Reliable R Package Documentation

Jonathan Sidi

2018-05-28

$whoami

Jonathan Sidi

- Research Scientist at **Metrum Research Group**
  - Biomedical Modeling and Simulation
- Statistics Graduate Student at HUJI
- Blog: **https://yonicd.netlify.com/**
- Twitter: **@yoniceedee**
- Facebook: hard pass

R Packages

- **What**: Bundled script to reproduce analysis
  - Code
  - Data
  - Documentation
  - Tests
- **Why**: Promotes collaboration across researchers
- **Added Value**: Consistent help documentation format

Portable Structure

- DESCRIPTION : Package Metadata
- NAMESPACE : Manages how the packages interacts with other packages
- R/ : script
- man/ : documentation (Rd files)
- data/ : data
- tests/ : unit tests

{roxygen2}

Popular `R` package that has an easy API to maintain the documentation for each function and the package namespace.

- **You do not have to manually maintain** the Rd files that are needed to populate the `man` subdirectory.
  - The documentation is in the same place as the object that it is describing.
- **You do not have to manually maintain** the NAMESPACE file.
  - Namespacing for the object is compiled as part of the documentation.

High Bars

This is great, but is still a pretty high bar to pass for a non-expert developers.

- You have to understand and track what are the `depends`, `imports` and `suggests` of the package.
  - `@import` and `@importFrom`
- You have to keep a consistent documentation layout across functions.
  - `@params`, `@examples`
- You have to manage links across packages
  - `@seealso`

Juggling Act

For more seasoned package developers this also can be an arduous task.

- Every change to the script
  - e.g. using another package, add a formal argument
- You need to update the {roxygen2} documentation
  - update `@params`, `@imports`, `@importFrom`
- Next level
  - manage quosures with tidyverse packages

{sinew}

- {sinew} is a package that progrmatically populates {roxygen2} fields with information found within the function you are documenting.

- Allowing the developer to focus on the development and not on the continuous managment of the namespacing and mundane manual documentation tasks.
  - CRAN: **https://cran.r-project.org/web/packages/sinew/index.html**
  - Github: **https://www.github.com/metrumresearchgroup/sinew**
  - Gitbook: **https://metrumresearchgroup.github.io/sinew/**

Package Goal

- automate nearly all of the manual tasks needed to document functions
- properly set up the import fields for oxygenation
- make it easier to attain documentation consistency across functions and packages.
- change and append updated parameters, definitions, namespacing to existing documentation

Next Level

When your package imports tidyverse packages it adds a new level of development stress… Managing Quosures

You probably have seen this…

```
* checking R code for possible problems ...
NOTE
xx: no visible binding for global variable
'mpg'
Undefined global functions or variables:
  mpg

R CMD check results
0 errors | 0 warnings | 1 note
```

{tidycheckUsage}

- This small utility package runs the same functions that `devtools::check()` to check for problems in variable usage and returns a `data_frame` containing all the warnings, ready for package introspection.

- Github: **https://www.github.com/yonicd/tidycheckUsage**

  - Proactive methods append {rlang} syntax to resolve the warnings.
  - Great for package development/maintenance and to teach correct syntax usage in small examples.

Workflow Example

- In the following 10 minute example we will create a fully functional `R` package with valid namespacing and documentation that will result in

```
R CMD check --as-cran

0 errors | 0 warnings | 0 notes
```

- Time permitting we will see the interactive capabilities of the packages.

Moving to Rmd….