

Package Development:

The Rest of the Owl

GitHub repo: pos.it/pkg-dev-conf25

Wifi: Posit Conf 2025 | conf2025

Discord: #workshop-pkg-dev

How to draw an owl

1.



1. Draw some circles

2.



2. Draw the rest of the owl



2

roxygen2



knitr

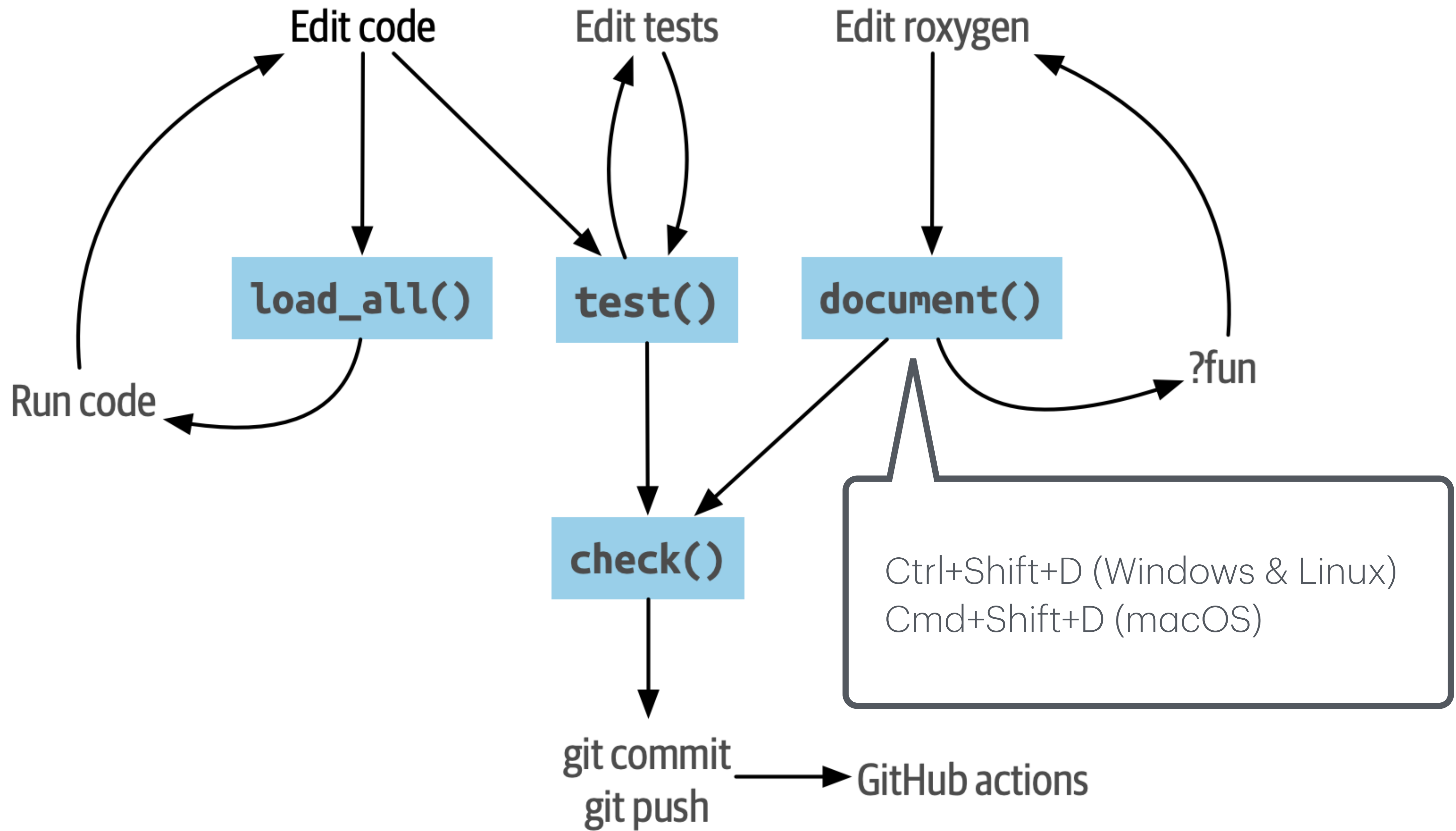
📁 .github
📁 R
📁 inst
📁 man
📁 pkgdown/favicon
📁 revdep
📁 tests
📁 vignettes
📄 .Rbuildignore
📄 .covrignore
📄 .gitignore
📄 DESCRIPTION
📄 LICENSE
📄 LICENSE.md
📄 NAMESPACE
📄 NEWS.md
📄 README.Rmd
📄 README.md
📄 _pkgdown.yml

Help topics live here, in **.Rd** files

But you (mostly) don't write to **man/** with your bare hands.

