

Standard Operating Procedure (SOP) for Internal Development

On behalf of the Neuroinformatics and Biostatistics (NIBS) platform

Updated as of March 1, 2021

Instructions for posting issues and bug prioritization for ONDRI applications.

Contents

| | | |
|----------|---|----------|
| 1 | Best Practices | 2 |
| 1.1 | Package version number | 2 |
| 1.2 | Commit messages | 3 |
| 1.3 | Branches | 3 |
| 1.4 | Package structure (For ONDRI ShinyApps) | 4 |
| 1.5 | Debugging (For ONDRI ShinyApps) | 4 |
| 2 | Review process | 5 |
| 2.1 | .gitignore | 5 |
| 2.2 | GitLab to GitHub | 5 |
| 3 | Glossary | 6 |
| 4 | References | 6 |

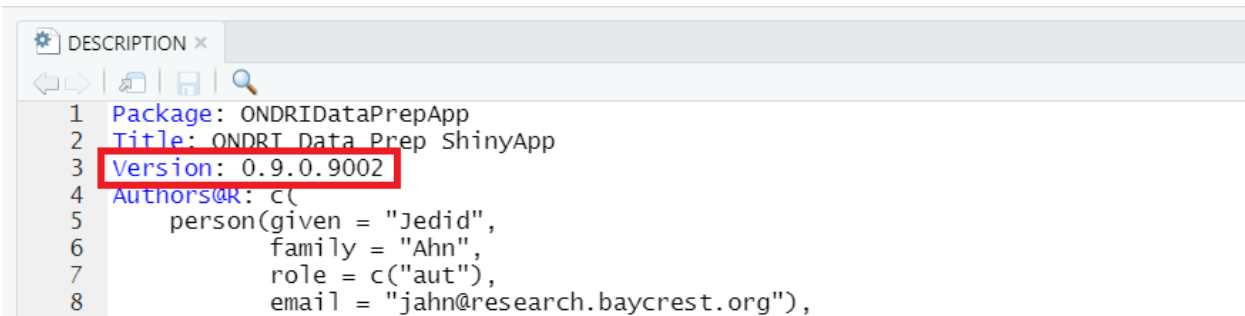
1 Best Practices

1.1 Package version number

Since all ONDRI-based R packages are currently under development, their version numbers are of the following format: **0.0.0.9000**. Hadley Wickham (2019) The fourth component represents the development version and should be incremented every time a patch or feature has been added. Hadley Wickham (2019) Until the packages are out of development phase, the fourth component will be edited most often.

The version numbers should be edited in the following three locations every time the package is updated:

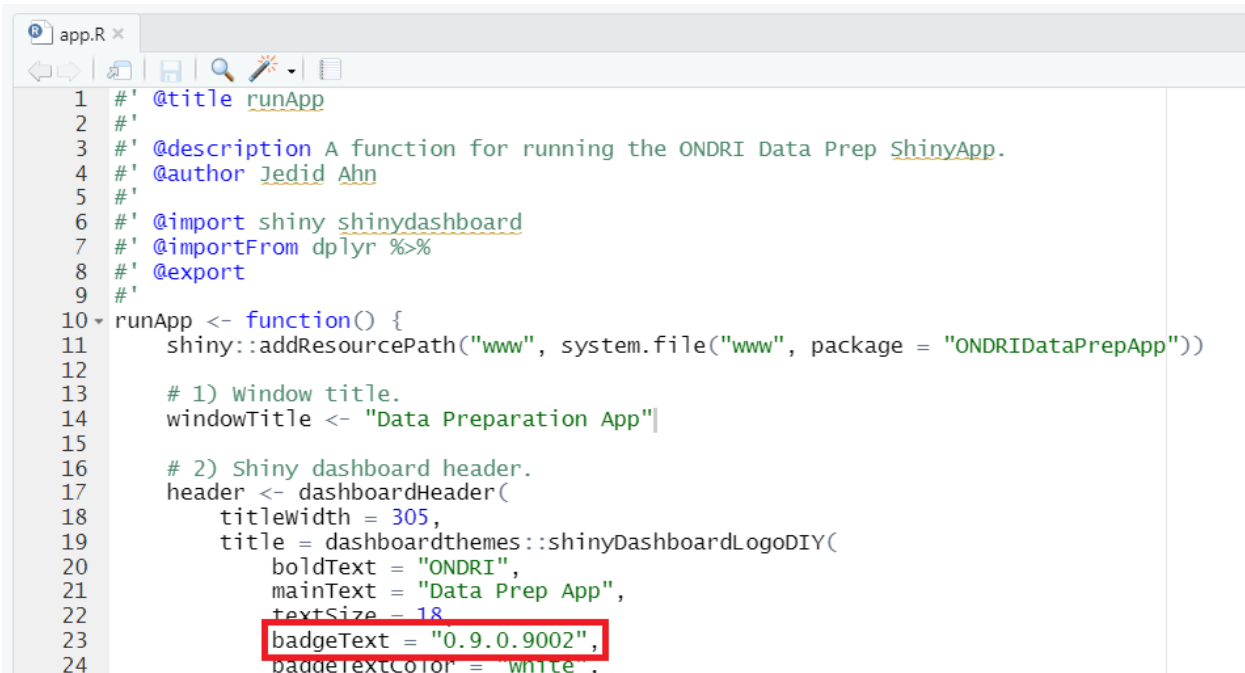
1. The DESCRIPTION file.



```
DESCRIPTION
1 Package: ONDRIDataPrepApp
2 Title: ONDRI Data Prep ShinyApp
3 Version: 0.9.0.9002
4 Authors@R: c(
5   person(given = "Jedid",
6     family = "Ahn",
7     role = c("aut"),
8     email = "jahn@research.baycrest.org"),
```

Figure 1: The version number will always be located in the third line of the DESCRIPTION file.

2. The app.R file, if it exists. This will show the version number in the user interface so users can keep track of the ONDRI ShinyApp version they are utilizing.



```
app.R
1 #' @title runApp
2 #'
3 #' @description A function for running the ONDRI Data Prep ShinyApp.
4 #' @author Jedid Ahn
5 #'
6 #' @import shiny shinydashboard
7 #' @importFrom dplyr %>%
8 #' @export
9 #'
10 runApp <- function() {
11   shiny::addResourcePath("www", system.file("www", package = "ONDRIDataPrepApp"))
12
13   # 1) window title.
14   windowTitle <- "Data Preparation App"
15
16   # 2) Shiny dashboard header.
17   header <- dashboardHeader(
18     titlewidth = 305,
19     title = dashboardthemes::shinyDashboardLogoDIY(
20       boldText = "ONDRI",
21       mainText = "Data Prep App",
22       textSize = 18,
23       badgeText = "0.9.0.9002",
24       badgeTextColor = "white",
```

Figure 2: The version number is located in the header of all ONDRI ShinyApps.

3. The corresponding reference guide, if it exists.

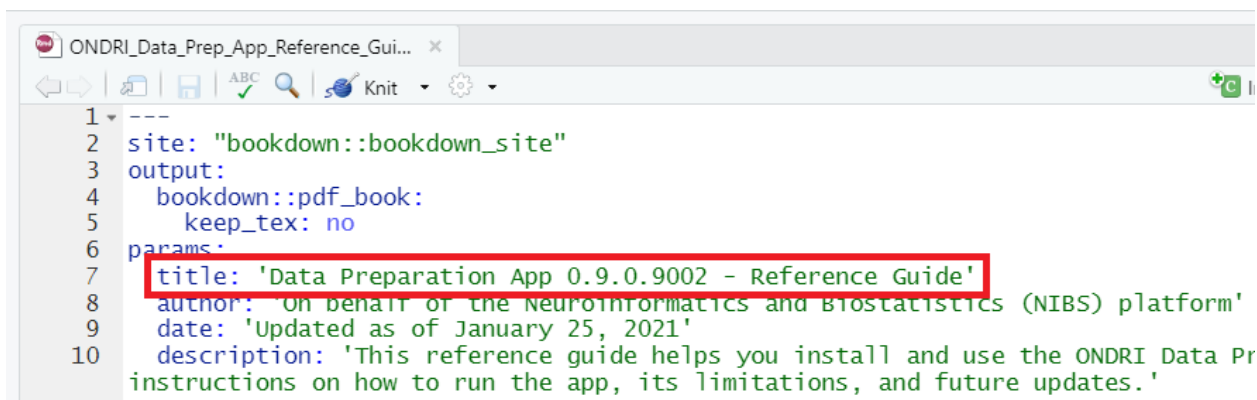


Figure 3: The version number will always be located in the RMarkdown title.

