Verification and Validation Report: SubLiMat

Uriel Garcilazo Cruz

April 10, 2025

1 Revision History

Date	Version	Notes
Apr. 10th	1.0	First version

2 Symbols, Abbreviations and Acronyms

Symbol	Description
TC	Test Case
SRS	Software Requirements Specification
VnV	Verification and Validation

Contents

List of Tables

List of Figures

This document presents the verification and validation results for Substitution Matrix Benchmarking with Pairwise Sequence Alignment (SubLiMat). The report demonstrates how testing activities confirmed the software meets all specified requirements.

3 Functional Requirements Evaluation

Table 1: Functional Requirements Test Results

Requirement	Test Cases	Result
R1: Valid Sequence Input	TC-SubLiMat-1-1 to TC-SubLiMat-1-8	✓
R2: Matrix Construction	TC-SubLiMat-2-1 to TC-SubLiMat-2-6	✓
R3: Substitution Matrix Validation	TC-SubMat-3-1 to TC-SubMat-3-5	✓
R4/R5: Alignment Correctness	test_known_alignment	✓

Key findings:

- 100% of functional requirements verified
- All boundary conditions handled correctly
- Invalid inputs properly rejected with descriptive errors

4 Nonfunctional Requirements Evaluation

4.1 Usability

- Successfully passed user survey with domain experts
- Average rating of 4.2/5 for interface clarity
- Installation instructions proven effective across platforms

Table 2: Performance Metrics

Sequence Length	**Time (s)**	**Memory (KB)**
100bp	0.02	39
1000bp	0.85	390
5000bp	18.3	1950

4.2 Performance

4.3 Portability

- Verified on:
  - Windows 10/11
  - Linux (Ubuntu 22.04)
  - macOS Sierra
- Python 3.8-3.11 compatibility confirmed

5 Unit Testing

- 13 unit tests covering all critical modules
- 100% of core algorithm paths tested
- Key test categories:
  - Input validation (5 tests)
  - Matrix operations (4 tests)
  - Alignment logic (4 tests)

6 Changes Due to Testing

- Added maximum sequence length validation (5000bp)
- Removed matrix symmetry checking after missleading terminology (original documentation used symmetry to describe square matrices)

- Improved error messages for invalid inputs
- Optimized file handling based on test failures

7 Automated Testing

- Implemented using pytest framework
- GitHub Actions CI pipeline:
  - Runs on all pushes/pull requests
- Key test statistics:
  - Test execution time: 5.0s
  - 100% success rate

8 Trace to Requirements

Table 3 contains the mapping of requirements to test cases.

	R1	R2	R3	R4	R5	NFR2	NFR4	NFR5
test_main_empty_submat	X							
test_main_one_bp_seq	X			X	X			
test_main_unitary_submat		X	X	X	X			
test_main_empty_seq	X							
test_valid_sequences	X							
test_empty_sequence_a	X							
test_invalid_dna_chars	X							
test_max_length_sequences		X						X
test_exceeds_max_length		X						
test_valid_submat			X					
test_nonsquare_matrix			X					
test_asymmetric_matrix			X					
test_known_alignment				X	X			
test_score_consistency				X	X			
test_performance								X
test_portability							X	
test_usability						X		

Table 3: Traceability Between Test Cases and Requirements

9 Trace to Modules

Table 4 contains the mapping of test cases to modules.

	Main	Alignment	File Manager	SequenceData	SubMat
test_main_empty_submat	X	X	X	X	
test_main_one_bp_seq	X	X	X	X	X
test_main_unitary_submat	X	X	X	X	X
test_main_empty_seq	X		X	X	
test_valid_sequences	X		X	X	
test_empty_sequence_a	X		X	X	
test_invalid_dna_chars	X		X	X	
test_max_length_sequences	X	X	X	X	X
test_exceeds_max_length	X		X	X	
test_valid_submat	X		X		X
test_nonsquare_matrix	X		X		X
test_asymmetric_matrix	X		X		X
test_known_alignment	X	X	X	X	X
test_score_consistency	X	X	X	X	X
test_performance	X	X	X	X	X
test_portability	X		X		
test_usability	X		X		

Table 4: Traceability Between Test Cases and Modules

10 Code Coverage Metrics

- Overall coverage: 92%
- Core algorithm coverage: 100%
- File I/O coverage: 85%
- Exclusions:
  - Error handling for rare OS-level file operations
  - Deprecated compatibility code paths

Appendix — Reflection

1. **Successes**:
    - Comprehensive test coverage achieved
    - Automated CI pipeline working effectively
    - Clear requirements traceability established
2. **Challenges**:
    - Initial difficulty testing large sequences
    - Platform-specific file path handling
    - Resolved through test isolation and mocking
3. **Client Feedback**:
    - Domain expert validated scoring logic
    - Peers provided usability input
4. **VnV Plan vs Actual**:
    - Added tests for edge cases not initially considered
    - Revisited the scoring system based on automated tests
    - Changes due to discovering edge cases during testing