📁 .github
📁 R
📁 inst
📁 man
📁 pkgdown/favicon
📁 revdep
📁 tests
📁 vignettes
📄 .Rbuildignore
📄 .covrignore
📄 .gitignore
📄 DESCRIPTION
📄 LICENSE
📄 LICENSE.md
📄 NAMESPACE
📄 NEWS.md
📄 README.Rmd
📄 README.md
📄 _pkgdown.yml

Help topics are generated from the roxygen comments you write alongside your functions (or other objects), inside **R/**.



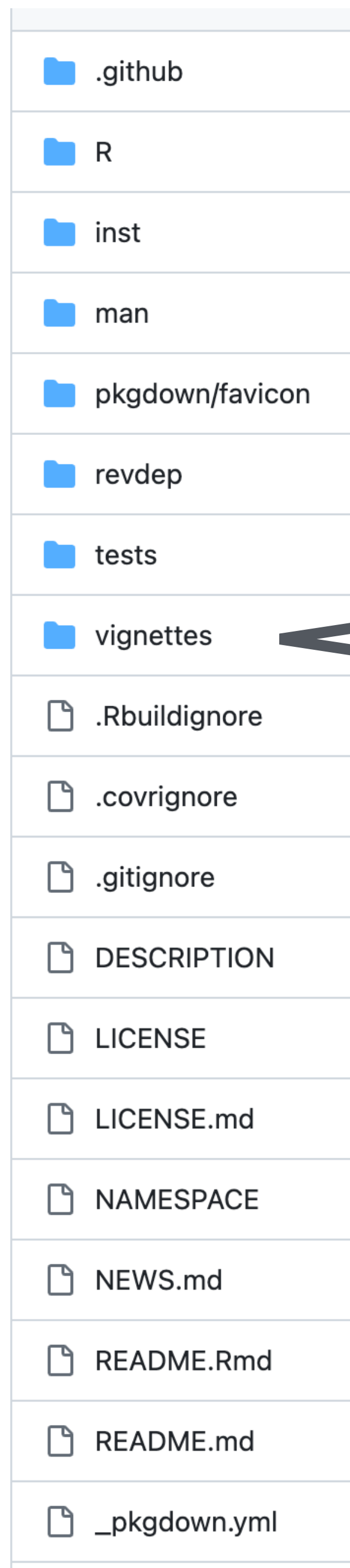
📁 .github
📁 R
📁 inst
📁 man
📁 pkgdown/favicon
📁 revdep
📁 tests
📁 vignettes
📄 .Rbuildignore
📄 .covrignore
📄 .gitignore
📄 DESCRIPTION
📄 LICENSE
📄 LICENSE.md
📄 NAMESPACE
📄 NEWS.md
📄 README.Rmd
📄 README.md
📄 _pkgdown.yml

`README.md` is important:

- on CRAN
- on GitHub
- in the pkgdown site (it's the default `index.html`)

Ideally, `README.md` is generated from `README.Rmd`.

`build_readme()` is the best way to convert `README.Rmd` to `README.md`, because it actually installs the current dev package prior to rendering.



Vignettes are a great way to show truly authentic usage, combining multiple functions in your package to accomplish a realistic task.

