## Project info

Project title: Fortification of the hyperSpec R Package

Project short title (30 characters): hyperSpec Fortification

URL of project idea page: https://github.com/rstats-gsoc/gsoc2020/wiki/hyperSpec (https://github.com/rstats-gsoc/gsoc2020/wiki/hyperSpec)

## Bio of Student

### Overview

I have always had an interest in technology. Everything from how the technology was designed to interact with users, the manufacturing innovations that the technology requires to reach multiple users, and the physical form a product takes over generations have always fascinated me. Now, as an electrical engineering and computer science (EECS) student I continue to develop my passion for applying mathematical, computational, and digital tools from EECS to the study and augmentation of biological, cognitive, cultural, and electro-mechanical systems. Concretely, my scientific background is rooted in my research experiences: employing cognitive and software-defined radios for mobile spectrum sensing, modeling the impact of media engagement on infectious diseases, utilizing sentiment analysis and physiological sensing (e.g.skin conductance, facial electromyography) for storytelling and cognition, and the design of speech recognition systems as a representation of early language acquisition. Unifying these diverse research interests and experiences is Systems theory – the interdisciplinary study of natural and man-made systems. Ultimately, through the application of systems theory (systems thinking), my research aims to reveal and understand the broad complex communication paradigms within and between animal, machine, plant, and environment

### Qualifying Background

My education as an undergraduate EECS student has provided me with foundations in programming, data structures, and software engineering, all of which are critical to successful open source development. Beyond the classroom, I have also been fortunate enough to undertake numerous research experiences and internships. For example, this past summer I was a research intern at Harvard University's Long-Term Ecological Research site (LTER), Harvard-Forest (HF). At HF I was working under the Data Provenance team where I had the opportunity to be mentored by computer science and ecology professors and graduate students in basic R software development. During my time as an intern, I wrote a proposal for a new R package called provBookR, collected informal user requirements from HF scientists and students. From there, I processed these user requirements to generate use cases and flow diagrams (a common practice in software engineering and product design). After generating user requirements I spent the rest of the summer developing the provBookR prototype and learning about the currently available R data provenance tools and packages created by the HF Data Provenance team. In the end, this experience was extremely valuable because it introduced me to data provenance, open-source development (working with in-person and remote teams), R package development, and being comfortable with going through code repositories and documentation until my eyes fall out. Additionally, during my other internships (i.e., Santa Fe Institute, KU Ecology Department) I primarily used R for data analysis and synthesis. Hence, I believe I'm qualified to learn and contribute to the hyperSpec project for GSoC 2020.

## Contact Information

Student name: Erick Oduniyi

Student postal address: 15502 E Lynnwood St, Wichita, KS 67230

Telephone(s): +1 316-990-1410

Email(s): eeoduniyi@gmail.com

Other communications channels:

- Skype: Erick Oduniyi (+1 316-990-1410)
- Hangouts: eeoduniyi@gmail.com

## Student affiliation

Institution: Wichita State University

Program:

- Degree: Undergraduate
- College: College of Engineering
- Major: Computer Engineering

Stage of completion: Senior Undergraduate

Contact to verify: Neal Hoelting

- Email: neal.hoelting@wichita.edu
- Phone: +1 316-669-3580

## Schedule Conflicts:

If accepted, I will have no other schedule conflicts and will treat the R GSoC project as a full time (40+hrs/week) job this summer.

## Mentors

Evaluating mentor name and email: Bryan Hanson (hanson@depauw.edu)

Co-mentor name(s) and email(s): Claudia Beleites (claudia.beleites@chemometrix.gmbh)

I have been in touch with Roman Kiselev and Claudia Beleites starting March, 24, 2020 through github.

# Coding Plan and Methods

## Project Background

The hyperSpec (http://hyperspec.r-forge.r-project.org/) package allows R users a suite of utilities for manipulating spectroscopic data. These utilities currently include functions for importing spectroscopic data (e.g., Raman, IR, NIR, UV/VIS), plotting, and preprocessing and wrangling spectroscopic objects (i.e., hyperSpec objects) so that they can be easily integrated with commonly used statistical analysis packages (e.g., **plsr** and **MASS**). On the one hand, the usefulness of the hyperSpec package for spectroscopic research is clear and immediately useful to R programmers and practitioners of bioinformatics, chemoinformatics, and medical statistics. On the other hand, the infrastructure afforded by hyperSpec has led to various dependencies on other R packages, large amounts of test data, and more than a hundred .R files. Accordingly, the current state of the package has become difficult to maintain and vulnerable to fixes related to dependencies on deprecated packages. Thus, the project **Fortification of the hyperSpec R Package** aims to tackle these issues.

