Package: pkgdown (via r-universe)

September 30, 2024

Title Make Static HTML Documentation for a Package

Version 2.1.1.9000

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 (>= 4.0.0)

Imports bslib (>= 0.5.1), callr (>= 3.7.3), cli (>= 3.6.1), desc (>= 1.4.0), digest, downlit (>= 0.4.4), fontawesome, fs (>= 1.4.0), httr2 (>= 1.0.2), jsonlite, openssl, purrr (>= 1.0.0), ragg, rlang (>= 1.1.4), rmarkdown (>= 2.27), tibble, whisker, withr (>= 2.4.3), xml2 (>= 1.3.1), yaml

Suggests covr, diffviewer, evaluate (>= 0.24.0), gert, gt, htmltools, htmlwidgets, knitr, lifecycle, magick, methods, pkgload (>= 1.0.2), quarto, rsconnect, rstudioapi, rticles, sass, testthat (>= 3.1.3), tools

VignetteBuilder knitr, quarto

Config/Needs/website usethis, servr

Config/potools/style explicit

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first build-article, build-quarto-article, build-reference

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements pandoc

Repository https://r-lib.r-universe.dev

RemoteUrl https://github.com/r-lib/pkgdown

RemoteRef HEAD

RemoteSha 0cb9d38d1dc25946f01ee7500be9665985d8784e

Contents

`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.
- 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,
  seed = 1014L,
  override = list(),
  preview = FALSE
)

build_article(
  name,
  pkg = ".",
  lazy = FALSE,
  seed = 1014L,
  new_process = TRUE,
  pandoc_args = character(),
  override = list(),
  quiet = TRUE
)

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

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.
seed	Seed used to initialize random number generation in order to make article output reproducible. An integer scalar or NULL for no seed.
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.
new_process	Build the article in a clean R process? The default, TRUE, ensures that every article is build in a fresh environment, but you may want to set it to FALSE to make debugging easier.
pandoc_args	Pass additional arguments to pandoc. Used for testing.

Index and navbar

You can control the articles index and navbar with a `articles` field in your `_pkgdown.yml`. If you use it, pkgdown will check that all articles are included, and will error if you have missed any.

The `articles` field 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 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/pkgdown` 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`.)

Missing articles:

pkgdown will warn if there are (non-internal) articles that aren't listed in the articles index. You can suppress such warnings by listing the affected articles in a section with `title: internal` (case sensitive); this section will not be displayed on the index page.

External articles:

You can link to arbitrary additional articles by adding an `external-articles` entry to `_pkgdown.yml`. It should contain an array of objects with fields `name`, `title`, `href`, and `description`.

```
external-articles:
- name: subsampling
  title: Subsampling for Class Imbalances
 description: Improve model performance in imbalanced data sets through undersampling or oversampling
  href: https://www.tidymodels.org/learn/models/sub-sampling/
```

If you've defined a custom articles index, you'll need to include the name in one of the `contents` fields.

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.

Output formats

By default, pkgdown builds all articles using the rmarkdown::html_document() output format, ignoring whatever is set in your YAML metadata. This is necessary because pkgdown has to integrate the HTML/CSS/JS from the vignette with the HTML/CSS/JS from rest of the site. Because of the challenges of combining two sources of HTML/CSS/JS, there is limited support for other output formats and you have to opt-in by setting the as_is field in your .Rmd metadata:

```
pkgdown:
  as_is: true
```

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

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

To work with pkgdown, the output format must accept template, theme, and self_contained arguments, and must work without any additional CSS or JSS files. Note that if you use _output.yml or _site.yml you'll still need to add as_is: true to each individual vignette.

Additionally, htmlwidgets do not work when as_is: true.

Suppressing vignettes

If you want articles that are not vignettes, use usethis::use_article() to create it. 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.

