Contact Information

Name: Jiayi Guo

Country: United States

Email: jiayiguo2002@gmail.com

Which method of communication do you prefer? (i.e. in person, email, chat, video conference, etc.): email or zoom

What processing foundation project(s) are you interested in?

grepreaper R Package Development.

About You

1. **If you have a link to a resume/CV/LinkedIn profile, include it here.**
   https://www.linkedin.com/in/jiayi-guo-08536a359?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=ios_app
2. **Please describe yourself, including your development background and specific expertise.**
   I am a graduate student at ucla and my current field of study is data science, but I don't want to limit myself to the study of data science and would like to explore more possibilities and become a developer!
3. **What do you hope to gain through the process of participating in GSoC, and specifically by contributing to a processing foundation project?**
   I wanted to be involved in an open source project with someone to mentor me, which let me know that I could realistically contribute something to the world. School programs are always limited to demo examples, leaving a lack of practicality.
4. **Why are you interested in the processing foundation coding project(s) you stated above?**
   To be honest, the project's description page captivated me, and I could tell from the statements that this is an open source community with passion and inclusiveness. Its philosophy also appeals to me, telling me what I can do for the world with these programming skills, a little something practical and helpful, and completely led and realized by me. A predictable outcome gives hope and motivation to strive.
5. **Have you reviewed the important dates and times for GSoC?**
   Yes of course! I am well aware that the submission is on April 8, I will get the acceptance result on May 8, and most importantly, I think the work should start from the moment I get the result, not June 2 as written in the timeline. It's not that I don't have confidence in myself and feel that I can't complete it within the expected time, but I think that the more prepared I am the better result I am likely to get.

6. **Do you have any significant conflicts with the listed schedule? If so, please list them here.**
   I'm going to have a busy time from June 6 to June 13, covering our last student's capstone and my own final week to cope with.
7. **Do you understand this is a serious commitment, equivalent to a full-time paid summer internship or summer job?**
   Yes, I totally agree, this job is expected to take up 35-40 hours of my time per week, and it's a fully focused and effective work schedule, and I don't find it any easier than a summer internship where I sit in an office, so I'm well prepared to deal with it as well. If I'm selected for this program, I'll be staying home for the summer and not going anywhere until I'm sure I can finish it on time!
8. **We strongly prefer students that provide code samples, ideally in a "social coding" site like GitHub. Please provide a link to code you've written, whether it's a zip file / tarball you host on your own or your Github profile.**
   https://github.com/jiayiguo0801

Proposal

Overview

Provide a brief overview of your proposed solution and objectives you want to achieve with your chosen project.

Overview: The grepreaper R package is designed to significantly enhance R's file reading capabilities by integrating the power of command-line grep with R's native tools. Many users of R rely on basic file reading capabilities, but there are numerous advanced functionalities that can be unlocked by linking file reading to grep commands. These features include counting records before reading data, extracting only records that match a specified pattern, and reading and aggregating data from multiple files simultaneously. By using fread() from the data.table package, this package will give users the ability to run custom grep commands to facilitate pattern matching and file handling. The goal of grepreaper is to make these advanced capabilities accessible to R users without requiring them to learn command-line tools, effectively bridging the gap between R and grep for more powerful data processing.

Solution: The package will be built around two core functions:

- grep.count: This function will count the number of records matching a specified pattern in one or more files, and it will return the count before loading the data.
- grep.read: This function will allow users to read and aggregate data from one or more files while filtering the data based on a matching pattern.

These functions will give users the flexibility to provide their own grep commands and additional command-line options such as inverted matches or the ability to handle full or partial patterns. The package will be optimized to ensure users can quickly process large datasets without the need to manually interact with the command line.

Objectives:

- Create User-Friendly Functions: Develop wrapper functions for grep that allow users to count matching records and read data based on specific patterns, making advanced file reading tasks accessible to R users.
- Pattern Matching and File Aggregation: Implement functionality to read and aggregate data from multiple files while matching specific patterns, with full support for advanced regular expressions.
- Pre-Reading Record Count: Allow users to count matching records before reading the full data, helping them estimate the size of the dataset.
- Error Handling and Flexibility: Ensure robust error handling, including file not found errors or invalid patterns, while providing users the ability to customize the command-line options.
- Documentation and Usability: Provide thorough documentation with usage examples, explanations, and troubleshooting tips, ensuring that the package is easy for users to understand and use.

Please include programming languages and technologies you plan to use.

programming languages: R

Technologies: data.table, system() (for interacting with command-line tools), grep, fread()

Detailed

Describe your proposed solution in as much detail as you can. Explain what algorithms/technologies you intend to use/study (if any). You can also include links to additional details like diagrams, etc., outlining your ideas acting as supplementary information for your proposal outside of this scope.

The strongest proposals will help us understand _why_ these features are a good idea. How will the features you propose help the project or its users?

The grepreaper package will leverage the powerful capabilities of the grep command-line tool and integrate them seamlessly with R, offering users a set of advanced functions for efficient file reading and data manipulation. The core idea behind this package is to provide R users with easy-to-use functions that allow them to perform advanced data operations (such as searching, counting, and aggregating data) using patterns, without needing to leave the R environment or interact directly with the command line.