If authentic usage involves moves that are impossible on CRAN, demonstrate that in an **article** instead. An article appears only in the pkgdown site.

folder	.github
folder	R
folder	inst
folder	man
folder	pkgdown/favicon
folder	revdep
folder	tests
folder	vignettes
file	.Rbuildignore
file	.covrignore
file	.gitignore
file	DESCRIPTION
file	LICENSE
file	LICENSE.md
file	NAMESPACE
file	NEWS.md
file	README.Rmd
file	README.md
file	_pkgdown.yml

Personally, a user's experience with the pkgdown website is top of my mind when writing docs.

usethis 3.0.0

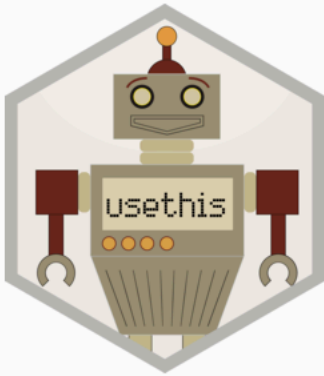
Setup

Reference

Articles ▾

News ▾

Search for



usethis

usethis is a workflow package: it automates repetitive tasks that arise during project setup and development, both for R packages and non-package projects.

Installation

Install the released version of usethis from CRAN:

```
install.packages("usethis")
```

LINKS

[View on CRAN](#)

[Browse source code](#)

[Report a bug](#)

LICENSE

[Full license](#)

[MIT](#) + file [LICENSE](#)

COMMUNITY

[Contributing guide](#)

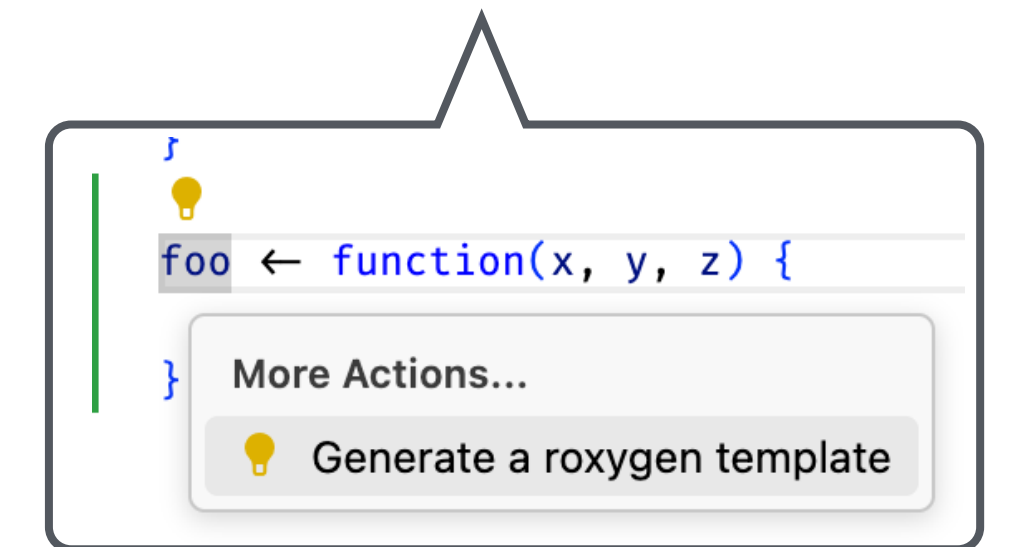
[Code of conduct](#)

[Getting help](#)

CITATION

Table stakes for a help topic

- Insert a roxygen skeleton in RStudio with *Code > Insert Roxygen Skeleton*. Or use Positron's *Generate a roxygen template* code action. Or do the equivalent "by hand".
- Complete the prepared fields
 - (Usually implicit) title and description
 - `@param` for function arguments
 - `@returns` for the return value
 - `@examples` for usage
 - `@export` to put in NAMESPACE
- Related workflow happiness: Cmd/Ctrl + Enter helps you develop your examples. Plays nicely with `load_all()` for staying synced with package source.



Typical roxygen comment

```
#' Remove duplicated strings
#'
#' `str_unique()` removes duplicated strings. Additional control over
#' how duplication is measured.
#'
#' @param string Input vector. Either a character vector, or something
#'   coercible to one.
#' @param ... Other options used to control matching behavior between duplicate
#'   strings. Passed on to [stringi::stri_opts_collator()].
#' @returns A character vector, usually shorter than `string`.
#' @seealso [unique()], [stringi::stri_unique()] which this function wraps.
#' @examples
#' str_unique(c("a", "b", "c", "b", "a"))
#'
#' # Use ... to pass additional arguments to stri_unique()
#' str_unique(c("motley", "mötley", "pinguino", "pingüino"))
#' str_unique(c("motley", "mötley", "pinguino", "pingüino"), strength = 1)
#' @export
str_unique ← function(string, ...) {
  ...
}
```