## Project Goal

At a high-level, there are three goals of the proposed project: ***distilling*** *the original hyperSpec package,* ***shielding*** *against package dependencies, and* ***bridging*** *hyperSpec with other relevant packages*. Accomplishing each of these goals will result in the fortification of hyperSpec's software architecture and the creation of a new ecosystem of R spectroscopic tools.

#### Goal 1: Distilling hyperSpec

The first goal works to reduce the package footprint by distilling and retaining core computationally inexpensive functionality within the original hyperSpec package and outsourcing computationally expensive functionality (i.e., file import functions) across smaller and specialized packages. The result of this goal will make the original and subsequent hyperSpec packages easier to maintain.

#### Goal 2: Shielding hyperSpec

The second goal works to protect the package by striving for a balanced approach to package dependencies. This goal is less about making sure hyperSpec is completely decoupled from other packages and more about ensuring the features and procedures the hyperSpec package depends are actively maintained on CRAN and successful non-CRAN packages that themselves are reliable. For instance, if hyperSpec depends on two packages that provide the same functionality, then hyperSpec should only rely one of those packages (assuming there is not a straightforward base R solution); ideally the package that itself has fewer dependencies and/or has a longer life history on CRAN (i.e., fewer updates) and success within the R developer community. Because CRAN package errors are commonly the result of updates in package dependencies, the result of this goal will bring control back to hyperSpec making it anti-fragile.

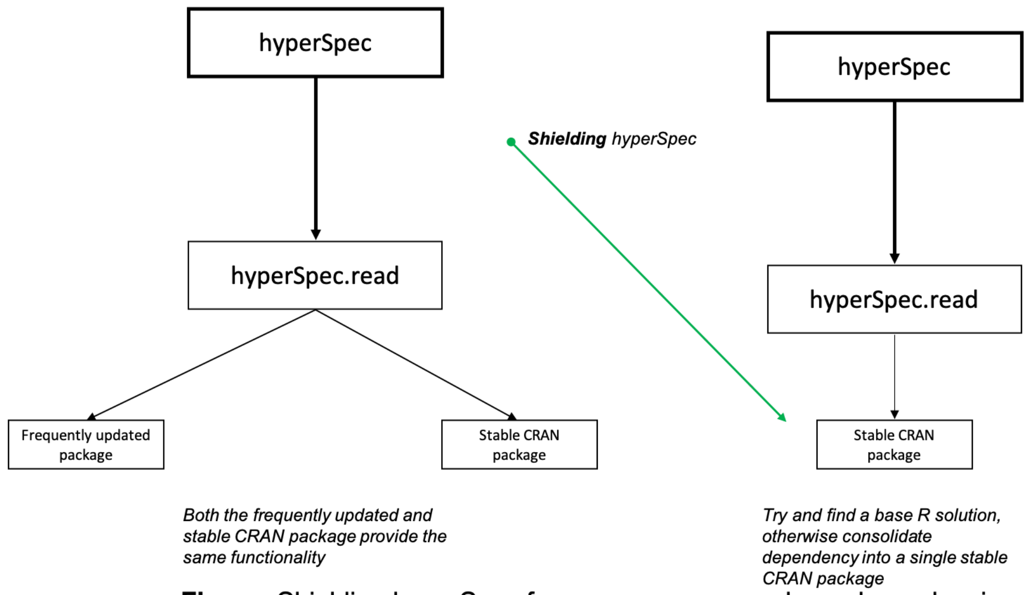

hyperSpec
hyperSpec.read
Frequently updated package
Stable CRAN package
Both the frequently updated and stable CRAN package provide the same functionality
***Shielding*** *hyperSpec*
hyperSpec
hyperSpec.read
Stable CRAN package
Try and find a base R solution, otherwise consolidate dependency into a single stable CRAN package

**Figure:** Shielding hyperSpec from unnecessary package dependencies

**Goal 3: Bridging hyperSpec**

The third goal works to build packages that make hyperSpec and hyperSpec objects easier to integrate with relevant data cleaning, spectroscopic, and statistical analysis packages. For example, integrating hyperSpec with spectra preprocessing packages like **baseline** and **EMC**, data wrangling and plotting packages like **dplyr** and **magrittr**, and **ggplot2**; file import functions like the **readJDX** package. The result of this goal will generate several small hyperSpec companion packages (e.g., hyperSpec.read, hyperSpec.tidyverse, etc.).

