Module Interface Specification for SubLiMat

Uriel Garcilazo Cruz

April 1, 2025

1 Revision History

Date	Version	Notes
March 10, 2025	1.0	Document's first version

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS Documentation

Contents

3 Introduction

The following document details the Module Interface Specifications for the SubLiMatsoftware. The software is designed to evaluate the effect of given substitution matrices in the quality of the alignment of a set of DNA sequences.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/UGarCil/UGarcil_capstone.

Many components from the present documentation follow the template for a MIS for scientific computing software used in Patel (2023), Bicket (2017). These documentations were adapted from the MIS template.

4 Notation

This section is taken from the MIS template.

> The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$....

The following table summarizes the primitive data types used by the SubLiMatsoftware.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{R}$	a real (i.e. non complex) number defined within (-∞, ∞), which common values between -3.0 to 3.0
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)

The specification of SubLiMat uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SubLiMat uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Alignment (Needleman-Wunsch) Interface Module Substitution Matrix Module Result Concatenation Module Data Output Module
Software Decision	Sequence Data Structure Module F-Matrix Handler Module Backtracking Algorithm Module

Table 1: Module Hierarchy

It is important to highlight the categorization for the Alignment Interface Module in the table above. While the use of the Needleman-Wunsch algorithm sits at a high level behavior that defines the main architecture of the software, it is also a Software Decision, as future releases of the software may opt for other alignment algorithms. Its placement in the documentation as Behaviour-Hiding follows a functional justification by making the Needleman-Wunsch algorithm a high-behavior decision component.

6 MIS of Alignment (Needleman-Wunsch) Module

6.1 Module

needleman_wunsch

6.2 Uses

- **Sequence Data Structure Module**: For representing and manipulating sequences.
- **Substitution Matrix Module**: For accessing substitution matrices and gap penalties.
- **Backtracking Module**: For determining the optimal alignment score.
- **F-Matrix Handler**: To traverse the scores calculated across the pairwise comparison between two sequences.

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
needleman_wunsch	seq0:str, seq1:str, submat: dict	score:float	See Table 2 and 3 in VnV plan

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

None

6.4.3 Assumptions

- The input sequences (seq0 and seq1) are valid strings containing only the characters A, T, G, C or _.
- The substitution matrix (submat) is a valid dictionary containing penalizing costs and a gap penalty.

6.4.4 Access Routine Semantics

align_sequences(seq0: str, seq1: str, submat: dict):

- **Transition**: Computes the optimal global alignment between seq0 and seq1 using the Needleman-Wunsch algorithm.
- **Output**: Returns a dictionary containing the alignment score and other relevant results.
- **Exception**: Raises `InvalidSequenceError` (see Tables 2 and 3 of VnV plan) if either sequence contains invalid characters or goes beyond the boundaries accepted by the program.

6.4.5 Local Functions

- **backtracking(matrix: List[List[float]], seq0: str, seq1: str) → float**:
  - Backtracks through the alignment matrix to determine the optimal alignment path.
  - Returns the final alignment score.
- **evaluateTransTransv(s1: str, s2: str, submat: List[List[float]]) → float**:
  - Evaluates the cost of a transition or transversion between two nucleotides.
- **evaluate(x: int, y: int, dir: str, submat: List[List[float]], matrix_F: List[List[float]], seq0: str, seq1: str) → float**:
  - Evaluates the cost of transitioning from one cell to another in the alignment matrix.

7 MIS of Substitution Matrix Module

7.1 Module

substitution_matrix

7.2 Uses

- **Sequence Data Structure Module**: For representing and manipulating sequences.
- **Alignment (Needleman-Wunsch) Module**: For using substitution matrices in sequence alignment.

7.3 Syntax

7.3.1 Exported Constants

- **penalizingCostOf_baseline**: A 4x4 matrix representing the baseline penalizing costs for nucleotide comparisons.
- **penalizingCostOf_JC**: A 4x4 matrix representing the Jukes-Cantor penalizing costs.
- **penalizingCostOf_K80**: A 4x4 matrix representing the Kimura 1980 penalizing costs.
- **penalizingCostOf_HKY85**: A 4x4 matrix representing the Hasegawa-Kishino-Yano 1985 penalizing costs.
- **penalizingCostOf_TN93**: A 4x4 matrix representing the Tamura-Nei 1993 penalizing costs.

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_substitution_matrix	name: str	submat: dict	see Table 4 in VnV Plan

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

- The substitution matrices are valid 4x4 matrices containing penalizing costs for nucleotide comparisons.
- The gap penalty is a valid float value.

7.4.4 Access Routine Semantics

get_substitution_matrix(name: str) → dict:

- **Transition**: Retrieves the substitution matrix and gap penalty associated with the given name.
- **Output**: Returns a dictionary containing the substitution matrix and gap penalty.
- **Exception**: Raises `InvalidMatrixError` (see Table 4 of VnV Plan for further details) if the matrix dimensions are invalid.

7.4.5 Local Functions

None

8 MIS of Result Concatenation Module

8.1 Module

result_concatenation

8.2 Uses

- **Alignment (Needleman-Wunsch) Module**: For performing sequence alignments and obtaining alignment scores.
- **Substitution Matrix Module**: For accessing substitution matrices and gap penalties.
- **Pandas Library**: For organizing and displaying results in a tabular format.