Title

Description

Parameters

What it returns

Examples

Export for
external use

Hard and/or fiddly but worth it

- Links!
 - Other help topics in same package
 - Topics in other packages
 - Vignettes in same or other package
 - Other URLs
- Examples
 - Show authentic usage that hopefully does not ...
 - Create huge headaches in CI and on CRAN

Backticks to format as code. Typical way to refer to a function in its own topic. Use trailing parentheses.

```
#' It's obvious that `thisfunction()` is better than  
#'[otherpkg::otherfunction()] or even our own [olderfunction()].
```

```
#' Read more in our own `vignette("stuff")` or elsewhere in  
#`vignette("things", package = "otherpkg")`.
```

```
#' It's obvious that `thisfunction()` is better than  
#'[otherpkg::otherfunction()] or even our own [olderfunction()].
```

Square brackets and parentheses for an auto-link to another function, in same package or other package.

See the [\(R\)Markdown support article, Function links](#)

```
#' Read elsewhere in  
#'\`vignette("things", package = "otherpkg")`.
```

```
#' It's obvious that `thisfunction()` is better than  
#'[otherpkg::otherfunction()] or even our own [olderfunction()].
```

This vignette syntax is auto-linked in pkgdown (and elsewhere) and gives working code, otherwise.

```
#' Read more in our own `vignette("stuff")` or elsewhere in  
#`vignette("things", package = "otherpkg")`.
```


URL links

```
#' Use C++ via the cpp11 package
```

```
#'
```

```
#' Adds infrastructure needed to use the [cpp11](https://cpp11.r-lib.org)
```

```
#' package, a header-only R package that helps R package developers handle R
```

```
#' objects with C++ code: ...
```



[link text](URL)



<URL>

```
#' See <https://happygitwithr.com/common-remote-setups.html> for more about
```

```
#' GitHub remote configurations and, e.g., what we mean by the source repo. This
```

```
#' function works for the configurations `"ours"`, `"fork"`, and `"theirs"`.
```

URL links

- "The goal of urlchecker is to run the URL checks from R 4.1 in older versions of R and automatically update URLs as needed."
- Extracts the URL-checking logic inside R `CMD check` and exposes as `urlchecker::url_check()`.

```
library(urlchecker)
```

```
# `url_check()` will check all URLs in a package, as is done by CRAN when  
# submitting a package.
```

```
url_check("path/to/pkg")
```

```
# `url_update()` will check all URLs in a package, then update any 301  
# redirects automatically to their new location.
```

```
url_update("path/to/pkg")
```

What's tricky about examples?

- Tension between
 - Showing readable and realistic code
 - However, there's
 - no user, i.e. for interaction or credentials
 - side effects are forbidden
 - can't throw an error
- It's tempting to use `\dontrun{}` but you can get pushback from CRAN

@examplesIf is a mighty weapon

Put something that evaluates to TRUE/FALSE here.

```
#' @examplesIf rlang::is_interactive()  
#'# load/refresh existing credentials, if available  
#'# otherwise, go to browser for authentication and authorization  
#'# drive_auth()  
#'  
#'# see user associated with current token  
#'# drive_user()
```

@examplesIf is a mighty weapon

Examples

```
if (FALSE) { # rlang::is_interactive()
# load/refresh existing credentials, if available
# otherwise, go to browser for authentication and authorization
drive_auth()

# see user associated with current token
drive_user()
```

@examplesIf loves a custom predicate

You can write a custom function to use here!

```
#' @examplesIf drive_has_token()  
#'# Create a blank Google Doc named 'WordStar' in  
#'# your 'My Drive' root folder and star it  
#'# wordstar ← drive_create("WordStar", type = "document", starred = TRUE)
```


@examplesIf loves a custom predicate

Examples

```
# Create a blank Google Doc named 'WordStar' in
# your 'My Drive' root folder and star it
wordstar <- drive_create("WordStar", type = "document", starred = TRUE)
#> Created Drive file:
#> • WordStar <id: 1fQ7yjQxlYG67jg4TKlTqTLkIFvgFYNIoxLNB6Bs78TI>
#> With MIME type:
#> • application/vnd.google-apps.document
```

Examples should not change the world

- If at all possible, just don't do anything like this:
 - write a file
 - set an option
 - change working directory
- If you must change the world, put it back the way you found it, e.g.
 - write to the temp directory AND delete the file
 - restore the option or working directory to original state
- Sadly, withr functions don't work here. Neither does **on.exit()**.
- Sadly, nothing like **@examplesIf** is available to hide the machinery.