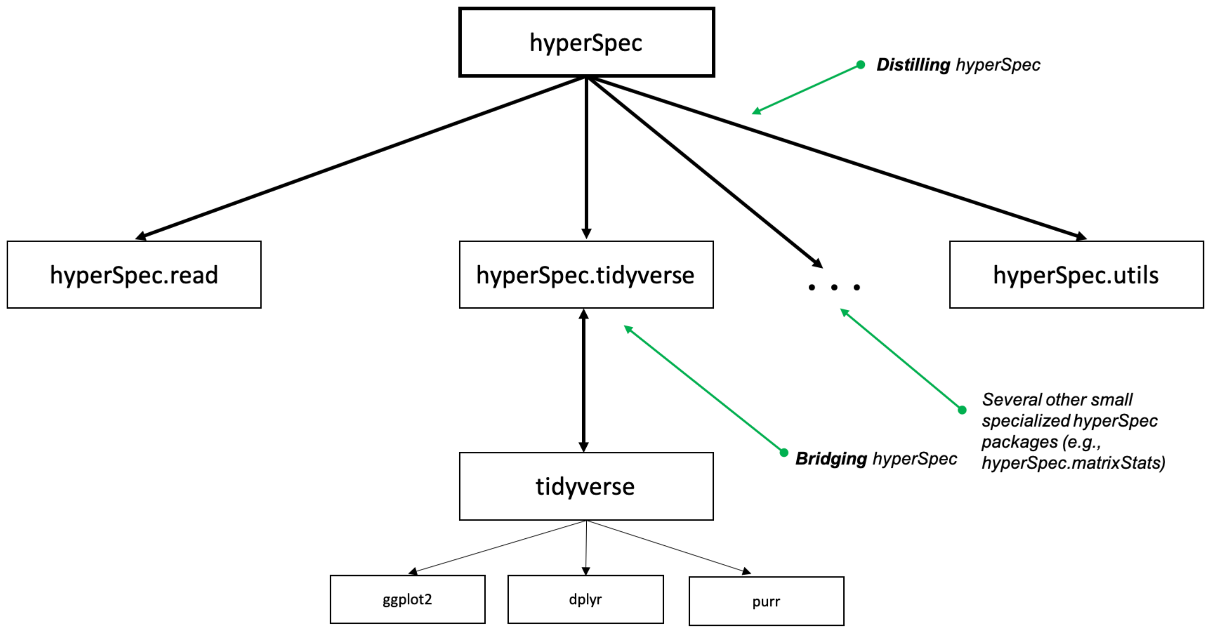

hyperSpec
***Distilling*** *hyperSpec*
hyperSpec.read
hyperSpec.tidyverse
. . .
hyperSpec.utils
Several other small specialized hyperSpec packages (e.g., hyperSpec.matrixStats)
***Bridging*** *hyperSpec*
tidyverse
ggplot2
dplyr
purr

**Figure:** Software fortification of the hyperSpec R package

## Project Planning

Whether developing big or small software projects, robust software development is no easy task. Software development requires extensive design and planning, clear implementation, verification, and testing; all of which should be done iteratively. In fact, software projects often require a team of developers that span these tasks: designing, implementing, reviewing, testing, and documenting. However, because of the nature of open-source projects one, more, or all of these tasks can fall on the shoulders of a single developer. As a result, the coverage of these tasks can vary widely. The beauty of open source is that through engagement and strong communication, solo endeavors turn into community projects that support contributors from around the world that work together to develop robust software systems. I believe at the heart of successful communities is planning and clear communication, the guiding principles of my coding project.

## Coding Project

From the three high-level goals, it is clear that my coding project is package fortification and lean package development. A plan that proposes to execute this coding project and as a result the goals of the project is the following development cycle: generate software description, function design, function implementation, quality plan, test plan, documentation plan, deliverables:

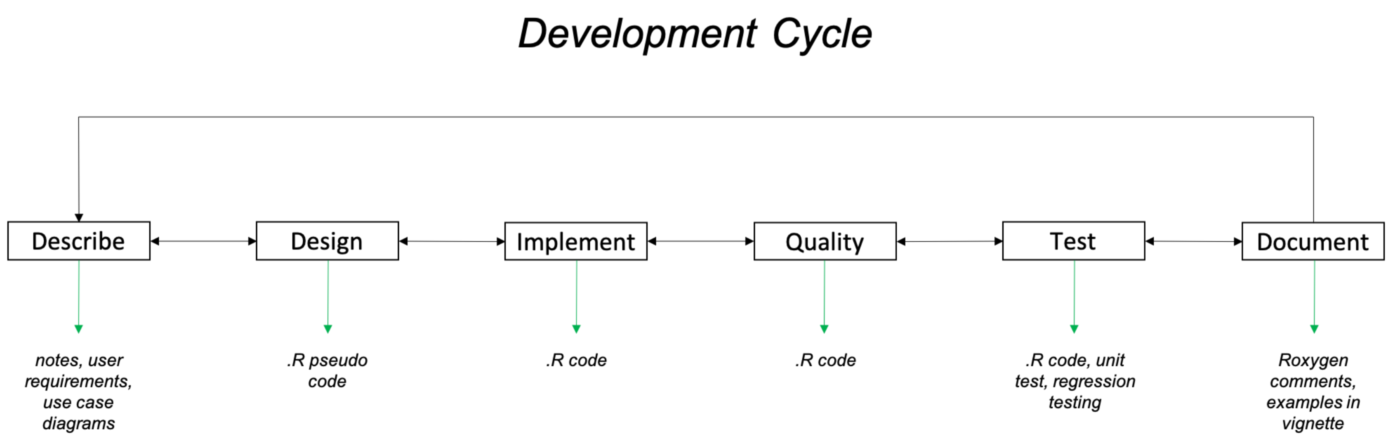
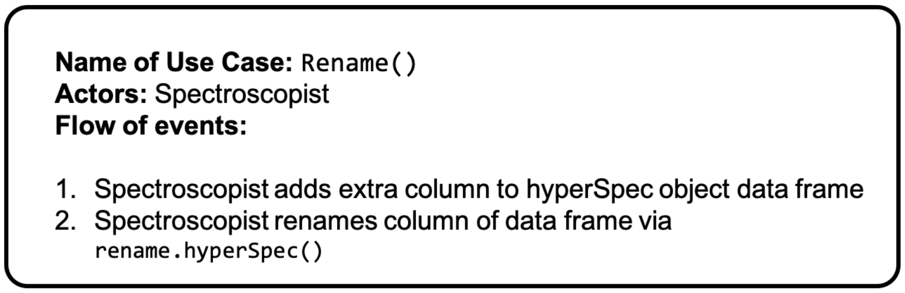

Development Cycle
Describe
Design
Implement
Quality
Test
Document
notes, user requirements, use case diagrams
.R pseudo code
.R code
.R code
.R code, unit test, regression testing
Roxygen comments, examples in vignette

**Figure:** Software development plan for making progress on coding project

## Software Description

I will always first "sketch" out the description of the task that need completed as specified and agreed upon with my mentors. I expect these tasks to often come in the form of GitHub issues or through other communication channels. Depending on the complexity of and formality of the task at hand, these description sketches will take the form of briefly typed notes or more formal mock-ups of user requirements and use case diagrams.



**Name of Use Case:** `Rename()`
**Actors:** Spectroscopist
**Flow of events:**

1. Spectroscopist adds extra column to hyperSpec object data frame
2. Spectroscopist renames column of data frame via `rename.hyperSpec()`

**Figure:** Example use case diagram from dplyr::rename()

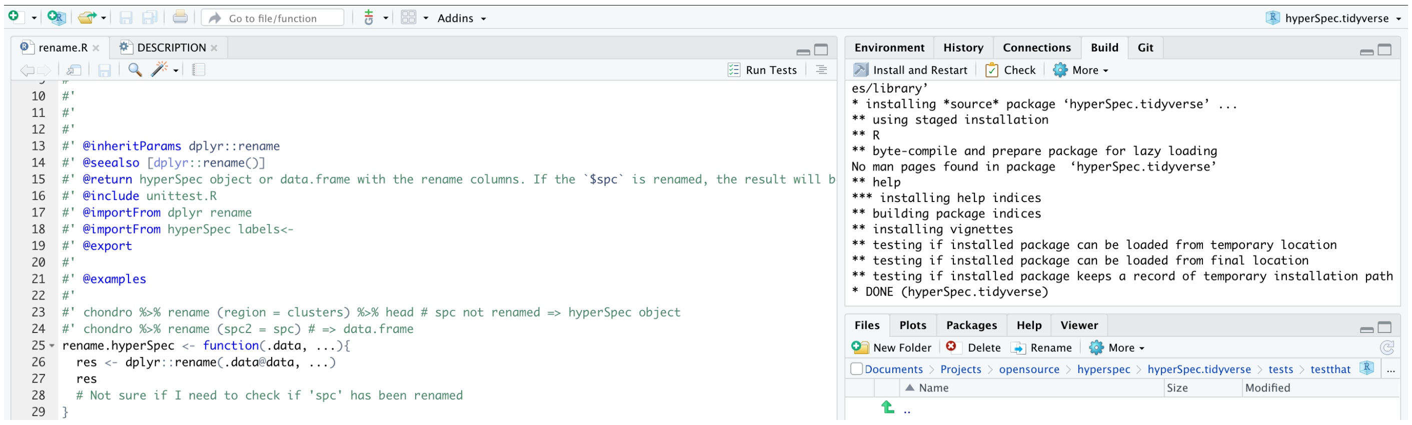
## Function Design

Depending on the complexity of the software description, I will develop R pseudo-code specifying function assumptions, inputs, outputs, and side effects. I find it helpful to generate software descriptions and function designs in Rmarkdown first.

```
rename.hyperSpec <- function(.data){
  res <- rename(.data)
  return{res}
}
```