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
other.parameters: []
```

Most of these parameters are interpreted similarly to knitr chunk options. `other.parameters` is a list of parameters that will be available to custom graphics output devices such as HTML widgets.

See Also

Other site components: `build_home()`, `build_news()`, `build_reference()`, `build_tutorials()`

---

build_favicons *Initialise 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`, created with `usethis::use_logo()`) 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.

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. (`usethis::use_logo()` does this for you!)

Usage

```
build_favicons(pkg = ".", overwrite = FALSE)
```

Arguments

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

build_home *Build home section*

Description

build_home() function generates pages at the top-level of the site including:

- The home page
- HTML files from any .md files in ./ or .github/.
- The authors page (from DESCRIPTION)
- The citation page (from inst/CITATION, if present).
- The license page
- A default 404 page if .github/404.md is not found.

build_home_index() rebuilds just the index page; it's useful for rapidly iterating when experimenting with site styles.

Usage

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

build_home_index(pkg = ".", override = list(), 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 main content of 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.

Extra markdown files in the base directory (e.g. ROADMAP.md) or in .github/ (e.g. CODE_OF_CONDUCT.md) are copied by build_home() to docs/ and converted to HTML.

The home page also features a sidebar with information extracted from the package. You can tweak it via the configuration file, to help make the home page an as informative as possible landing page.

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 chunk with `include = FALSE`.

````
```{r chunk-name, include=FALSE}`r ''`
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 <img src="man/figures/logo.png" align="right" />
```

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

YAML config - title and description:
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 on search engines, 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 > i.e. "YAML pipes"; this is a convenient way of writing paragraphs of text.)

Dev badges:
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()`. There should always be an empty line after the `<!-- badges: end -->` line. If you divide badges into paragraphs, make sure to add an empty line before the `<!-- badges: end -->` line.
- Any image-containing links within `<div id="badges"></div>`.
- Within the first paragraph, if it only contains image-containing links.

Identified badges are **removed** from the *main content*. They are shown or not in the *sidebar* depending on the development mode and sidebar customization, see the sidebar section.

Authors

By default, pkgdown will display author information in three places:

- the sidebar,
- the left part side of the footer,
- the author page.

This documentation describes how to customise the overall author display. See `?build_home` and `?build_site` for details about changing the location of the authors information within the home sidebar and the site footer.

Authors ORCID and bio:

Author ORCID identification numbers in the `DESCRIPTION` are linked 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")
    )
  )
```

If you want to add more details about authors or their involvement with the package, you can use the comment field, which will be rendered on the authors page.

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

Additional control via YAML:

You can control additinal aspects of the authors display via the `authors` YAML field:

- display of each author in the footer, sidebar and authors page,
- which authors (by role) are displayed in the sidebar and footer,
- text before authors in the footer,
- text before and after authors in the sidebar,
- text before and after authors on the authors page.

You can modify how each author's name is displayed by adding a subsection for `authors`. Each entry in `authors` should be named 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. Use an absolute URL to an image, not a relative link. Use an empty alternative text rather than no alternative text so a screen-reader would skip over it.

```
authors:
  firstname lastname:
    href: "http://name-website.com"
    html: "<img src='https://website.com/name-picture.png' width=72 alt=''>"
```

By default, the "developers" list shown in the sidebar and footer is populated by the maintainer ("cre"), authors ("aut"), and funder ("fnd") from the `DESCRIPTION`. You could choose other roles for filtering. With the configuration below:

- only the maintainer and funder(s) appear in the footer, after the text "Crafted by",
- all authors and contributors appear in the sidebar,
- the authors list on the sidebar is preceded and followed by some text,
- the authors list on the authors page is preceded and followed by some text.

