Charlotte Crauwels [1,2,3,†], Sophie-Luise Heidig [1,2,3,4,†], Adrián Díaz[1,2,3] , Wim F. Vranken [1,2,3,5,6,*]

[1]Interuniversity Institute of Bioinformatics in Brussels, ULB-VUB, Brussels, 1050, Belgium

[2]Structural Biology Brussels, Vrije Universiteit Brussel, Brussels, 1050, Belgium

[3]AI Lab, Vrije Universiteit Brussel, Brussels, 1050, Belgium

[4]Evolutionary Biology & Ecology, Université libre de Bruxelles, Brussels, 1050, Belgium

[5]Chemistry Department, Vrije Universiteit Brussel, Brussels, 1050, Belgium

[6]Biomedical Sciences, Vrije Universiteit Brussel, Brussels, 1050, Belgium

*Corresponding author. E-mail: wim.vranken@vub.be

†Equal contribution.

Title

Scalable and User-Friendly Structure-Guided Protein Sequence Alignment with SIMSApiper

Running head

Structure-Guided Protein Sequence Alignment with SIMSApiper

Abstract/Summary

SIMSApiper is a structure-informed multiple sequence alignment tool capable of processing protein datasets ranging from two to thousands of sequences. It integrates structural information, either provided by the user or automatically retrieved from online resources such as the AlphaFold Database to guide the alignment process. To handle large datasets efficiently, SIMSApiper employs a parallelization strategy that clusters sequences into smaller subsets for simultaneous alignment, substantially reducing computational time compared with traditional structure-based methods such as T-Coffee. An optional post-processing step further refines the alignment by minimizing gap regions and leveraging conserved secondary structure elements to enhance accuracy. The pipeline also generates diagnostic visualizations and quality metrics, including sequence entropy and structural alignment assessments.

This chapter provides step-by-step instructions and two illustrative use cases, demonstrating how SIMSApiper can be easily adapted to different datasets, research goals, and computational setups. Developed with Nextflow, SIMSApiper emphasizes reproducibility, modularity, and accessibility, making it a practical solution for evolutionary analyses that require both structural context and scalability.

Key words: Protein alignment, Multiple sequence alignment, MSA, Structure-informed alignment, Large-scale alignment, Nextflow

1 Introduction

Multiple Sequence Alignments (MSAs) are widely used to study protein families, providing insight into conserved or functionally relevant residues, and evolutionary relationships ***(1, 2)***. Sequence-based alignment methods are fast and efficient, which makes them suitable for large datasets. However, their accuracy declines when proteins share low sequence identity (SI) (<30%) or when dealing with multi-domain architectures ***(1–3)***. In these cases, structure-informed MSAs are generally more reliable, capturing conservation at the fold level and therefore serving as benchmarks for evaluating sequence-based approaches ***(2)***.

Historically the main limitation of structure-guided MSAs was the small number of available experimental structures ***(4)***. With the emergence of accurate AI-based prediction of protein structure from sequence, this bottleneck has largely been resolved: over 200 million protein structure models are now available through the AlphaFold Protein Structure Database (AF2 DB) ***(5)***. Methods such as AlphaFold 3 ***(6)*** and ESMFold (ESMF) ***(7)*** can also provide protein structure models of reasonable accuracy within minutes. Even when predicted models are not of experimental accuracy, they still improve alignment quality ***(3, 8)***. The remaining challenge lies in scale: retrieving or predicting structures and computing structure-based alignments for large datasets is computationally intensive, and the complexity grows quickly with the number of sequences ***(9–11)***.

SIMSApiper ***(12)*** is a pipeline that addresses these limitations by generating structure-informed MSAs of any dataset independently of its size or SI. Built in Nextflow ***(13)***, SIMSApiper integrates established tools such as T-Coffee ***(1)*** and MAFFT ***(14)*** within a framework designed for efficiency and reproducibility across platforms. Protein sequences are divided into subsets, either defined by the user or determined by SI thresholds, to reduce computational burden and enable analyses of thousands of proteins. Structural information can be provided directly or obtained through prediction resources. The extracted secondary structure information is used to help generate accurate alignments and in a subsequent step to refine them by reducing gaps. Benchmarking demonstrates that SIMSApiper achieves alignment quality comparable to T-Coffee while offering significant performance improvements, running up to twenty times faster on large datasets such as a TIM-barrel proteins set with 350 sequences ***(12)***.

