Project group #7: Visualization of Type Usage in R

Julia Belyakova, Cameron Moy, Younes El Idrissi

CS 7250 Spring 2019 — Prof. Cody Dunne, Northeastern University

Outline

1. Introduction & Motivation
2. Demonstration
3. Technical Aspects
4. Conclusion & Perspective

Introduction

- R is a dynamically typed programming language (known for statistics).
- Our service-learning partner, Alexi Turcotte, is designing a static type system for R.
- It will be based on empirical data from a dynamic analysis of R.

Dynamic analysis records information about function calls

```
x = 2 * 3
```


```
base.* | integer, integer -> integer
```

Motivation

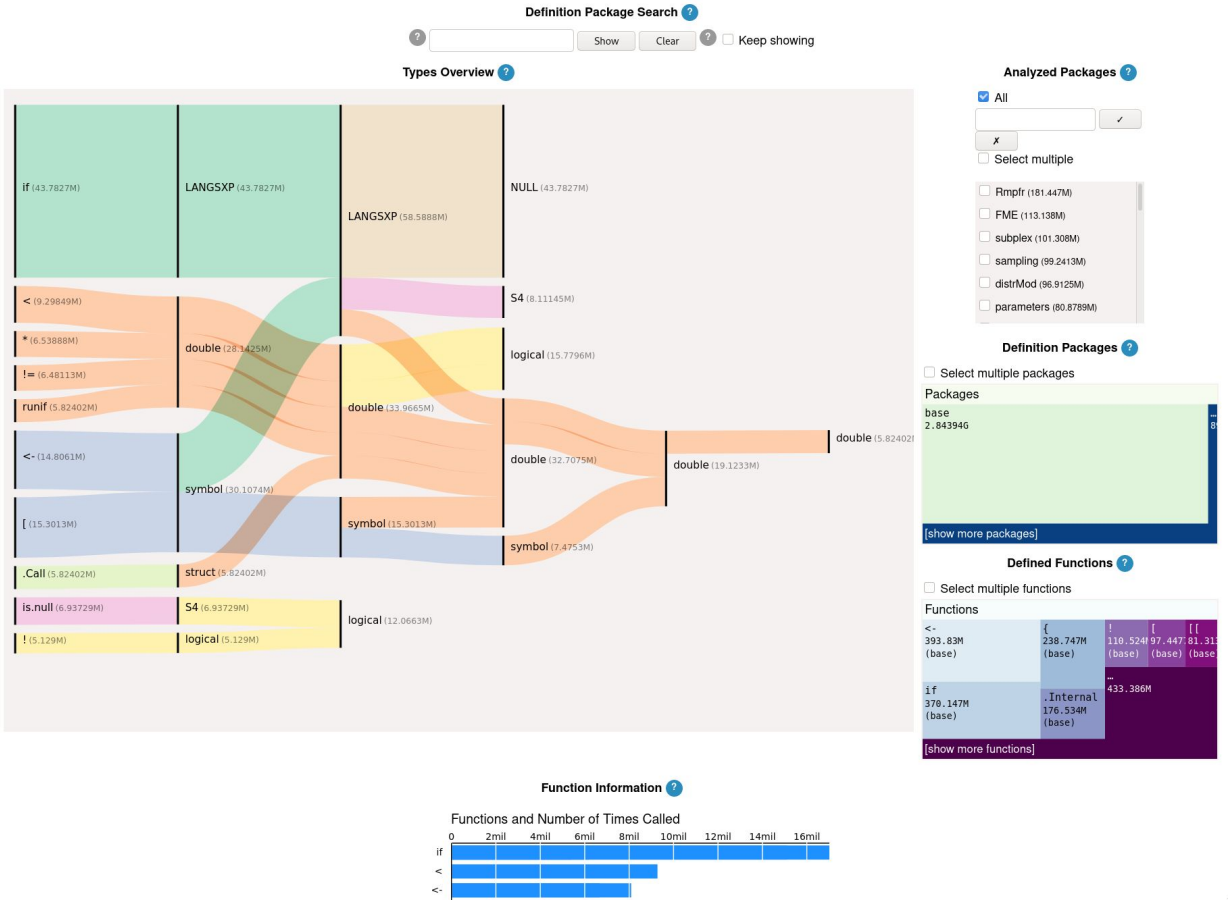
- Unfortunately, the existing dataset is massive (several gigabytes).
- The data is stored in CSV files, so it is difficult to reason about.

Our project visualizes this type information in a way amenable to analysis.

Goals

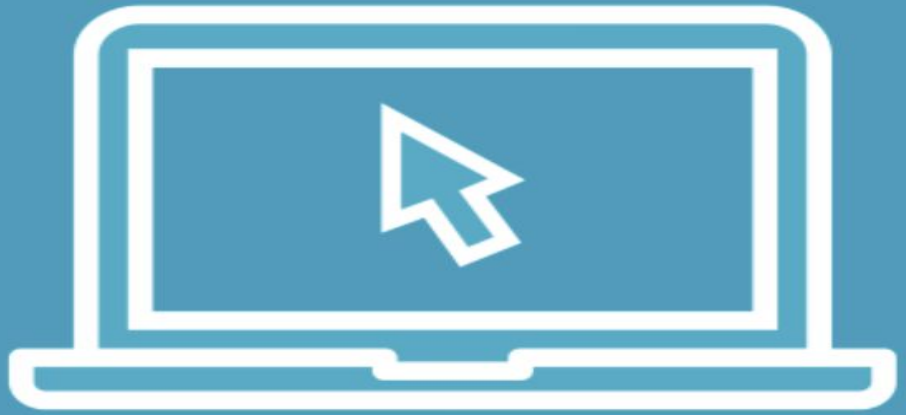
- A user should be able to extract common usage patterns and discover widely used type signatures.
  - How common are polymorphic functions and what arguments are they called with?
- Our partner should be able to derive insights from our visualizations that will help in designing a gradual type system for R.