```
authors:
  footer:
    roles: [cre, fnd]
    text: "Crafted by"
  sidebar:
    roles: [aut, ctb]
    before: "So *who* does the work?"
    after: "Thanks all!"
  before: "This package is proudly brought to you by:"
  after: "See the [changelog](news/index.html) for other contributors. :pray:"
```

If you want to filter authors based on something else than their roles, consider using a custom sidebar/footer component (see `?build_home`/`?build_site`, respectively).

Sidebar

You can customise the homepage sidebar with the `home.sidebar` field. It's made up of two pieces: `structure`, which defines the overall layout, and `components`, which defines what each piece looks like. This organisation makes it easy to mix and match the pkgdown defaults with your own customisations.

This is the default structure:

```
home:
 sidebar:
   structure: [links, license, community, citation, authors, dev]
```

These are drawn from seven built-in components:

- `links`: automated links generated from `URL` and `BugReports` fields from `DESCRIPTION` plus manual links from the `home.links` field:

```
home:
  links:
  - text: Link text
    href: https://website.com
  - text: Roadmap
    href: /roadmap.html
```

- `license`: Licensing information if `LICENSE`/`LICENCE` or `LICENSE.md`/`LICENCE.md` files are present.
- `community`: links to to `.github/CONTRIBUTING.md`, `.github/CODE_OF_CONDUCT.md`, etc.
- `citation`: link to package citation information. Uses either `inst/CITATION` or, if absent, information from the `DESCRIPTION`.
- `authors`: selected authors from the `DESCRIPTION`.
- `dev`: development status badges extracted from `README.md`/`index.md`. This is only shown for "development" versions of websites; see "Development mode" in `?build_site` for details.
- `toc`: a table of contents for the README (not shown by default).

You can also add your own components, where `text` is markdown text:

```
home:
  sidebar:
    structure: [authors, custom, toc, dev]
    components:
      custom:
        title: Funding
        text: We are *grateful* for funding!
```

Alternatively, you can provide a ready-made sidebar HTML:

```
home:
  sidebar:
    html: path-to-sidebar.html
```

Or completely remove it:

```
home:
  sidebar: FALSE
```

See Also

Other site components: `build_articles()`, `build_news()`, `build_reference()`, `build_tutorials()`

build_news *Build news section*

Description

A NEWS.md will be broken up into versions using level one (#) or level two headings (##) that (partially) match one of the following forms (ignoring case):

- {package name} 1.3.0
- {package name} v1.3.0
- Version 1.3.0
- Changes in 1.3.0
- Changes in v1.3.0

Usage

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

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

A common structure for news files is to use a top level heading for each release, and use a second level heading to break up individual bullets into sections.

```
# foofy 1.0.0

## Major changes

* Can now work with all grooveable grobbles!

## Minor improvements and bug fixes

* Printing scrobbles no longer errors (@githubusername, #100)

* Wibbles are now 55% less jibbly (#200)
```

Issues and contributors will be automatically linked to the corresponding pages on GitHub if the GitHub repo can be discovered from the DESCRIPTION (typically from a URL entry containing github.com)

If a version is available on CRAN, the release date will automatically be added to the heading (see below for how to suppress); if not available on CRAN, "Unreleased" will be added.

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

Other site components: `build_articles()`, `build_home()`, `build_reference()`, `build_tutorials()`

`build_redirects` *Build redirects*

Description

If you change the structure of your documentation (by renaming vignettes or help topics) you can setup redirects from the old content to the new content. One or several now-absent pages can be redirected to a new page (or to a new section of a new page). This works by creating a html page that performs a "meta refresh", which isn't the best way of doing a redirect but works everywhere that you might deploy your site.

The syntax is the following, with old paths on the left, and new paths or URLs on the right.