In this chapter, we provide a detailed procedure for installing and setting up SIMSApiper. We also include a flowchart and practical examples to illustrate how SIMSApiper can help you generate an accurate MSA tailored to your dataset's size and complexity. In addition, we provide practical tips to support your analysis and help you overcome potential technical challenges.

2 Materials

In this section we outline the computational resources, environment setup, and input files required to run SIMSApiper. SIMSApiper is a command line tool tested in Linux and MacOS environments. We expect the users to have basic knowledge of the command line interface (e.g. *cd*, *mkdir*) and to understand absolute and relative file paths. The Methods section below will detail how to execute SIMSApiper using the appropriate combination of flags when executing the software via a terminal or a custom launch file. A glossary of all important abbreviations and concepts can be found in Note 1.

2.1 Software and Dependencies

First, SIMSApiper and its dependencies must be installed:

1. Download SIMSApiper pipeline from GitHub.
2. Install Nextflow (tested until v25).
3. Install containerization platform for dependencies, SIMSApiper will handle the rest:
    a. On your own computer, Docker is recommended.
    b. On shared servers or HPC environments, Apptainer (formerly Singularity) is recommended and may be already installed.
4. If Docker or Apptainer are not used: install all required tools locally:
    a. T-Coffee's tool 3DCoffee (V13.45.61.3c310a9) (referenced to as T-Coffee hereafter) with these algorithms for structural alignments:
        i. TMalign (V20190822) ***(19)***
        ii. SAP (V1.1.3) ***(20)***
    b. MAFFT ***(14)*** to merge the sub-alignments.
    c. DSSP (V3.0) for the secondary structure-based refinement.
    d. CD-Hit and BLAST (V2.14) for the automatic subset generation.
    e. Python packages: Biopython, Pandas, Matplotlib, secstructartist.

2.2 Hardware & Storage

The following setup is recommended for running SIMSApiper:

1. Multi-core CPU.
2. Optional: GPU with VRAM>16GB if local structure prediction with ESMF is enabled (see Note 2).
3. Required disk space and RAM depend on the dataset. For example: ~300 sequences of 400 residues with 30% SI require 30GB disk space and 32GB RAM

4. Minimal tested working conditions for *toy_example* dataset (no local structure prediction): MacBook Air 2022, 8 core CPU, 16GB RAM, 5GB storage.

2.3 Input Data

To keep your project organized, we recommend creating a dedicated directory that holds both input and output data (see Figure 1).

Next, the file(s) containing the sequences to align must be provided to SIMSApiper:

1. Place all protein sequence file(s) in the *your_project/data/seqs/* directory (Note 3,4).
2. These may consist of a single FASTA file (Note 5) containing all sequences, or multiple files (Note 6).

Structural input to guide the sequence alignment from the user is optional: none, some, or all structures corresponding to the sequences to be aligned can be provided (Note 7). SIMSApiper can also add missing 3D structures automatically (Note 2,8,9,10)

1. In case of user provided structures, place the structures (in PDB format) in the *your_project/data/structures/* directory.
2. To enable SIMSApiper to correctly link a structure to its sequence, the filename must exactly match the corresponding sequence identifier used in the input file. For example, *>P25106* (identifier in FASTA file) will only match *P25106.pdb* (file in the *your_project/data/structures/* directory).
3. The simplest approach is to omit manually providing structures. This also avoids potential issues associated with experimental 3D structures (Note 7). For sequences without an associated structure in the *structures* directory, SIMSApiper

can automatically retrieve AF2 models from the AF2 DB (Note 8), or predict models using ESMF's API (Note 9) or ESMF locally with GPU support (Note 2).

2.4 Launch File

We recommend running SIMSApiper via a launch file. This approach captures all required settings and flags in a single, editable file. While it is also possible to run SIMSApiper directly from the command line (see GitHub Wiki), using a launch file makes it easier to document, share, and rerun analyses.

In the SIMSApiper repository on GitHub, we provide an example launch file, *magic_align.sh,* which will work out-of-the box if dependencies are provided with Docker. It serves to verify the installation on your machine. Moreover, we provide several other example launch scripts in the main *simsapiper/* directory tailored for different basic use cases described in Table 1.

We recommend copying one of these launch files in your project directory (see location *launch_file.sh* in Figure 1). Next, edit the script with a plain text editor to update paths according to your local setup. In the Methods section, we will further explain how to select the appropriate settings for your specific use case. Note 11 contains trouble shooting advice. Finally, refer to Note 12 how to launch SIMSApiper.