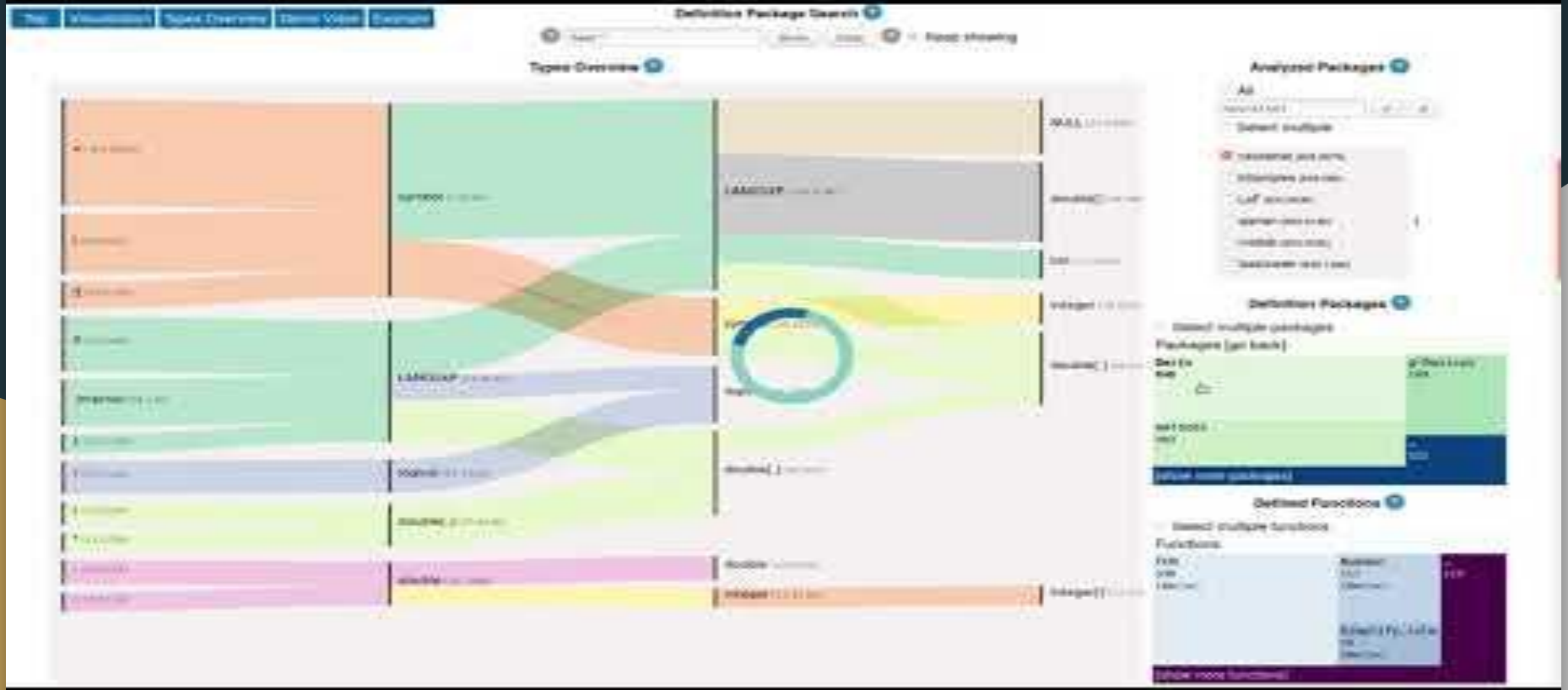
The Final Design



Visualization of Type Usage in R

Demo




Definition Package Search
Types Overview
Analyzed Packages
Definition Packages
Defined Functions

Frontend

- HTML, CSS, JavaScript
- D3 (and many libraries)
  - Sankey
  - Treemap
  - Dispatch
- Dispatch and modularity was important. Getting our internal protocol correct made collaboration possible.

Backend

- Python
- Flask
- SQLite3
  - Large amounts of data must be processed server-side.
  - Migration script to convert CSV to SQLite database.

Microframework

Minimal features

Minimal code

Challenges (Sankey)

- How do we perform brushing and linking on Sankey?
- SVG doesn’t have implementation of intersection primitives.
- Using a library gave poor performance (linear in number of paths).
- Heavyweight solution: uniformly spaced points within paths stored in a quadtree.

Challenges (Frontend)

- Designed a nice protocol for each component to communicate.
  - Doesn’t work well when requirements keep changing.
  - We need more events, different events, things get messy.
  - Everything broke many many times.
- Had to preprocess data for the treemap.
  - Treemap needs hierarchical data (but the information is linear).
  - Information is not visible if there are too many elements or they are disproportional.

Challenges (Backend)

- SQL is fast, but we have a tons of data.
- A second for a query over millions of rows is fast, but not fast enough.
- Especially when the query requires computed columns (e.g. summing).
- Solutions:
  - Caching
  - Indexing
  - Aggregation
- Sadly, not as real-time as we would like (need a spinner still).

Conclusion and Perspective

- Our service learning partner was quite happy with our visualization. We have already incorporated some of his previous feedback into the final design.
- Where are we taking this?
  - We plan to incorporate this visualization into Alexi’s paper.
  - There are plenty more features to add!

Thank you for your attention!