1.2 Commit messages

GitHub and GitLab commit messages should be professional and informative. They should outline whether the changes are bug fixes, a revamp of the code, enhancements/additions, or changes to documentation or the README. This will allow for easier navigation of previous commits for developers during debugging sessions.

For any repository housing an R package, a commit *intended for release to GitHub* should always include the new package version number in the commit message and in the following format:

release <version number> <message>.

The message should summarize the changes to the package between version numbers. These commits are considered stable and should be the only commits utilized if attempting to revert to a working copy of the package.

1.2.1 Examples of non-release messages

- REVAMP: Replaced gather and spread functions (which have been deprecated) with pivot_longer and pivot_wider respectively.
- DOCUMENTATION: Added screenshots to etc folder and updated README with new installation process.

1.2.2 Examples of release messages

- release 0.9.0.9002 Major revamp of variable selection (step 6) including rehaul of the UI and removal of drag and drop functionality.
- release 0.9.0.9004 Patch to fix frontend components including progress bar stuck at 0% and subject IDs not outputting on right side pane.

1.3 Branches

The default branch for all repositories in GitHub is the **main** branch. The **main** branch should be the only branch that exists.

The default branch for public facing repositories in GitLab is also the **main** branch. All other repositories including archived repositories have **master** as the default branch.

No repository in neither GitLab nor GitHub should contain both a **main** branch and a **master** branch. However, a GitLab repository can contain multiple development branches.

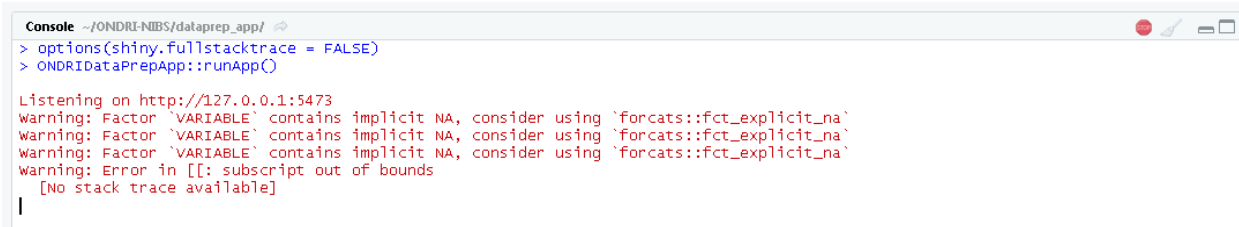
1.4 Package structure (For ONDRI ShinyApps)

As the ONDRI ShinyApps are in package format, all R scripts including frontend and backend files were added to the R directory. Wickham (2020) It is not possible to add subfolders to the R directory to create an organized package structure. For reference, all scripts ending in UI control the frontend of the app only. You may assume that all other scripts (excluding *app.R*) control the backend. Finally, the CSS is controlled through *inst/www/style.css* (for standards and outliers) or *inst/www/custom.css* (for dataprep). (*Using Custom CSS in Your App*, 2021) (*Appearance*, 2014)

1.5 Debugging (For ONDRI ShinyApps)

As the ONDRI ShinyApps are in package format, please enable the shiny stack trace by running the following command in your RStudio console once: `options(shiny.fullstacktrace = TRUE)`


This will provide further information on the source and cause of an error (including exact file and line(s) of code) when it occurs. Otherwise, error output will always mention **[No stack trace available]**. An example of the difference in error output when setting the shiny stack trace to false versus true is provided below.



```
Console ~/ONDRI-NIBS/dataprep_app/
> options(shiny.fullstacktrace = FALSE)
> ONDRIDataprepApp::runApp()

Listening on http://127.0.0.1:5473
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Error in [[: subscript out of bounds
[No stack trace available]
```