3 Methods

After installing SIMSApiper and providing the input files, its settings are chosen via flags (Note 13) and must be indicated in the launch file or command line. As shown in Figure 2, selecting SIMSApiper's flags comes down to 4 questions:

1. Should the input sequence dataset be cleaned?

2. How many structures are provided by the user?
3. How large is the alignment dataset: do we need to create subsets to keep compute time reasonable?
4. Is post-processing, i.e. gap minimization, required?

While the relevance of these questions was discussed in the

Introduction, choosing the right flags to tackle these questions can be challenging. The examples below illustrate how to select the best set of flags to obtain a high-quality alignment of a dataset of specific size and diversity. Example 1 demonstrates how to use the predefined sets of flags (--*minimagic* and --*magic*) and the basic SIMSApiper workflow. Example 2 shows how to iteratively tune parameters to produce a reproducible, refined alignment and highlights the benefits of parallelization.

3.1 Example 1: Aligning small datasets (<100 sequences)

In this example, we align a dataset of 30 G Protein-Coupled Receptors (GPCRs). This protein family is characterized by a low sequence identity (<30%) but a highly conserved structure, making structural information critical for guiding the alignment. SIMSApiper is particularly well-suited for this task, not only because it uses a structure-based aligner, but also because it can automatically retrieve the required 3D structures from the provided sequences, streamlining the workflow for the user.

A FASTA file containing the sequences to be aligned is placed in the *your_project/data/seqs/* directory (the data is available on GitHub). No structures are provided. In the launch file (see Materials section, see Figure 1), only the --*minimagic* flag is set. This mode is a user-friendly shortcut that is equivalent to setting all following flags together:

```
--data your_project/data
--outFolder your_project/results/simsa_time --outName "minimagicMSA"
--seqQC 10
--useSubsets
--retrieve --model --strucQC 5
--dssp --squeeze --squeezePerc 60
--reorder
```

These flags mean:

1. Data cleaning: *--seqQC 10* removes sequences in which more than 10% of residues are non-standard amino acids.

In practice, for the GPCR dataset, no sequences are removed. Nevertheless, including this flag is good practice to prevent low-quality or biologically irrelevant sequences from negatively impacting the alignment. For larger datasets, stricter cleaning is recommended (Note 14).

2. Structural data retrieval: SIMSApiper first checks if 3D structures have been provided in the *your_project/data/structures/* folder, matching sequence identifiers to structure filenames (see Materials section for naming requirements and Note 4). Missing structural data will automatically be retrieved thanks to the flags *--retrieve* and *--model* set here (Note 8,9). See Note 10 about the order of execution of these two flags. To avoid proceeding without sufficient structural coverage, *--strucQC* ensures that no more than 5% of sequences are missing a structure.

In practice, due to the *--retrieve* flag, 29 AF2 models from the AF2 DB were retrieved, as 29 sequences were named with their UniProt IDs in the inputted sequence file. The remaining sequence was named differently and was therefore modeled via the ESMFold API (due to *--model*). All resulting models are automatically stored in *your_project/data/structures/*.

3. Since the dataset is small, no subsets must be created before alignment (see Note 6, 14 for recommendations about dealing with subsets). This is handled by *--useSubsets* (this is equivalent to *--useSubsets true,* see Note 6), which in this case means *use all sequences directly*.

In practice, in this example, all sequences and their corresponding structures are aligned in a single batch using SIMSApiper's aligner T-Coffee. Multi-batch alignment will be shown in Example 2. The resulting alignment is stored in *your_project/results/simsa_time/msas/merged_minimagicMSA.fasta* (Figure 3, Note 16).

Alignment improvement: Due to the high sequence variability, post-processing is recommended to reduce the number of gaps in the outputted alignment (see

4. Introduction). Post-processing uses:

    --dssp: generates secondary structure assignment files via DSSP (Note 17).

    --squeeze: minimizes gaps by squeezing columns with non-conserved secondary structure elements towards conserved columns.

    --squeezePerc 60: defines a column as conserved if ≥60% of its residues share the same secondary structure.

In practice, this generates 30 DSSP files in *your_project/data/dssp/*, based on the structures in *your_project/data/structures/*. The initial alignment is annotated with the DSSP codes (see *your_project/results/simsa_time/msas/dssp_merged_minimagicMSA.fasta* in the result directory, see Figure 3) and this file is used to identify the columns with conserved secondary structure (60% threshold). The non-conserved columns are then squeezed towards the conserved ones, and this results in the final MSA file (*your_project/results/simsa_time/msas/squeezed_merged_minimagicMSA.fasta*).