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	seq0: str, seq1: str	result: List[dict]	

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

- As stated in Alignment (Needleman-Wunsch) and Substitution Matrix modules.
- The Pandas library is available for data manipulation.

8.4.4 Access Routine Semantics

main(seq0: str, seq1: str) → pandas.DataFrame:

- **Transition**: Computes the alignment scores for the given sequences using all available substitution matrices.
- **Output**: Returns a list of dictionaries, where each dictionary contains the name of the substitution matrix and the corresponding alignment score.
- **Exception**: Raises `InvalidSequenceError` if either sequence contains invalid characters.

8.4.5 Local Functions

None

9 MIS of Data Output Module

9.1 Module

data_output

9.2 Uses

- **Result Concatenation Module**: For providing the alignment results to be displayed.
- **Pandas Library**: For organizing and displaying results in a tabular format.

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
display_results	result: List[dict]	None	InvalidResultError

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

- The input result is a valid list of dictionaries, where each dictionary contains the name of the substitution matrix and the corresponding alignment score.
- The Pandas library is available for data manipulation and display.

9.4.4 Access Routine Semantics

display_results(result: List[dict]) → None:

- **Transition**: Converts the list of alignment results into a Pandas DataFrame and displays it in a tabular format.

- **Output**: None (the results are displayed directly to the console or output medium).
- **Exception**: Raises `InvalidResultError` if the input result is not a valid list of dictionaries.

9.4.5 Local Functions

None

10 MIS of Sequence Data Structure Module

10.1 Module

sequence_data_structure

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
validate_sequence	seq: str	bool	InvalidSequenceError

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Assumptions

- Sequences are represented as strings containing only the characters A, T, G, C, or _.
- Sequences are non-empty and valid for alignment purposes.

10.4.4 Access Routine Semantics

validate_sequence(seq: str) → bool:

- **Transition**: Validates the input sequence to ensure it contains only valid characters (A, T, G, C, or _).
- **Output**: Returns `True` if the sequence is valid, otherwise raises `InvalidSequenceError`.
- **Exception**: Raises `InvalidSequenceError` (see Table 2 and 3 from VnV documentation) if the sequence contains invalid characters.

10.4.5 Local Functions

None

11 MIS of F-Matrix Handler Module

11.1 Module

matrix

11.2 Uses

- **Sequence Data Structure Module**: For accessing sequences to be aligned.
- **Substitution Matrix Module**: For accessing substitution matrices and gap penalties.

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize_matrix	seq0: str, seq1: str	matrix: List[List[float]]	InvalidSequenceError
fill_matrix	seq0: str, seq1: str, submat: List[List[float]]	matrix: List[List[float]]	InvalidSequenceError

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

None

11.4.3 Assumptions

- The input sequences (seq0 and seq1) are valid strings containing only the characters A, T, G, C, or _.
- The substitution matrix (submat) is a valid 4x4 matrix of penalizing costs.
- The gap penalty is a valid float value.

11.4.4 Access Routine Semantics

initialize_matrix(seq0: str, seq1: str) → List[List[float]]:

- **Transition**: Initializes the F-matrix with dimensions based on the lengths of seq0 and seq1.
- **Output**: Returns a 2D list (matrix) initialized with zeros.
- **Exception**: Raises `InvalidSequenceError` if either sequence contains invalid characters.

fill_matrix(seq0: str, seq1: str, submat: List[List[float]]) → List[List[float]]:

- **Transition**: Fills the F-matrix with alignment scores based on the sequences and substitution matrix.
- **Output**: Returns the filled F-matrix.
- **Exception**: Raises `InvalidSequenceError` if either sequence contains invalid characters.

11.4.5 Local Functions

None

12 MIS of Backtracking Algorithm Module

12.1 Module

backtracking

12.2 Uses

- **F-Matrix Initialization and Manipulation Module**: For accessing the filled F-matrix.
- **Sequence Data Structure Module**: For accessing the sequences to be aligned.

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
backtracking	matrix: List[List[float]], seq0: str, seq1: str	float	InvalidMatrixError

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Environment Variables

None

12.4.3 Assumptions

- The input matrix is a valid 2D list (F-matrix) filled with alignment scores.
- The input sequences (seq0 and seq1) are valid strings containing only the characters A, T, G, C, or _.

12.4.4 Access Routine Semantics

backtracking(matrix: List[List[float]], seq0: str, seq1: str) → float:

- **Transition**: Backtracks through the F-matrix to determine the optimal alignment path (i.e. the highest value out of the global alignment).
- **Output**: Returns the final alignment score.
- **Exception**: Raises `InvalidMatrixError` if the input matrix is not a valid 2D list or if the sequences are invalid.

12.4.5 Local Functions

None

References

Isobel Bicket. Module interface specification for spectrumimageanalysispy. Software documentation, SpectrumImageAnalysisPy, December 2017. Technical documentation, 231 KB.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

Deesha Patel. Module interface specification for scec (solar cooker energy calculator). Software documentation, CAS-741-Solar-Cooker, April 2023. Technical documentation, 214 KB.