

Kaiaulu Final Presentation

By: Ruben Jacobo, Waylon Ho





Table of contents

01

Introduction

02

Problem Statement

03

Major Accomplishments

04

Project Management and
Delivery

05

Reflecting

06

Next Steps





01

Introduction





Kaiaulu Refresher

- Open Source Package designed to provide a means for analyzing a variety of software development stacks (git log, mailing list, issue trackers, source code, etc.)
- Kaiaulu API returns tables, which allows for simple data exploration and integration from various sources.
- Project configuration files are used to store project-specific information for reproducibility.
- Configuration files are decoupled from R Notebooks, allowing users to alter parameters and inspect output.



Problem Statement & Solution

- Kaiaulu's codebase has many different parts and functions
 - Not very "user-friendly", requires familiarity with Kaiaulu to use efficiently
 - Old code is dependent on outdated software that could change
- Technical Solution
 - Augment Kaiaulu to have more functionality, easier to understand code interface, improve overall usability of tool for future users
 - Refactor older code into more flexible functions for more customization and testing



Major Accomplishments



Experience Report + Git Fake Data

- Implement git functions that allow us to customize fake git data



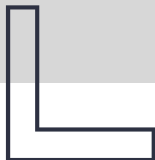
Jira and Mbox Fake Data generators

- Jira and Mbox fake data generator functions and testing.



R Cheat Sheet

- Created a one page documentation visual of all functions



Mbox Fake Data Generator

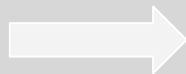
Objective: Develop fake data generation functions for mailbox analysis.

make_mbox_reply:

Purpose: Generates a single entry in the mailbox.

Input: Recipient information, Sender information, email body, time, etc

Output: A single reply entry for the mailbox in proper mbox format



make_mbox:

Purpose: Compiles multiple replies into one mailbox.




Input: Accepts any number of replies generated by make_mbox_reply.

Output: A compiled mbox ready for analysis.





Mbox Fake Data Continued

Example Data Creation:

- 
- 
- 
- Utilized the functions to create example data
 - Demonstrated the flexibility of the functions by customizing content and structure.

Unit Testing:

- 
- Implemented unit tests for the existing `parse_mbox()` function.
 - Ensured that `parse_mbox()` accurately processes the fake data generated by `make_mbox`.
- 

Git Fake Data Generator



- Finalized the fake data generator for Git infrastructure. This includes functions like `git_init()`, `git_commit()`, `git_add()` and more to fully encapsulate the functionality of a real git repository
- Allows for full customization of git data that can be used for testing in the next steps
- Used those functions to create example git repos and then tested their expected output using third party tools like perceval

Files Changed



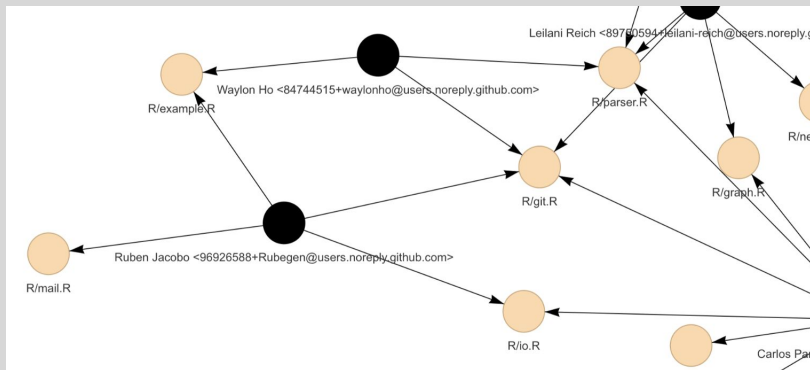
Example.R

- Use git log functions we parameterized to create example data “scenarios” that would help us create our unit tests



TestParser.R

- Write our unit tests in TestParser.R
- Create Unit tests that can test the output of an existing function- `parse_gitlog()` using our example fake data





Jira Fake Data Generator

- Similar to Mbox functions, adapted to create JIRA issues and issue comments
- Able to mimic realistic JIRA data from user customized fields
 - very useful for testing parse functions
 - tests indicate if changes to code lead to unexpected results
- Modular function design includes create_base_info, create_ext_info, create_comments -> make_jira_issue, make_jira_issue_comments
- Consolidates previous multi function generation into one single neat interface with clear parameters the user can understand
- Contributes to larger goal of making Kaiaulu more flexible, user friendly, and continuable in the future (more parameters)

