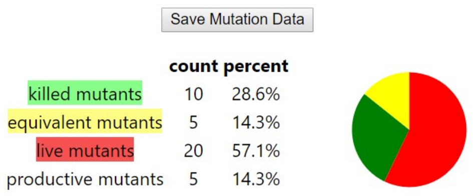
What is MutViz?

MutViz is a web-based interface that can be used to visualize the output of an existing mutation analyzer. The interface accepts a formatted JSON file with metadata about each mutant (see cover photo example). The main page displays a table consisting of all the mutants described in the input file and allows users to inspect each mutation, marking them as "equivalent" and/or "productive." The interface also displays an overall summary about the mutants that illustrates how many are live, equivalent, killed, and productive. At any time, the updated input file can be downloaded.



	count	percent
killed mutants	10	28.6%
equivalent mutants	5	14.3%
live mutants	20	57.1%
productive mutants	5	14.3%

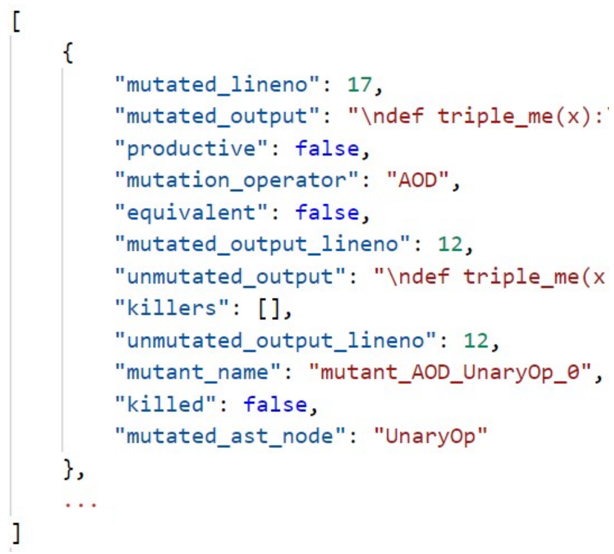
Example summary chart displayed by MutViz, with option to save current mutation data.

Future Work

MutViz can be easily integrated with an existing mutation analyzer. However, we are still looking to improve its functionality. Possible improvements include **(1) hosting the tool online** for even greater ease of use **(2) leveraging a database** either remotely or locally to persist mutation data rather than relying on JSON uploads (offloading the input processing to a server rather than the app) **(3) providing a better visualization scheme for higher order mutants** (namely, a better diff visualization for multi-line mutations).

MutViz

A Tool to Visualize Mutation Results



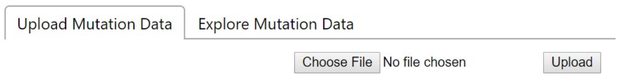
```
[
    {
        "mutated_lineno": 17,
        "mutated_output": "\ndef triple_me(x):
        "productive": false,
        "mutation_operator": "AOD",
        "equivalent": false,
        "mutated_output_lineno": 12,
        "unmutated_output": "\ndef triple_me(x
        "killers": [],
        "unmutated_output_lineno": 12,
        "mutant_name": "mutant_AOD_UnaryOp_0",
        "killed": false,
        "mutated_ast_node": "UnaryOp"
    },
    ...
]
```

University of Washington
CSEP 590 Spring 2019
Josh Bean, Ayaz Latif, Sai Nimmagadda, Rajneil Rana

Motivation

Mutation testing is a powerful method to evaluate the strength of a test suite. However, it has seen little adoption since its inception due to the productivity burden it poses on developers. While generating and executing mutants is relatively cheap, writing tests is expensive. **Thus, being able to efficiently reason about mutation data can help developers focus on the important mutants (those that elicit effective tests) rather than those that are unproductive.**

Many mutation analyzers simply present mutation results as command line output, often displaying little to no additional information than the percentage of mutants killed. This makes it extremely difficult for developers to reason about mutation data, as they cannot see which mutants were killed, which survived, which tests killed mutants, and more. **MutViz aims to solve these issues by providing an intuitive and engaging web-interface that allows developers to efficiently reason about and interact with mutation data.**



MutViz provides the option to upload and explore mutation results.

Design

MutViz is a dynamic React based application that can be easily ported to a hosted site. Users do not have to execute scripts locally to generate visualization content but can instead simply upload a formatted JSON file that is not tied to a specific language or framework. Once the input data is uploaded, users are directed to an exploration page in which they can begin analyzing their mutants.

Uploaded JSON mutation data is persisted in the state of the app component. That is, any time the state changes (such as another file is uploaded or a mutant is edited in-app), the changes are propagated throughout the website. Since mutants may or may not be productive and/or equivalent (sometimes this can only be determined after analyzing the mutant), **we added switches for users to mark mutants as appropriate**. These switches also update the app state appropriately (such as updating the summary page and table information). **The user can save the (updated) mutation data at any time** to a new file to store as a record for future uploads or as data to potentially feed back into a mutation analyzer to improve future test executions.

We discussed several different designs for MutViz, including possibly displaying the entire source code and/or abstract syntax tree (with the applied mutations). However, we ultimately decided on the simpler, more flexible approach described above.

Testing Strategy

During the development of MutViz, smaller component implementations often changed. However, the core design always adhered to the high-level specification. To accommodate this workflow, we employ some snapshot testing to verify that specific HTML components are rendered, but **primarily rely on tests that mock dependencies and "spy" on the behavior being tested**. That is, any component not under test is "mocked" out such that only the behavior of the component under test is evaluated. This verifies that the appropriate calls are made. To help guide us, we used the code converge tool generated by React.

Evaluation

Prior to developing MutViz, we examined several different mutation tools. We found that these tools **typically failed to present mutation results in a comprehensible manner and were often tied to a specific language or framework**. We used these shortcomings as our primary evaluation criteria when comparing our project to other tools. We also wanted to provide a level of interaction not seen in other tools to further improve user ease and productivity while analyzing potentially large amounts of mutation data.

Project Links

https://github.com/jbean96/Mutation-Visualization
https://github.com/sainimma/muupi