The package will primarily use the following technologies:

1. R and the data.table package: The data.table package will be used for reading large datasets efficiently using the fread() function, which will be used in combination with grep commands. This will allow users to read data directly from files and perform operations without loading the entire dataset into memory.
2. grep: This powerful command-line tool will be used to search for patterns in files, count matching records, and extract relevant data. It is widely used in Unix-like systems for text processing and is highly optimized for these tasks.
3. Regular Expressions: We will provide robust support for regular expressions, allowing users to search for complex patterns in their data. This includes the use of extended regular expressions with the -E flag, which supports advanced matching logic (e.g., handling partial patterns, inverted matches).
4. Command-Line Integration: By using R's system() function, we can directly invoke the grep command in the background, providing R users with an efficient way to use grep for file processing, while abstracting away the complexity of command-line programming.

Proposed Features and Their Impact
The proposed features are designed to streamline and optimize the workflow for users who need to perform pattern-based searches and aggregations on large text files, which is a common task in data science, especially for processing logs, CSVs, and large datasets.

1. grep.count: This feature will allow users to count the number of matching records in one or more files before loading the data. This is especially useful when dealing with large datasets, as it provides an efficient way to quickly assess the size of the data of interest without having to load all the data into memory. This feature will be highly beneficial for users who are working with massive text-based files and need to filter out irrelevant data quickly.
2. grep.read: This feature will enable users to read data from one or more files while applying pattern matching. It supports aggregation across multiple files, assuming identical structures, and returns only the data of interest. By reading data in chunks and only fetching matching records, this functionality will significantly reduce memory usage when processing large files and improve the speed of data processing tasks. Additionally, it allows users to seamlessly aggregate data from multiple sources.
3. Error Handling and Flexibility: The package will allow users to pass custom command-line options to grep, enabling flexibility for more advanced use cases. For example, users could specify -v for inverted matches or use -E for extended regular expressions, providing them with a highly customizable and powerful tool for text processing. The package will also handle common errors (e.g., missing files or invalid patterns), making it more user-friendly.
4. Multi-File Support and Aggregation: The ability to process multiple files simultaneously and aggregate matching data will be crucial for users working with log files or large datasets spread across several files. This feature will reduce the need for manual processing and enable users to handle large volumes of data efficiently. It will also

provide more advanced functionalities for users who need to process datasets that span multiple files, saving time and effort.

5. Output Customization: The package will allow users to customize the output, whether they want to see only the matching data, the grep command itself, or both. This will give users greater control over their results, allowing them to decide what information is most relevant for their analysis.

Project Plan

How do you plan to spend your summer? Please include a preliminary plan, broken down by pre-Midterm and post-Midterm, on how you think you can fulfill your project in the time allocated. Try to be as specific and realistic as you can with your goals and timeline.

The project plan and its timeline will form a significant part of the assessment of your application, as well as mid-term and final evaluations.

Important Schedule:

June 6 - June 13: Final week of school.

June 14 - September 23: Summer vacation.

project plan

5.8-6.1:

Contact the mentor to discuss the feasibility of the plan.

Set up a GitHub repository to track version control in real-time.

6.2-8.25:

Week 1

- Set up a GitHub repository to track version control and project tracking.
- Familiarize with the grep command-line tool, fread(), and the various options available for file reading and pattern matching.
- Begin developing the grep.count function, which will count matching records before reading the entire data.

Week 2

- Continue developing the grep.read function for reading data and aggregating it from multiple files based on a matching pattern.
- Begin implementing error handling for common issues (e.g., missing files, invalid regex patterns).

Week 3

- Implement support for extended regular expressions using the -E flag in grep to handle more complex patterns.
- Add functionality to allow users to pass custom command-line arguments to grep, such as inverted matches (-v) or only showing matching lines (-o).

Week 4

- Test both functions using different datasets (CSV, TXT, etc.) to ensure correct functionality across file formats.
- Begin documenting the functions and their parameters, providing usage examples.

Week 5

- Implement the functionality for counting matching records before reading data using grep -c.
- Start optimizing the performance of the functions to handle large files efficiently, ensuring the package can process data without excessive memory use.

Week 6

- Continue performance testing with larger datasets, making necessary optimizations to improve speed and memory usage.
- Add more error handling features, such as providing clear error messages for issues like invalid file paths or unsupported file formats.

Week 7

- Develop the functionality for aggregating data from multiple files, ensuring that files with identical structures are processed correctly.
- Test the functionality on multiple files to ensure it works as expected.

Week 8

- Begin building a minimal user interface (CLI or Shiny app) to allow users to interact with the functions easily.
- Continue refining the functions, addressing edge cases, and improving performance.

Week 9

- **F**ocus on writing unit tests for each function using the testthat package to ensure robustness.
- Refine the user interface based on feedback and usability testing.

Week 10

- Write detailed documentation for the package, including function descriptions, installation guides, and examples.
- Ensure that the README file provides comprehensive instructions for using the package.

Week 11

- Finalize testing and complete the documentation.
- Address any remaining issues or bugs identified during testing.

Week 12

- Prepare a final report summarizing the development process, challenges, and the outcomes achieved.
- Demonstrate the completed functionality through examples and use cases.
- Submit the package to CRAN or publish it on GitHub.