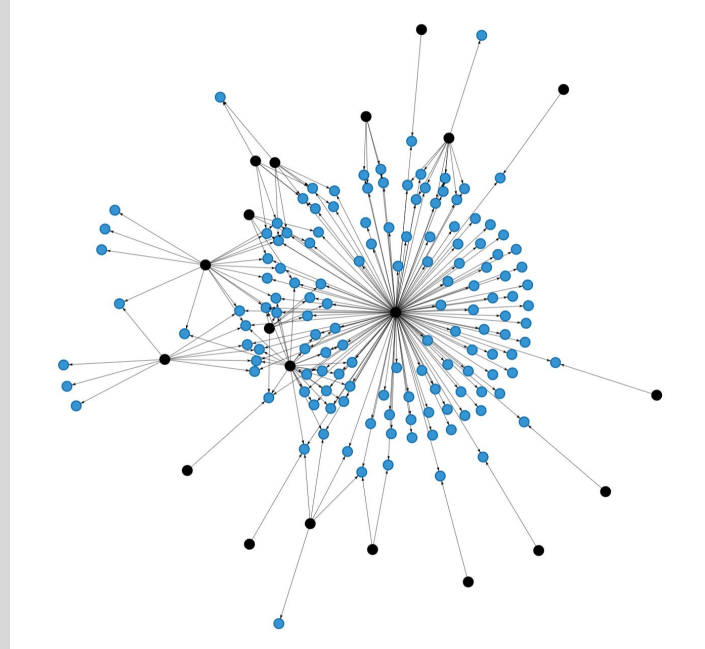


Jira Software



Project Management and Delivery

- Sponsor created issues through github with an extensive explanation of the problem and we were given creative freedom to solve the problem.
- We first laid out the groundwork of the code we planned to add, then received approval from Carlos to proceed with the coding process.
- Found that committing code and getting regular feedback from our sponsor was the key to our success in Milestone 2.



Reflections

- Milestone 2 went smoothly when we adapted to the codebase more and to RStudio and its capabilities.
- Technical problems with installing third party software
- Communication could have been improved
 - more concrete deadlines and frequent updates with our sponsor
 - constant updates and iterating back and forth achieved much more progress towards the end
- Able to clarify effectively with our sponsor about the exact solution he wants of us.



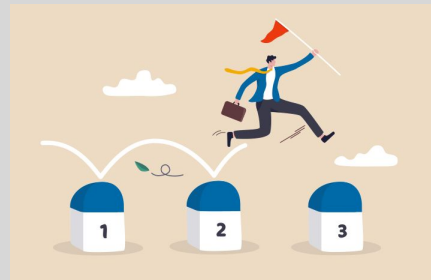


06

Next Steps



Next Steps



- Create more example data with our functions
- Add more Unit testing to our Jira and Mbox fake data generator functions as well as the associated parser functions
- Add more Fake data generators from different sources (ex: bugzilla issues)
- Finalize cheat sheet and other documentation for future developers to continue improving Kaiaulu



Acknowledgements

We'd like to thank our sponsors:

Carlos Paradis and Rick Kazman

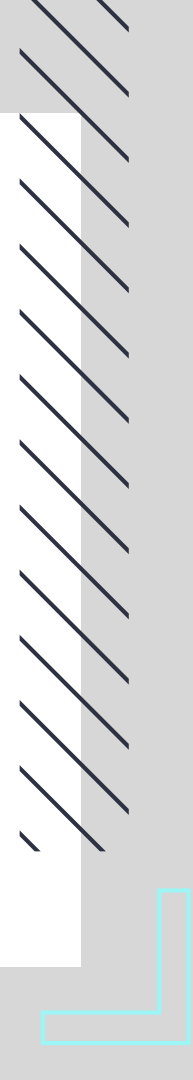
for the opportunity to contribute to Kaiaulu and for their guidance and patience throughout the semester.



A decorative vertical bar on the left side of the slide, featuring a series of light blue plus signs arranged in a grid pattern.

Thank you!

CREDITS: This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#), and infographics & images by [Freepik](#)

A decorative vertical bar on the right side of the slide, featuring a series of dark blue diagonal lines and a light blue L-shaped corner bracket at the bottom right.

Presentation Deliverables Here:

https://docs.google.com/document/d/1X6rlt4Wj-04mqMSyvDpYm49KelEMN4iVZky5_5q9oZE/edit

At a minimum, the presentation should contain the following.

Problem Statement

Technical Solution Overview

Major Accomplishments

Also, include requirements that were not implemented or partially implemented and the rationale behind missing/partial implementation

If the sponsor approves, include screenshots.

Include brief details about the technical implementation

Client feedback on delivered/implemented features/modules

Quality Assurance Metrics (Overall)

Defects reported, fixed, open, etc.

Project Management and Delivery

An overview of the software process the team followed

Reflections

What went right?

What went wrong?

What could have been done differently?

Includes technical and soft skills individuals in the team acquired or improved upon

Next Steps

What are the next steps for the client?

Acknowledgments

