

DaSL Developer Guide

Scott Chamberlain

Table of contents

Welcome	3
Inspiration	3
1 Help	4
1.1 House Calls	4
1.2 Slack	4
2 Contributing	5
2.1 Code of Conduct	5
2.2 Communication Channels	5
2.3 xxx	5
3 Style	6
3.1 Package styler	6
3.2 IDE and Text editor support	6
4 Code review	7
4.1 DaSL interal guidelines	7
4.2 Community guidelines	7
5 Documentation	8
5.1 R	8
5.1.1 DaSL pkgdown template	8
5.2 Python	9
5.3 Guidelines	9
References	10

Welcome

This book is a resource for anyone in the [Fred Hutch Cancer Center Data Science Lab](#) community contributing to or using software.

The book attempts to cover all important aspects of software development, including how to get involved in software as a user or contributor, code style, code review, package documentation, and more. It includes both internal facing guidelines as well as for any contributions from folks other than DaSL staff.

- Getting help: [Chapter 1](#)
- Contributing guidelines: [Chapter 2](#)
- Style guidelines: [Chapter 3](#)
- Code review guidelines: [Chapter 4](#)
- Package documentation guidelines: [Chapter 5](#)

Inspiration

Inspiration for this guide is taken in part from:

- [Tidyverse Style Guide](#)
- [rOpenSci Dev Guide](#)

1 Help

Getting help

1.1 House Calls

XXXX

1.2 Slack

XXXX

2 Contributing

Contributing guidelines

2.1 Code of Conduct

xxxx

2.2 Communication Channels

- Slack

2.3 xxx

xxxx

3 Style

What is code style? It has to do with how you organize your code. The end goal of styling your code is that your code is more consistent. Styling code by definition has to be opinionated. You may not agree with every decision made in a style guide, but having style be done automatically for you gives you more time to think about the important decisions in your code.

If you always write code by yourself and you're the only one that will ever look at it, that's one thing. But that's rarely the case, especially if you consider that yourself 6 months or 5 years from now is a user that is keenly interested in readable, consistent code. Readable, consistent code will be especially appreciated by other users and contributors.

3.1 Package styler

The [styler](#) package allows you to interactively format your code according to the [tidyverse style guide](#).

The [lintr](#) package does automated checks of your code according to the style guide.

3.2 IDE and Text editor support

RStudio supports styler via the Addins drop down; see the [RStudio User Guide](#).

Support for `styler` in other editors is provided via the [R languageserver](#):

- VSCode: [vscode-R](#)
- Atom: [atom-ide-r](#)
- Sublime Text: [R-IDE](#)
- Vim/NeoVim: [LanguageClient-neovim](#)

See the [R languageserver](#) GitHub repository for more information on using the R language-server.

4 Code review

Code review

4.1 DaSL interal guidelines

XXXX

4.2 Community guidelines

XXXX

5 Documentation

All formally supported DaSL R and Python packages should have package documentation.

5.1 R

We use the [pkgdown](#) package to create documentation for R packages, and host it on [GitHub Pages](#). To get started with `pkgdown`, in R within the root of your package run `usethis::use_pkgdown_github_pages()` - it will set up a `_pkgdown.yml` file in the root of your repo used to configure `pkgdown`, and add a `.github/workflows/publish.yml` file to build the package documentation on each push, pull request, or release. See [pkgdown documentation](#) for details on configuring documentation.

After pushing the above changes up to your repository, go to the Actions tab and you should see the new `publish` action running. It will build and then deploy the rendered package docs. The URL for your docs will vary depending on the GitHub organization your repository lives within. For WILDS, the base URL for now is <https://getwilds.github.io>. If your package was in the WILDS org (at <https://github.com/getwilds>) your package (named `mypkg`) docs would live at <https://getwilds.github.io/mypkg>.

You can also build `pkgdown` docs locally - after running `usethis::use_pkgdown()` or `usethis::use_pkgdown_github_pages()` - by running `pkgdown::build_site()`. If you run `build_site()` within RStudio it should open up your site in your default browser, but may not do so if you run in a terminal. You can open the site in your browser by navigating to and opening the file `docs/index.html` within your repo.

5.1.1 DaSL pkgdown template

We are planning to have a DaSL specific `pkgdown` “package template” (see [pkgdown docs](#) for what this means) - but it’s not ready to use yet. When it is ready, you will be able to specify our template like:

```
template:  
  package: dasltemplate
```

For now just use the default theme that `pkgdown` provides.

5.2 Python

Just like there's a variety of ways to do packing in Python there's a variety of documentation options. Two of the well known options are:

- [Sphinx](#) - been around longer, uses [reStructuredText](#)
- [MkDocs](#) - newcomer, used [Markdown](#)

For either of the options above, they can be hosted in many places, including [GitHub Pages](#) and [ReadTheDocs](#).

`Sphinx` and `MkDocs` are less automatic relative to `pkgdown`, so just be prepared for a bit more manual work.

5.3 Guidelines

- README: This is most likely the first place potential users will interact with your package. Make sure the README clearly states what the package does, and how to get started.
- Examples: All user facing functions should have examples. Make sure to be careful about how examples are run if there's any sensitive data or connections to remote services.
- (anything else?)

References