## Function Implementation

For function implementation I will follow Hadley Wickham's R code workflow as discussed in his [R packages book (http://r-pkgs.had.co.nz/r.html)](http://r-pkgs.had.co.nz/r.html): create .R file giving it an appropriate and clear name, explore the code in the console, "rinse and repeat" until the function is implemented as specified.



```
#'
#'
#'
#' @inheritParams dplyr::rename
#' @seealso [dplyr::rename()]
#' @return hyperSpec object or data.frame with the rename columns. If the `$spc` is renamed, the result will b
#' @include unittest.R
#' @importFrom dplyr rename
#' @importFrom hyperSpec labels<-
#' @export
#'
#' @examples
#'
#' chondro %>% rename (region = clusters) %>% head # spc not renamed => hyperSpec object
#' chondro %>% rename (spc2 = spc) # => data.frame
rename.hyperSpec <- function(.data, ...){
  res <- dplyr::rename(.data@data, ...)
  res
  # Not sure if I need to check if 'spc' has been renamed
}
```

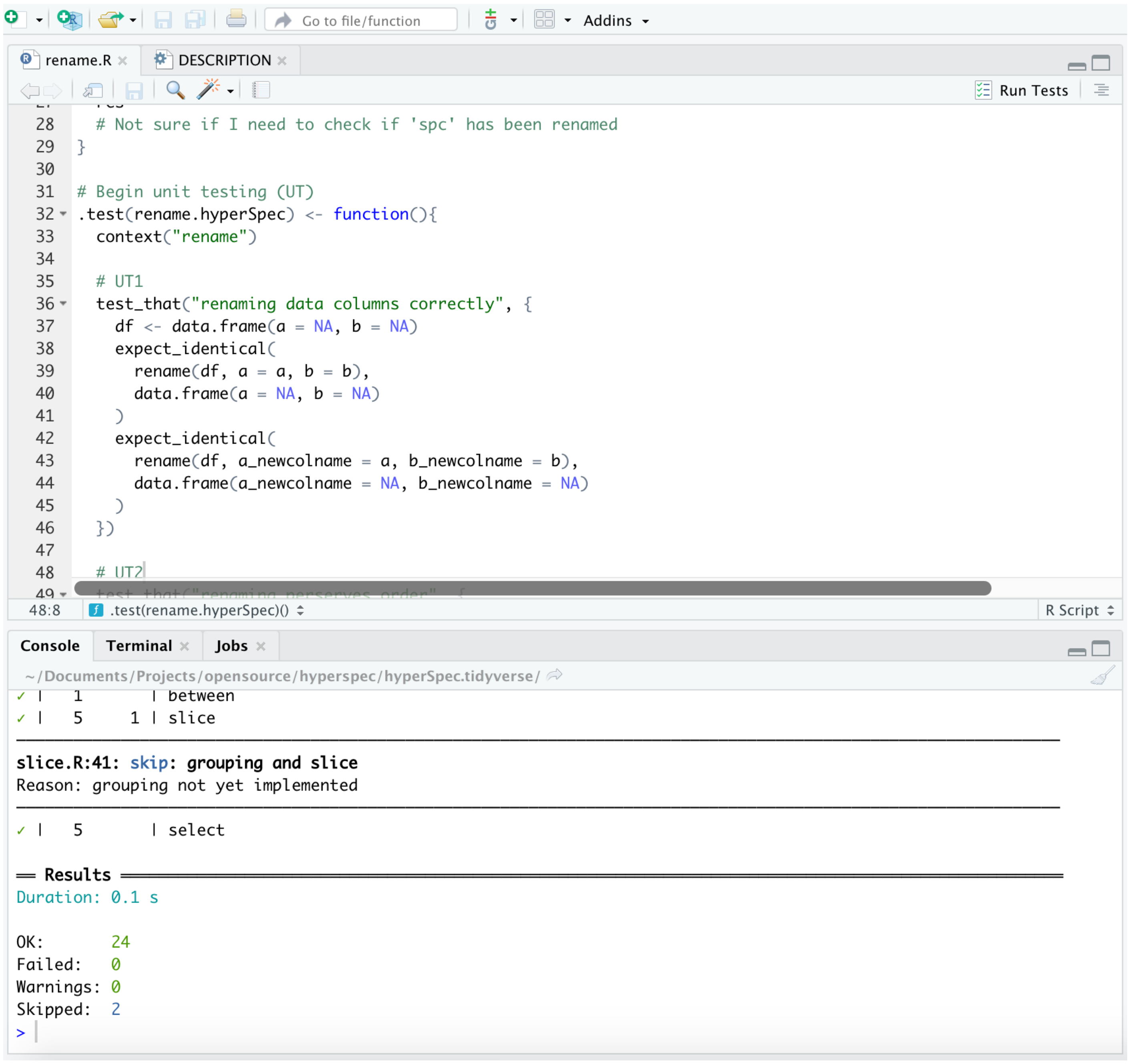
```
es/library’
* installing *source* package ‘hyperSpec.tidyverse’ ...
** using staged installation
** R
** byte-compile and prepare package for lazy loading
No man pages found in package  ‘hyperSpec.tidyverse’
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* DONE (hyperSpec.tidyverse)
```