```
redirects:
  - ["articles/old-vignette-name.html", "articles/new-vignette-name.html"]
 - ["articles/another-old-vignette-name.html", "articles/new-vignette-name.html"]
 - ["articles/yet-another-old-vignette-name.html", "https://pkgdown.r-lib.org/dev"]
```

If for some reason you choose to redirect an existing page make sure to exclude it from the search index, see `?build_search`.

Usage

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

Arguments

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

---

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 = 1014L,
  override = list(),
  preview = FALSE,
  devel = TRUE,
  topics = NULL
)

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

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 random number generation in order to make article output reproducible. An integer scalar or `NULL` for no seed.
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.
topics	Build only specified topics. If supplied, sets lazy and preview to FALSE.

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()`.
- Topics from other installed packages, e.g. `rlang::is_installed()` (function name) or `sass::font_face` (topic name).
- `has_lifecycle("deprecated")` will select all topics with lifecycle deprecated.

All functions (except for `has_keyword()`) 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 error if you have missed any.

Missing topics:

pkgdown will warn if there are (non-internal) topics that not listed in the reference index. You can suppress these warnings by listing the topics in section with "title: internal" (case sensitive) which will not be displayed on the reference index.

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.

Examples

If you need to run extra code before or after all examples are run, you can create `pkgdown/pre-reference.R` and `pkgdown/post-reference.R`.

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
  other.parameters: []
```

Most of these parameters are interpreted similarly to knitr chunk options. `other.parameters` is a list of parameters that will be available to custom graphics output devices such as HTML widgets.

See Also

Other site components: `build_articles()`, `build_home()`, `build_news()`, `build_tutorials()`

`build_search` *Build search index*

Description

Generate a JSON search index from the built site. This is used by fuse.js to provide a javascript powered search for BS5 powered pkgdown sites.

NB: `build_search()` is called automatically by `build_site()`; you don't need call it yourself. This page documents how it works and its customisation options.

Usage

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

Arguments

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

YAML config

You can exclude some paths from the search index using `search.exclude`. Below we exclude the changelog from the search index:

```
search:
  exclude: ['news/index.html']
```

Debugging and local testing

Locally (as opposed to on GitHub Pages or Netlify for instance), search won't work if you simply use pkgdown preview of the static files. You can use `servr::httw("docs")` instead.

If search is not working, run `pkgdown::pkgdown_sitrep()` to eliminate common issues such as the absence of URL in the pkgdown configuration file of your package.

`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()`
- `build_redirects()`

See the documentation for the each function to learn how to control that aspect of the site. This page documents options that affect the whole site.

Usage

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

Arguments

`pkg`	Path to package.
`examples`	Run examples?
`run_dont_run`	Run examples that are surrounded in \dontrun?

seed	Seed used to initialize random number generation in order to make article output reproducible. An integer scalar or NULL for no seed.
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.

General config

- destination controls where the site will be generated, defaulting to docs/. Paths are relative to the package root.
- url is optional, but strongly recommended.

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

  It specifies where the site will be published and is used to allow other pkgdown sites to link to your site when needed (vignette("linking")), generate a sitemap.xml, automatically generate a CNAME when deploying to github, generate the metadata needed rich social "media cards" (vignette("metadata")), and more.
- title overrides the default site title, which is the package name. It's used in the page title and default navbar.

Navbar and footer

The navbar and footer fields control the appearance of the navbar footer which appear on every page. Learn more about these fields in vignette("customise").

Development mode

The development field allows you to generate different sites for the development and released versions of your package. To use it, you first need to set the development mode:

```
development:
  mode: auto
```

Setting development mode:

The development mode of a site controls where the built site is placed and how it is styled (i.e. the colour of the package version in the navbar, the version tooltip), and whether or not the site is indexed by search engines. There are four possible modes:

- **automatic** (`mode: auto`): determines the mode based on the version:
  - `0.0.0.9000` (`0.0.0.*`): unreleased.
  - four version components: development.
  - everything else -> release.
- **release** (`mode: release`), the default. Site is written to `docs/` and styled like a released package, even if the content is for an unreleased or development version. Version in navbar gets the default colouring. Development badges are not shown in the sidebar (see `?build_home`).
- **development** (`mode: devel`). Site is written to `docs/dev/`. The navbar version gets a "danger" class and a tooltip 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. Development badges are shown in the sidebar (see `?build_home`).
- **unreleased** (`mode: unreleased`). Site is written to `docs/`. Version in navbar gets the "danger" class, and a message indicating the package is not yet on CRAN. Development badges are shown in the sidebar (see `?build_home`).