Use `try()` to show an error

' @examples

! ...

```
#' # Row sizes must be compatible when column-binding
```

```
#' try(bind_cols(tibble(x = 1:3), tibble(y = 1:2)))
```

Put code that errors inside `try()`.

```
# Row sizes must be compatible when column-binding
try(bind_cols(tibble(x = 1:3), tibble(y = 1:2)))
#> Error in bind_cols(tibble(x = 1:3), tibble(y = 1:2)) :
#> Can't recycle `..1` (size 3) to match `..2` (size 2).
```

Vignette basics

- `use_vignette()` is a good way to initiate a vignette.
- `use_article()` initiates a article that only exists in the pkgdown website.
- Today these functions create an `.Rmd` document, by default.
- `use_vignette("cool-stuff.qmd")` initiates a Quarto vignette. Probably the right move for new vignettes and will become the default in the fullness of time.
- A vignette or article focuses on how to solve a target problem, versus documenting a specific function.

Vignette gotchas

- Vignettes can only use packages you formally depend on, i.e. they appear in **Imports** or **Suggests**
- Help topics live as **.Rd** files in a source package, but there's no equivalent for rendered vignettes
 - Creates some inherent awkwardness in vignette workflow
- Similar to examples, **R CMD check** treats vignettes like big, weird tests.
 - Creates tension between showing realistic usage and ensuring the code can be evaluated, 100% of the time and quickly.
 - Challenges: wrapping an external service, long-running code, etc.
 - Different techniques can be used to cope, chiefly **eval = FALSE**.

Some workflows that aren't terrible

- Interactive work: `load_all()` + usual gestures for developing code chunks
- See a draft: install the in-dev package + usual gesture for rendering .Rmd
- See a draft: install the in-dev package and request that vignettes be built, via `install(build_vignettes = TRUE)`
- See a draft: `devtools::build_rmd("vignettes/my-vignette.Rmd")`, which temporarily installs in-dev package before rendering
- See a draft: use pkgdown and preview your site locally or remotely

The **eval** chunk option is a mighty weapon

- **eval** chunk option controls whether code is evaluated
- Similar to **@examplesIf**, you can call a function to determine if it's safe to evaluate a chunk
- `eval = requireNamespace("somedependency")`
- `eval = !identical(Sys.getenv("SOME_THING_YOU_NEED"), "")`
- `eval = file.exists("credentials-you-need")`
- You can use **eval** local to individual chunks or set it globally in a setup chunk.

```
```{r setup, include = FALSE}
can_decrypt ← gargle::secret_can_decrypt("googlesheets4")
knitr::opts_chunk$set(
 collapse = TRUE,
 comment = "#>",
 error = TRUE,
 eval = can_decrypt
)
```

```{r eval = !can_decrypt, echo = FALSE, comment = NA}
message("No token available. Code chunks will not be evaluated.")
```

```{r index-auth, include = FALSE}
googlesheets4::gs4_auth_docs()
```
```

Your turn

Ideas for documentation-related activities:

<https://github.com/posit-conf-2025/pkg-dev/blob/main/documentation-prompts.md>

Feel free to share what you're doing, how it's going, etc. in the Discord channel.

| Big Idea | Featured implementation |
|--|---|
| Make writing docs fun(-ish) and easy(-ish), so you actually do it | Use roxygen2 and embrace (R)Markdown |
| Rich linking is a kindness to your user | roxygen2 syntax for auto-linking |
| README is your "welcome mat" | Use README.Rmd so you can show compelling usage |
| Users are motivated to solve a problem, not to use a specific function | Write vignettes/articles |
| Don't let the difficulty of realism in examples and vignettes defeat you | @examplesIf
eval
just make it an article |