## Quality Plan

I will write anti-fragile code that can be refactored, extended, and tested easily. At a high-level, this means the code will not be unnecessarily clever or confusing to my mentors, other students, and especially my future self. Any code that is written will adhere to the style/standard of the hyperSpec code base.

## Testing and Verification Plan

For each function and its associated code, I will make sure to develop a comprehensive unit test. Depending on the complexity of the function (simple functions versus functions with side effects like file import/reading functions) prepare tests that checks for points of failure and potentials bugs.



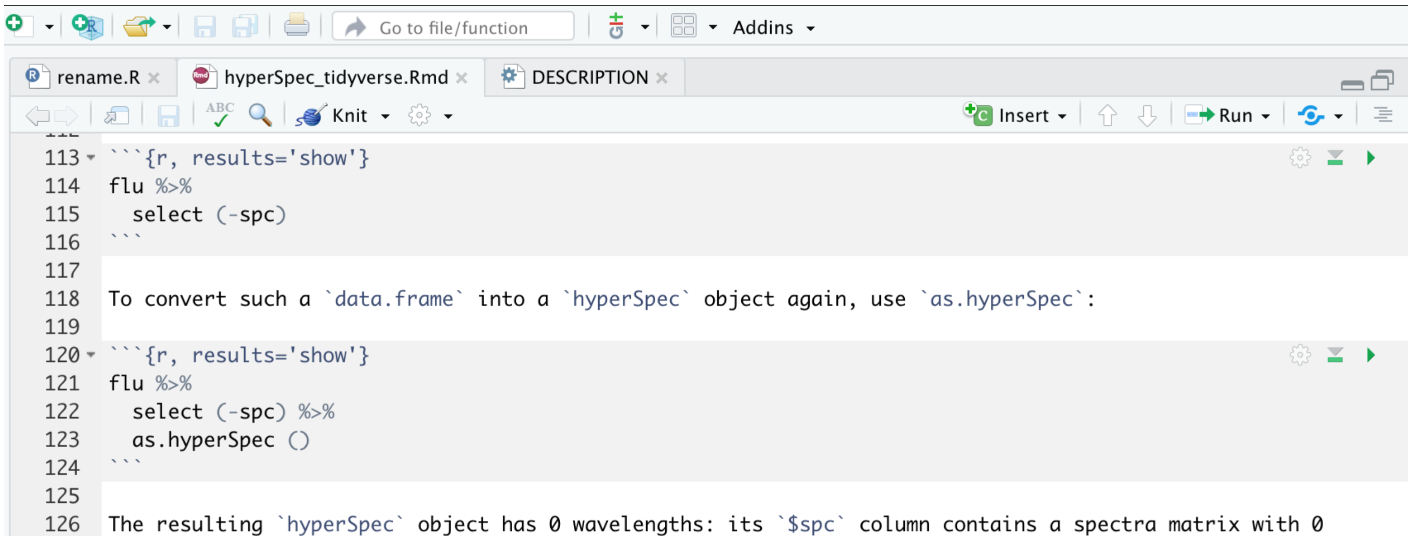
```
  # Not sure if I need to check if 'spc' has been renamed
}

# Begin unit testing (UT)
.test(rename.hyperSpec) <- function(){
  context("rename")

  # UT1
  test_that("renaming data columns correctly", {
    df <- data.frame(a = NA, b = NA)
    expect_identical(
      rename(df, a = a, b = b),
      data.frame(a = NA, b = NA)
    )
    expect_identical(
      rename(df, a_newcolname = a, b_newcolname = b),
      data.frame(a_newcolname = NA, b_newcolname = NA)
    )
  })

  # UT2
  test_that("renaming perserves order", {
```

48:8
.test(rename.hyperSpec)()
R Script
Console
Terminal
Jobs
~/Documents/Projects/opensource/hyperspec/hyperSpec.tidyverse/

```
✓ |   1       | between
✓ |   5     1 | slice
────────────────────────────────────────────────────────────────────────────────
slice.R:41: skip: grouping and slice
Reason: grouping not yet implemented
────────────────────────────────────────────────────────────────────────────────

✓ |   5       | select

══ Results ═════════════════════════════════════════════════════════════════════
Duration: 0.1 s

OK:       24
Failed:   0
Warnings: 0
Skipped:  2
>
```