Use `mode: auto` if you want both a released and a dev site, and `mode: release` if you just want a single site. It is very rare that you will need either devel or unreleased modes.

You can override the mode specified in the `_pkgdown.yml` by setting by setting `PKGDOWN_DEV_MODE` to `devel` or `release`.

Selective HTML:

You can selectively show HTML only on the devel or release site by adding class `pkgdown-devel` or `pkgdown-release`. This is most easily accessed from `.Rmd` files where you can use pandoc's `<div>` syntax to control where a block of markdown will display. For example, you can use the following markdown in your README to only show GitHub install instructions on the development version of your site:

```
::: {.pkgdown-devel}
You can install the development version of pkgdown from GitHub with:
`remotes::install_github("r-lib/pkgdown")`
:::
```

You can use a similar technique to control where badges are displayed. This markdown show the CRAN status badge on the site for the released package and the GitHub check status for the development package:

```
[![CRAN Status](https://www.r-pkg.org/badges/version/pkgdown)]
  (https://cran.r-project.org/package=pkgdown){.pkgdown-release}
[![R-CMD-check](https://github.com/r-lib/pkgdown/workflows/R-CMD-check/badge.svg)]
  (https://github.com/r-lib/pkgdown/actions){.pkgdown-devel}
```

Other options:

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`.

Template

The `template` field is mostly used to control the appearance of the site. See `vignette("customise")` for details. But it's also used to control

Other aspects:

There are a few other `template` fields that control other aspects of the site:

- `noindex: true` will suppress indexing of your pages by search engines:

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

- `google_site_verification` allows you to verify your site with google:

  ```
  template:
    params:
      google_site_verification: _nn6ile-a6x6lctOW
  ```

- `trailing_slash_redirect: true` will automatically redirect `your-package-url.com` to `your-package-url.com/`, using a JS script added to the `<head>` of the home page. This is useful in certain redirect scenarios.

  ```
  template:
    trailing_slash_redirect: true
  ```

Analytics:

To capture usage of your site with a web analytics tool, you can make use of the `includes` field to add the special HTML they need. This HTML is typically placed `in_header` (actually in the `<head>`), `before_body`, or `after_body`. You can learn more about how includes work in pkgdown at https://pkgdown.r-lib.org/articles/customise.html#additional-html-and-files.

I include a few examples of popular analytics platforms below, but we recommend getting the HTML directly from the tool:

- plausible.io:

  ```
  template:
    includes:
      in_header: |
      <script defer data-domain="{YOUR DOMAIN}" src="https://plausible.io/js/plausible.js"></scri
  ```

- Google analytics:

```
template:
  includes:
    in_header: |
       <!-- Global site tag (gtag.js) - Google Analytics -->
     <script async src="https://www.googletagmanager.com/gtag/js?id={YOUR TRACKING ID}"#' ></scr
       <script>
         window.dataLayer = window.dataLayer || [];
         function gtag(){dataLayer.push(arguments);}
         gtag('js', new Date());

         gtag('config', '{YOUR TRACKING ID}');
       </script>
```

- GoatCounter:

```
template:
  includes:
    after_body: >
    <script data-goatcounter="https://{YOUR CODE}.goatcounter.com/count" data-goatcounter-setti
```

Source repository

Use the `repo` field to override pkgdown's automatically discovery of your source repository. This is used in the navbar, on the homepage, in articles and reference topics, and in the changelog (to link to issue numbers and user names). pkgdown can automatically figure 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` field:

```
repo:
  url:
    home: https://github.com/r-lib/pkgdown/
    source: https://github.com/r-lib/pkgdown/blob/HEAD/
    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 default branch (more on that below).
- `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.

When creating the link to a package source, we have to link to a specific branch. The default behaviour is to use current branch when in GitHub actions and `HEAD` otherwise. You can overide this default with `repo.branch`:

```
repo:
  branch: devel
```

pkgdown can automatically link to Jira issues as well if specify both a custom `issue` URL as well Jira project names to auto-link in `jira_projects`. You can specify as many projects as you would like:

```
repo:
  jira_projects: [this_project, another_project]
  url:
    issue: https://jira.organisation.com/jira/browse/
```

Deployment (`deploy`)

There is a single `deploy` field

- `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 behind a 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_site_github_pages` *Build site for GitHub pages*

Description

Designed to be run as part of automated workflows for deploying to GitHub pages. It cleans out the old site, builds the site into `dest_dir` adds a `.nojekyll` file to suppress rendering by Jekyll, and adds a `CNAME` file if needed.

It is designed to be run in CI, so by default it:

- Cleans out the old site.
- Does not install the package.
- Runs `build_site()` in process.

Usage

```
build_site_github_pages(
  pkg = ".",
  ...,
  dest_dir = "docs",
  clean = TRUE,
  install = FALSE,
  new_process = FALSE
)
```

Arguments

`pkg`	Path to package.
`...`	Additional arguments passed to `build_site()`.
`dest_dir`	Directory to build site in.
`clean`	Clean all files from old site.
`install`	If `TRUE`, will install the package in a temporary library so it is available for vignettes.
`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.

`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 = FALSE)
```

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/
```

See Also

Other site components: build_articles(), build_home(), build_news(), build_reference()

---

check_pkgdown *Check* _pkgdown.yml

Description

This pair of functions checks that your _pkgdown.yml is valid without building the whole site. check_pkgdown() errors at the first problem; pkgdown_sitrep() reports the status of all checks.

Currently they check that:

- There's a url in the pkgdown configuration, which is also recorded in the URL field of the DESCRIPTION.
- All opengraph metadata is valid.
- All reference topics are included in the index.
- All articles/vignettes are included in the index.

Usage

```
check_pkgdown(pkg = ".")

pkgdown_sitrep(pkg = ".")
```

Arguments

pkg	Path to package.

---

clean_site *Clean site*

Description

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

Delete all files in the pkgdown cache directory.

Usage

```
clean_site(pkg = ".", quiet = FALSE)

clean_cache(pkg = ".", quiet = FALSE)
```

Arguments

pkg	Path to package.
quiet	If TRUE, suppresses a message.

---

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. Use usethis::use_pkgdown_github_p to automate this process using GitHub actions.

Usage

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

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().
subdir	The sub-directory where the site should be built on the branch. This argument can be used to support a number of site configurations. For example, you could build version-specific documentation by setting subdir = "v1.2.3"; deploy_to_branch() will build and deploy the package documentation in the v.1.2.3/ directory of your site.

---

init_site *Initialise site infrastructure*

Description

init_site():

- creates the output directory (docs/),
- generates a machine readable description of the site, used for autolinking,
- copies CSS/JS assets and extra files, and
- runs build_favicons(), if needed.

Typically, you will not need to call this function directly, as all build_*() functions will run init_site() if needed.

The only good reasons to call init_site() directly are the following:

- If you add or modify a package logo.
- If you add or modify pkgdown/extra.scss.
- If you modify template.bslib variables in _pkgdown.yml.

See vignette("customise") for the various ways you can customise the display of your site.

Usage

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

Arguments

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

Build-ignored files

We recommend using `usethis::use_pkgdown_github_pages()` 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.

---

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

`preview_site()` opens your pkgdown site in your browser. pkgdown has been carefully designed to work even when served from the file system like this; the only part that doesn't work is search. You can use `servr::httw("docs/")` to create a server to make search work locally.

Usage

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

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 andversion. See the full contents by running data_template().
path	Location to create file; relative to destination directory.
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