Writing your first documentation PR

The goal of this worksheet is to guide you through the steps needed to successfully create a documentation PR (pull request). Empty spaces have been placed on this sheet for your notes.

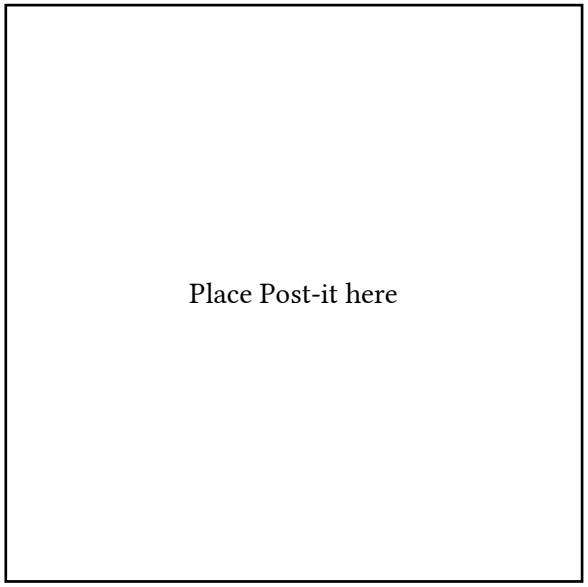
0. Get ready

- □ `install.packages("pak")`
- □ `pak::pak("devtools")`
- □ .Set up your GitHub personal access token.
  - `usethis.r-lib.org` -> Articles -> Managing Git(Hub) Credentials
- □ Call `usethis::git_sitrep()` and check that:
  - Your name and email address appears in "Git config (global)"
  - Your GitHub user name is found under "Github"

1. Find and claim an issue

Browse the Post-Its, looking for something of interest. If you've never done a PR before, we recommend that you start with a documentation issue since there are slightly fewer moving parts. We also encouraged you to team up with a (new) friend and tackle something together!

Once you've picked a Post-it, open the issue and read the details in full. At this point, you might discover the issue is outside your wheelhouse; if so, no problem, just return the Post-it to the wall and try again.



2. Get the source locally

- □ **Fork** and clone the repository using this syntax:
  - `usethis::create_from_github("  /  ")`
  - (e.g `usethis::create_from_github("tidyverse/ggplot2")`)
- □ `pak::local_install_dev_deps()` to make sure you've got the necessary packages.
- □ If you're working on a documentation issue, run `devtools::document()` to make sure you have everything necessary to update the docs before you make any changes.
- □ If you're working on a bug or feature, run `devtools::check()` to get a baseline.
- □ If running `devtools::documents()` or `devtools::check()` fails, something is probably off with the setup of your machine and you should seek help.

3. Make the change

- [ ] Create a new branch for your changes to live with
  - `usethis::pr_init("                    ")`
- [ ] Make your change.
  - If you're writing documentation, use `devtools::document()` to update the docs.
  - If you're writing code, add a test or two. Iterate quickly with `devtools::load_all()` and `devtools::test()`.
- [ ] `devtools::check()` to make sure all is still well.
- [ ] Commit the change with your git client.
- [ ] If the feature is user-facing (i.e. it adds a new feature or fixes a bug), add a bullet to NEWS. Be concise, link to the issue, and tag yourself.

```
* `use_circleci()` creates a `.circleci/config.yaml` config
file for CircleCI (#703, @jdblischak).
```

4. Submit your PR

Push and make a PR with `usethis::pr_push()`.

- [ ] Make the title descriptive:
  - Not good: "Working on issue 45"
  - Your title: "   "
- [ ] In the description (not the title), use the magic fixes keyword, e.g. "Fixes #123", to link & close your issue.
- [ ] Example with descriptive title and description containing magic keywords: https://github.com/r-lib/usethis/pull/742
- [ ] Write your PR number on the Post-it and move it to the review wall.
- [ ] Ring the gong to celebrate your successful submission!

5. Wait for review

A person from the tidyverse team will try and review your issue as quickly as possible, reading through your PR and offering suggestions for how to improve it. If you want to work on something else while you wait, use `pr_pause()` to pause your pull request while you work on other stuff. It's possible the reviewer might make changes directly to your review - if that happens you can use `pr_pull()` to get their code back onto your computer.

If you've committed a bigger PR, it might take a bit longer to review, and we might not be able to complete it on the day. Don't worry if this happens to you! We really appreciate your contribution and just need a bit more time to take it in.

Wrap up

At the end of the day, make sure that you've reinstalled the CRAN versions of any packages you might have installed development versions of. You can find a list of dev versions with this code:

```
subset(
  pak::lib_status(),
  as.numeric(package_version(version)[, 4]) >= 9000,
  c(package, version)
)
```