Figure 4: The error message shown when disabling the option before running the ONDRIDataprepApp package.



```
Console ~/ONDRI-NIBS/dataprep_app/
> options(shiny.fullstacktrace = TRUE)
> ONDRIDataprepApp::runApp()

Listening on http://127.0.0.1:5473
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Factor `VARIABLE` contains implicit NA, consider using `forcats::fct_explicit_na`
warning: Error in [[: subscript out of bounds
86: h
85: .handleSimpleError
84: pivot_wider_spec
83: tidyr::pivot_wider
82: function_list[[i]]
81: freduce
80: _fseq
79: eval
78: eval
77: withVisible
76: %>%
75: observeEventHandler [C:/Users/jahn/documents/ONDRI-NIBS/dataprep_app/R/step4Server.R#44]
74: ..stacktraceon..
73: handlerFunc
72: ..stacktraceon..
```

Figure 5: The error message shown when enabling the option, which shows that the source of the error is line 44 in step4Server.R.

2 Review process

2.1 .gitignore

The .gitignore file (located in the repository root folder) exists as a fail-safe in case any sensitive data is added accidentally to either a GitLab and/or GitHub repository. Any user data in the following formats will not be pushed even if the contents exist in the repository: **.csv, .txt, .rda, .rdata, .rds, .html, .xls, .xlsx, .Rhistory**. For a full list, please take a look at the contents of the .gitignore file.

When a file with an ignored format requires pushing to GitLab and/or GitHub, you should comment out the necessary line(s) in the .gitignore file in the same commit. Immediately following the commit, revert the .gitignore file back by uncommenting those line(s) and committing again to keep the fail-safe intact.

These files conflict with the .gitignore file in their corresponding packages:

- ONDRIDataPrepApp
 - *data-raw/codes.csv*
 - *R/sysdata.rda*
- ONDRISTandardsApp
 - *data-raw/platformspecificSiteCodes.csv*
 - *R/sysdata.rda*

2.2 GitLab to GitHub

When a package is ready for release to GitHub, all code and contents will be copy-pasted from GitLab to GitHub. Additions and/or modifications should never be made to the GitHub repository directly, as the newest release version on GitLab will always overwrite the previous release version on GitHub. Finally, the GitHub commit message should follow the conventions outlined in section 1.2. Official instructions for developers are below:

1. Check the contents of the repository to ensure that no sensitive data was pushed accidentally.
2. Check that the README (and screenshots if applicable) is up to date.
3. For ONDRI ShinyApps only: Check that the version number in the DESCRIPTION file matches the version number in the title header of the user interface.
4. For ONDRI ShinyApps only: Update the corresponding reference guide, if it exists, with the new version number. If the user interface has been modified, please ensure that screenshots are up to date. If core functionality has been revamped, please ensure that any wording is updated.
5. Send a request to Derek requesting approval to push publicly. **Until further notice, Derek will be the only one with permissions to modify the GitHub repository contents directly.**

3 Glossary

GitLab: Version control platform that NIBS utilizes to house software and scripts that are **currently under development**. Access to GitLab repositories is restricted to Brain-CODE users only.

GitHub: Version control platform that NIBS utilizes to house software and scripts that are **ready for release** publicly. GitHub repositories are accessible to all users and can be found at *github.com/ondri-nibs*

Frontend: The client-side of an application, consisting of a user interface (UI) that users interact with directly.

Backend: The server-side of an application, which handles core functionality behind the scenes and allows elements to function.

CSS: Stands for cascading style sheet. It is a file that describes how elements and widgets are displayed on the user interface, and allows for the control of colours, positioning, and overall layout.

More to come.

4 References

Appearance. (2014). <https://rstudio.github.io/shinydashboard/appearance.html>.

Hadley Wickham, J. B. (2019). *Package metadata*. <https://r-pkgs.org/description.html>.

Using custom CSS in your app. (2021). <https://shiny.rstudio.com/articles/css.html>.

Wickham, H. (2020). *Packages*. <https://mastering-shiny.org/scaling-packaging.html>.



This document is licensed under CC BY-SA.