Finally, the *--reorder* flag arranges the sequences in the final MSA to match the order of the input file, generating *your_project/results/simsa_time/msas/reordered_squeezed_ merged_minimagicMSA.fasta* and facilitating subsequent analysis.

After generating the MSA, the next step is to analyze it and evaluate its quality. Depending on the dataset, the generic *--minimagic* settings may not have produced the most optimal results (see Example 2). To support this evaluation, SIMSApiper generates a set of log files (Note 18) and plots. Figure 4 is illustrating the alignment of secondary structure elements. Their alignment serves as quality control: if equivalent secondary structures align, this suggests a high-quality alignment. The file *your_project/results/simsa_time/msas/dssp_squeezed_merged_minimagicMSA.fasta* provides a more detailed view of the secondary structure alignment. If an error occurs and no output is generated, refer to the log files and Notes 18-20 to identify and resolve the issue(s).

To summarize, using the *--minimagic* flag allows small datasets (<100 sequences) to be processed easily and efficiently in a generic manner. For larger datasets (>100 sequences), the *--magic* flag can be used in a similar way. The main difference is that *--magic* clusters the data into subsets after cleaning, aligns each subset independently, and then merges the resulting sub-MSAs (see Example 2 and Note 14). In Example 2, we also demonstrate how to adjust the flags to handle a specific dataset.

3.2 Example 2: Aligning large and complex datasets (>100 sequences)

This example aims to demonstrate how to iterate with different flags to generate a reproducible, refined alignment. It also highlights the power of SIMSApiper's parallelization step.

We will consider here a dataset of ca. 4000 non-redundant penicillin binding proteins. They contain a β-lactamase domain, which is known both for its biological and medical relevance in antibiotic resistance as well as its low sequence conservation (~10 conserved residues) while maintaining a defined structure. As such, the structure presents crucial information to generate an MSA.

Below you can find the command line for this example and explanations for the settings:

```
--data your_project/data
--dropSimilar 90
--localModel 4
--createSubsets 30 --minSubsetID "min"
--dssp --squeeze --squeezePerc 80
```

1. Data reduction: Due to the well-defined dataset a 90% SI cutoff is applied (*--dropSimilar 90*, Note 14). The remaining dataset contains only 633 protein sequences.
2. Structure acquisition: Using the GPU resources of our local HPC, SIMSApiper predicted the 3D structures (*--localModel 4*). Since the average sequence length is below 400 amino acids, we can request less time than recommended in Note 2.

Subset creation: While 600 sequences are a lot more manageable than 4000, they still present a challenge to SIMSApiper's aligner T-coffee, as explained in the

3. Introduction. The dataset is therefore split based on a 30% SI cutoff (*--createSubsets 30 --minSubsetID "min",* Note 14*)*.
4. Post-processing: We use a secondary-structure based refinement to make the MSA more compact (*--dssp --squeeze --squeezePerc 80,* Note 17).

SIMSApiper reports (Note 18) for this MSA of 2136 positions nearly 1 million gaps and only 13% SI. So far, we have described the general workflow of SIMSApiper. Next, we will explain how to evaluate the initial MSA and adapt this workflow to integrate external knowledge while maintaining reproducibility.

Figure 5 shows the secondary structure of each sequence mapped onto the MSA. A few sequences are causing huge gaps as no other sequence aligns to them (red arrow). Since these sequences add a lot of noise, it can be useful to prune them from the alignment. The region around position 1700 represents the conserved β-lactamase domain. A few blocks stand out from this alignment (yellow highlight). We can see that the structure of three β-sheets is present in all sequences, yet some are not aligned. This can be an effect of the automatic subset creation, especially given the low SI in this dataset. SIMSApiper outputs these automatically generated subsets from the previous run in the results directory *(your_project/results/simsa_time/seqs/CD-HIT_for_t-coffee/*.fasta)*.

We use these previously generated subsets as a new input to SIMSApiper (Note 6). To counteract the influence of the subsets, we merge each misaligned subset (yellow highlight, Figure 5) with a separate, larger subset. Now we rerun SIMSApiper using these new settings:

```
--data your_project/data
--useSubsets
--dsspPath your_project/data/dssp --squeeze "E" --squeezePerc 70
```

1. In previous run, SIMSApiper already reduced the dataset's size and manually curated the subsets (*--useSubsets*). It will also automatically find that all structures were already predicted previously. We can therefore omit the lines *--dropSimilar 0.9 --createSubsets 30 --minSubsetID "min" --localModel 4.*
2. Providing the path to the secondary structure files ensures that the *--dssp* step is also not repeated (*--dsspPath your_project/data/dssp*).

