Exposing the TestU01 and PractRand PRNG test suites to scientific libraries via Python

Sean Callahan, Maximillian Otto vonBlankenburg, and Mark Gondree

Computer Science Department, Sonoma State University

Objectives

Bring the collection of PRNGs and Statistical Tests from the *TestU01* and *PractRand* libraries to Python, to assist scientific research on PRNG security.

- Determine the best method to pass arguments to generators and tests using the most idiomatic Python.
- Employ macro-based code generation to simplify logic.
- Create a common interface for returning test results.
- Support cross-library generator & test interoperability.

Project Status

- Exposed all generators and tests from *TestU01* and *PractTand* via Python.
- Re-implemented the SmallCrush *TestU01* battery and the *PractRand* Core and Expanded Tests battery.
- Implemented unit testing for new logic.
- Developed basic support for interoperability testing PRNG data via files.

Background: PRNG Test Suites

Pseudorandom Number Generators (**PRNGs**) are an essential component of many practical secure applications.

Security of PRNGs is assessed (broadly) via two approaches: (1) mathematical proofs and (2) statistical tests

Collections of tools exist to support (2), including the:

- NIST Statistical Test Suite
- Dieharder Test Suite
- Diehardest Test Suite
- TestU01 Test Suite
- PractRand Test Suite

The **evaluation** of the efficacy of each tool is an active line of research, hindered by: interfaces limiting direct access to test outputs, incomparable test outputs, incomparable test runners.

Pyhardest

The goal of **Pyhardest** is to **expose existing PRNG test suite logic via Python** —

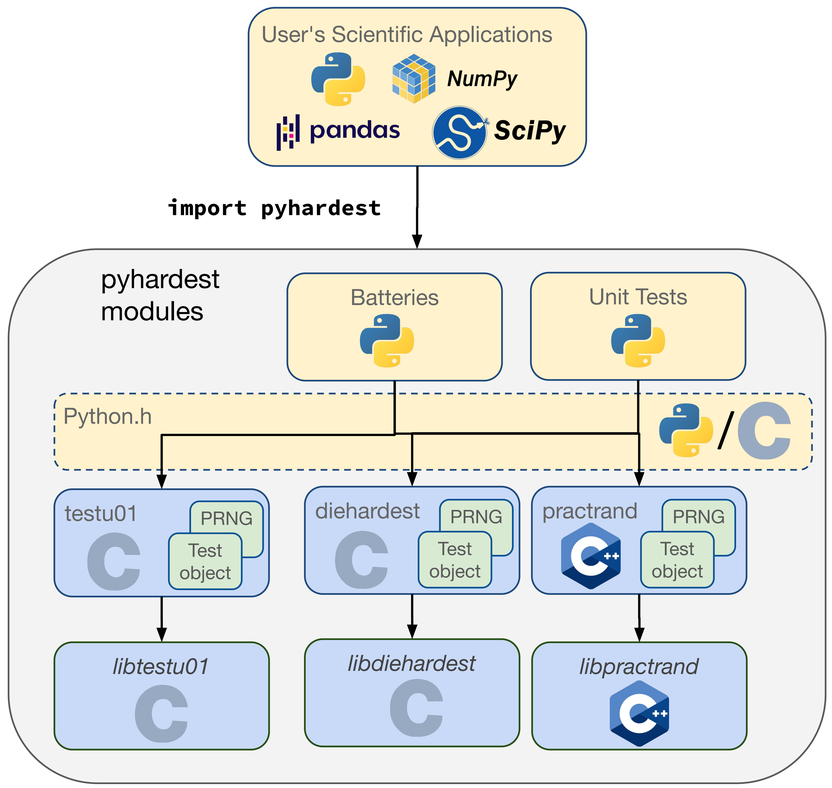
- leverage data science tools available in Python ecosystem
- facilitate apples-to-apples comparisons of suites
- develop new methods to prototype test runner logic

Python Binding of PRNGs and Tests

Exposing test suites allows us to

- Better organize results returned by tests and batteries
- Leverage a common interface for result analysis
- Explore new methods of visualization
- Implement popular test batteries directly in Python
- Experiment with new methods of dynamic testing
- Leverage more standardized unit testing frameworks

Figure: pyhardest library structure



Shown above are three wrapped PRNG libraries and Python unit tests and battery implementations.

- Each library wrapper is organized in its own module
- Provides own interface

Figure: Example of how TestU01 functions are wrapped

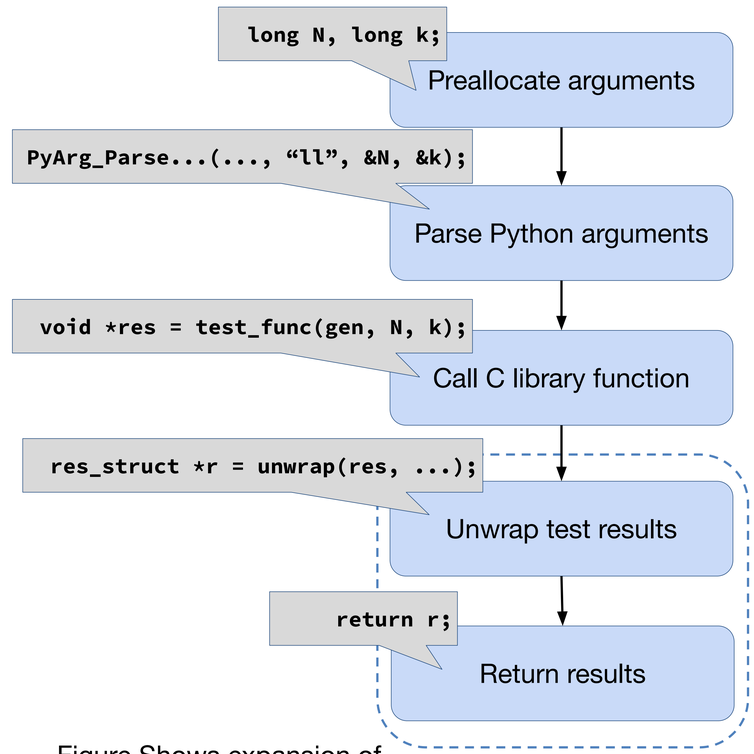


Figure Shows expansion of

```
TESTU01_TEST2(test_func, "ll",
              long, N, long, k)
```

Code Generation using Macros

Determined code generation approach would best promote code reuse and readability.

- Implemented using compiler preprocessor macros
- Performs bounds checking for C library functions
- Unwraps C struct results into Python dictionaries

Summary of Exposed Logic

Test Suite	PRNGs	Tests
TestU01	188	58
PractRand	169	26
Total	357	84

SONOMA STATE UNIVERSITY

Acknowledgments

This work was funded under the **SSU Research, Scholarship and Creative Activities Program**.

Thank you Robert G. Brown (Duke University) for special permission to adapt tests from *Dieharder* under a permissive license.

Future Work

The *pyhardest* project is still a work-in-progress. Ideas of extending these achievements include

- Productizing the project to be distributable via pip
- Enhancing interoperability of tests and PRNGs
- Creating common documentation generation and interface using Sphinx