### Documentation Plan

I will make sure each .R file is appropriately documented (i.e., parameters, examples, etc.) using roxygen. I will also make sure to comment sections of the code that highlight assumptions or any outstanding nuances. Finally, for each function and feature make sure there is a brief explanation on how to use it in the package vignette.



~~~
```{r, results='show'}
flu %>%
  select (-spc)
```

To convert such a `data.frame` into a `hyperSpec` object again, use `as.hyperSpec`:

```{r, results='show'}
flu %>%
  select (-spc) %>%
  as.hyperSpec ()
```

The resulting `hyperSpec` object has 0 wavelengths: its `$spc` column contains a spectra matrix with 0
~~~

**Deliverables**

The highest level deliverable is a well documented and tested R package that distills, shields, and bridges hyperSpec with relevant R packages and tools. Of course, this high-level deliverable depends on well tested and documented functions and code.

```
chondro %>% rename(region = clusters)  # YES!

# as opposed to
chondro$region <- chondro$clusters
chondro$region <- NULL
```

## Project Obstacles

Unfortunately, there is no such thing as perfect and no plan is ever problem-free, so I foresee a couple of obstacles: misunderstanding of software descriptions and deliverables from mentors, lack of time to implement deliverables or spending too much time on a deliverable, and forgetting about unresolved errors, maintaining consistent code quality. To overcome these obstacles I will always keep mentors up to date with progress and set expectations early and adjust them accordingly. In particular, my expectations for a successful R GSoC experience consist of keeping myself enthusiastic and accountable for completing work and associated deliverables throughout the summer, as well as, considerate of my mentors, other participating students time. Finally, I expect myself to be appropriately open and remain cooperative across disciplines and people in addition to actively listening to concerns, ideas, and issues, so that cross-pollinating conversations are encouraged, and assumptions, biases, and intentions are not misplaced.

## Timeline

### Brief Timeline

- March 31 - May 4: Pre-GSoC Period
- May 5 - May 31: Community Bonding Period
- June 1 - June 28: Coding Period 1
- June 29 - July 3: Phase 1 Evaluations
- July 4 - July 26: Coding Period 2
- July 27 - July 31: Phase 2 Evaluations
- August 1 - August 31: Coding Period 3
- August 31 - September 7: Final Evaluations
- September 8: Final Results

### Detailed Timeline

**March 31 - May 4: Pre-GSoC Period**

**Week 1** Continue to spend time understanding the codebase and solving open issues. Start reading all R documentation related to hyperSpec (https://cran.r-project.org/web/packages/hyperSpec/vignettes/hyperspec.pdf). *(Make sure I have all of the development tools I need to contribute efficiently and effectively)*

**Week 2** Continue to spend time understanding the codebase and solving any open issues. Continue reading documentation related to hyperSpec. Start reading documentation on other spectroscopy packages (https://bryanhanson.github.io/FOSS4Spectroscopy/)

**Week 3** Continue to spend time understanding the codebase and solving any open issues. Continue reading documentation related to hyperSpec. Start reading the documentation on other spectroscopy packages. Start reading the documentation on spectra preprocessing packages (baseline and EMSC) packages.

**Week 4** Continue to spend time understanding the codebase and solving any open issues. Continue reading documentation related to hyperSpec. Continue reading the documentation on other spectroscopy packages. Start reading the documentation on data wrangling and visualization (ggplot2/tidyverse).

**Week 5** Continue to spend time understanding the codebase and solving any open issues. Continue reading documentation related to hyperSpec. Continue reading the documentation on other spectroscopy packages. Start reading the documentation on packages and tools related to the importing of spectroscopy data (readJDX).

**Week 6** Brush up on R programming. Find good resources for learning base R and advance R techniques. Brush up on git and git-lfs.

**Week 7** Brush up on R programming. Find good resources for learning base R and advance R techniques. Brush up on git and git-lfs. *(Make sure I have all of the development tools I need to contribute efficiently and effectively)*

**May 5 - May 31: Community Bonding Period**

**Week 1** Create a clear communication schedule for weekly meetings, protocols for check-ins, and a high-level roadmap for successful contribution to the hyperSpec project (e.g., forking, merging, pulling, etc.).

**Week 2** Finalize development tools and workflow for contributing to the project over the summer. Inquiry about helpful R programming and spectroscopy resources from mentors and digest them.

**Week 3** Continue reading resources from my mentors. Start working on software development descriptions for goals, proposed features, functions, and other tasks related to development.

**Week 4** Continue reading resources from my mentor. Start working on software development descriptions for goals, proposed features, functions, and other tasks related to development.