3. Since in this example we are mainly interested in the β-sheets, we can adapt our previous postprocessing settings to reflect that: *--squeeze "E"* (Note 17) and *--squeezePerc 70.*

Checking the logfiles (Note 18), the larger subsets result in a slower structure-based alignment step (6h vs 8h if run consecutively). The resulting alignment contains 1500 positions and 600k gaps, which is much more compact than before. Moreover, Figure 6 highlights how much the post-processing settings influence the final alignment. Our section of interest, now around position 1200, shows the three columns of highly conserved β-sheets we expected based on the 3D structures.

SIMSApiper's standard settings aim to be as generic as possible (*--squeeze "H,E"*). The settings chosen in this example (*--squeeze "E"*) aimed for a clean alignment of the β-sheets in the dataset (around position 700 and 1200), at the cost of the alignment of sections with only helices (around position 500-700), which got shifted towards a less optimal final state. The post-processing settings can be adapted without redoing the slow structure-based alignment step using the *–resume* option (Note 12.d).

This highlights how SIMSApiper allows to include external knowledge in the alignment process. SIMSApiper automatically reports all settings and intermediate steps in its output files to ensure reproducibility.

5. Notes

1. Glossary:

a. MSA (Multiple Sequence Alignment): A matrix where corresponding (ideally homologous residues from different sequences are aligned in the same column. Positions without a corresponding residue in a sequence are represented by a gap.
b. Gap: A placeholder in an MSA indicating the absence of a corresponding residue in a given sequence at that alignment position.
c. SI (Sequence Identity): A measure of similarity between two or more sequences, expressed as the percentage of identical residues at aligned positions.
d. Subset: Partial set of sequences from the complete dataset, usually defined based on SI thresholds.
e. Secondary structure: The local arrangement of the amino acid chain, such as α-helices, β-sheets, or loops.
f. 3D structure: A static representation of the spatial arrangement of the amino acid chain, illustrating how helices, sheets, and loops are positioned relative to one another.
g. Launch file: A shell script (with a .sh extension) used to execute a predefined set of commands.
h. Flags: Command-line options that modify the behavior of SIMSApiper runs, typically written in snake case (e.g. *--my_flag*).

2. Prediction of 3D models with ESMF on local GPUs: Use *--localModel n* (e.g. *--localModel 1*) to avoid ESMF's API sequence length restrictions (Note 9):
   a. Increase *n* by 1 for each set of 100 proteins to model. It represents the number of hours requested via SLURM on the HPC. You will have to adapt the

clusterOptions line in the *simsapiper/nextflow.config* file if you are using the profiles *server* or *hpc* to your local environment (see Note 11.d, 12.f).

   b. Predictions are very fast (~1 minute per sequence of 400 amino acids) but the prediction times are influenced by the sequence length (increases modelling time exponentially) and the VRAM of the graphics card (ideally 40GB).
   c. If possible, favor retrieval of 3D models from the AF2 DB (Note 8).
3. Sequence file(s) naming: Spaces are not permitted in the filenames of input sequence files. Use underscores "_" instead.
4. Sequence identifiers:
   a. Spaces, as well as .,*/;: are not permitted. Use underscores "_" instead. SIMSApiper changes the sequence identifiers automatically to this pipeline-safe format. If structures are provided, it will not be able to match renamed sequence identifiers to structure files. For example, the sequence previously named >*seq 1:a*, renamed to >*seq_1_a* by SIMSApiper, will not match the provided structure file *seq 1:a.pdb*.
   b. The length of the sequence identifiers in the sequence file(s) must not exceed 30 characters.
   c. To enable SIMSApiper to retrieve the 3D model of the sequences from AF2 DB (--*retrieve*), use the Uniprot ID as sequence identifier.
5. Format sequence file(s): If the input sequence file(s) are not in FASTA format, specify the format using the --seqFormat option, following Biopython format nomenclature.
6. Using custom subsets: The default SI-based clustering (Note 14) can be overly stringent, potentially separating sequences that should be aligned together:

a. The clusters' composition can influence the overall alignment quality of the merged alignment.

b. Custom-defined subsets allow for more biologically informed grouping, which may lead to improved results, especially in heterogeneous datasets.

c. To use custom subsets, place the individual sequence files (each representing a subset) in the *your_project/data/seqs/* directory and enable the *--useSubsets* flag.

