pkgdown*

Description

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

Usage

```
in_pkgdown()
```

Examples

```
in_pkgdown()
```

preview_site *Open site in browser*

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 *Translate an Rd string to its HTML output*

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 *Render page with template*

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: • `site`: `title` and path to `root`. • `yaml`: the `template` key from _pkgdown.yml. • `package`: package metadata including `name` and`version`. See the full contents by running `data_template()`.
`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 *Generate YAML templates*

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