**June 1 - June 28: Coding Period 1**

**Week 1** Let the coding begin! Start making progress on Goal 1: Stick to the development cycle for all weekly tasks (i.e., describe, design, implement, quality check, test, document, and iterate). Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 2** Let the coding continue! Continue making progress on Goal 1. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 3** Let the coding continue! Continue making progress on Goal 1. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 4** Let the coding continue! Wrap up progress on Goal 1. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**June 29 - July 3: Phase 1 Evaluations**

This period will be used to write a detailed report on the work done in Coding Period 1. All the work done will be uploaded and documentation will be created/uploaded to hyperSpec and associated packages.

***Deliverables:***

- Distilled hyperSpec package
- New specialized hyperSpec packages for file I/O
- New implemented import filters for new file formats

**July 4 - July 26: Coding Period 2**

**Week 1** Let the coding continue! Start making progress on Goals 2 and 3: Stick to the development cycle for all weekly tasks. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 2** Let the coding continue! Continue making progress on Goals 2 and 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 3** Let the coding continue! Continue making progress on Goals 2 and 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 4** Let the coding continue! Continue making progress on Goals 2 and 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**July 27 - July 31: Phase 2 Evaluations**

This period will be used to write a detailed report on the work done in Coding Period 2. All the work done will be uploaded and documentation will be created/uploaded to hyperSpec and associated packages.

***Deliverables:***

- Shielded hyperSpec and associated hyperSpec packages
- Fortify hyperSpec for tidyverse with bridge packages

**August 1 - August 31: Coding Period 3**

**Week 1** Let the coding continue! Start making progress on Goal 3: Stick to the development cycle for all weekly tasks. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary and leave enough time for)*

**Week 2** Let the coding continue! Continue making progress on Goal 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 3** Let the coding continue! Continue making progress on Goal 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**Week 4** Let the coding continue! Wrap up progress on Goal 3. Compile a weekly report of progress made. Meet with mentor(s). *(check-in with mentors as necessary)*

**August 31 - September 7: Final Evaluations**

This period will be used to write a detailed report on the work done in Coding Period 3. All the work done will be uploaded and documentation will be created/uploaded to hyperSpec and associated packages.

***Deliverables:***

- Fortify hyperSpec for baseline with bridge packages
- Fortify hyperSpec for EMSC with bridge packages
- Fortify hyperSpec for matrixStats with bridge packages

**September 8: Final Results**

All documentation, modules, and tests will be uploaded and Travis CI will be integrated into the project Github page. All the deliverables promised for R GSoC 2020 will be provided by this stage.

## Contingency Plan

Again, there is no such thing as perfect and no plan is ever problem-free. Also, life doesn't wait for coding projects or internships to finish. So, if there is a decline in health or other personal and family issues arise, I will take a deep breath and let mentors know what's going on without TMI. If I foresee a setback in development make sure I'm communicating my difficulties and get suggestions on how to pivot from mentors so that milestones and deliverables can still be met in a timely manner.

### Additional Information

There are currently no other summer commitments that interfere with R GSoC 2020, so I expect to be working 40+hrs per week. Additionally, I will be using the 7 weeks before the coding period to get a running start with development and consistent communication between mentors.

## Management of Coding Project

In addition to writing weekly reports, I will maintain a blog that goes over my development progress to ensure I'm following my schedule and getting feedback from my mentors. I will be committing at least once a day and any change in that behavior will be expressed to my mentors beforehand. Testing will be done throughout the summer and during the summer I will be working on setting up Travis for regression testing.

## Test

To get prepared for the fortification hyperSpec project I have worked on tests/tasks that are easy, medium, hard, and very hard as outlined by my mentors. These tests range from installing the hyperSpec packages to creating GitHub repositories and related skeleton hyperSpec packages. I have submitted solutions for the easy and medium tasks and will continue to work on the hard and very hard task and developing familiarity with the hyperSpec team and codebase.

While at Harvard Forest I worked on solving similar coding problems as part of an onboarding process. A document that details these problems and output can be viewed here (https://storage.googleapis.com/root-proposal-1246/Harvard-Forest-REU2019/REU-onboarding-Clean.html).

## Anything Else

### Working Environment And Schedule

I will be mostly working full-time on the development cycle (i.e., describe, design, implement, quality check, test, document, and iterate) Monday to Saturday. My timezone follows Central Daylight Time and will make sure to be aware of my mentor's timezone. I typically like to reserve Sundays for periods of rest. Additionally, If I plan on traveling during the summer I will make sure my mentors are aware of such plans.

### Communication

I am very flexible with my schedule and can adjust accordingly. I do have a habit of working late at night and early in the morning. As for the channels of communication, I'm comfortable with any form of communication that suits my mentor.