7. User provided structural information:

   a. The structures can be experimentally resolved or predicted.

   b. Must be in the PDB format.

   c. Files including structures with multiple chains are not allowed. Extract the chain of interest using pdb-tools. Also, make sure to extract the chain of interest in the SEQRES section.

   d. Sequence variations of the 3D structure compared to the user inputted sequence to aligned will not impact the MSA quality if the mutations do not impact the overall organization of the protein (*pdb_min_sim* flag of the aligner T-Coffee set to 1 by default). Nevertheless, the post-processing step (squeezing) will fail if more than 5% of your protein structure is mutated compared to its sequence.

   e. The sequence to align can be a section of the sequence of the structure (*pdb_min_cov* parameter of the aligner T-Coffee set to 1 by default)

   f. Modeled structure information (even when of lower quality) has been shown to improve the quality of MSAs ***(3, 8)***.

   g. Omit issues related to experimental 3D structures (e.g., mutations, multiple chains, …) and let SIMSApiper handle it. SIMSApiper can compute/retrieve the predicted model of your sequences (see the flags *--retrieve*, *--model* and *--localModel*, see Notes 2,8,9,10).

8. Retrieval of 3D models from the AF2 DB: When using *--retrieve*, ensure that only the UniProt ID is used as the sequence identifier in the sequence input files. This is required for SIMSApiper to locate and download the corresponding model from the AF2 DB.
9. Prediction of 3D models with ESMF: When using *--model*, SIMSApiper predicts models through the ESMFold API. This approach has two limitations:
    a. Only sequences shorter than 400 residues can be modeled.
    b. The API may become overloaded and stop responding if too many sequences are submitted at once. If this occurs, relaunch SIMSApiper.

    If possible, favor retrieval of 3D models from the AF2 DB (Note 8).
10. Resource aware structure retrieval: If more than one flag to retrieve/predict 3D models (Notes 2,7,8,9) is set, the order of action is always the following: (i) check which models are provided locally, (ii) *--retrieve*, (iii) *--model*, (iv) *--localModel*. After every step, SIMSApiper compares the available structures with the list of sequence identifiers and searches only for structures still missing.
11. Problems with the launch file may be caused by one of these:
    a. Permission errors can be fixed by running *chmod +x launch_file.sh* in the terminal and rerun.
    b. Spaces behind "\" in the launch file, there cannot be any.
    c. Replace "|& tee" with " >>"
    d. Select a different execution profile, defined in the *simsapiper/nextflow.config* file, matching your local environment, from Table 2.

12. SIMSApiper is a Nextflow pipeline. Here are a few general tips about running Nextflow pipelines:
    a. To launch the pipeline use: run *./launch_file.sh* in the terminal.
    b. Cancel a running Nextflow job: Crtl + C
    c. Pipeline failed to start: Confirm that you provide absolute file paths to your Nextflow pipeline and data directory:
        i. Absolute path: */Users/me/simsapiper/toy_example/data*
        ii. Relative path from SIMSApiper directory: *toy_example/data*
    d. Pipeline failed to complete:
        i. to rerun the last job: append *-resume* to the launch command
        ii. to rerun a specific job: check the last line of the execution log file (Note 18.d) to get the unique hash:
        nextflow run simsapiper.nf -resume
        9ae6b81a-47ba-4a37-a746-cdb3500bee0f
        iii. Attention: last state will be permanently overwritten.
    e. All intermediate results are unique subdirectory in the work directory. The unique subdirectory can be found in the execution log:
    [f0/da5197] process > getAFmodels (Q4JCD8)
    means that in *your_project/work/f0/da5197 b16b69f0e4b8a6af4c006083* you will find the intermediate files for retrieving the structure of Q4JCD8 from AF2 DB
    f. Learn how to adapt this pipeline using Nextflow here.
13. Flags:

   a. One can combine one of the predefined set of flags (*--magic* or *--minimagic*) with other flags. For example, *--minimagic --dropSimilar 90* will apply the *minimagic* flags and in addition, the *dropSimilar* flag.
   b. One can also overwrite a flag of *--minimagic* will keeping the other ones set. For example, *--minimagic --seqQC 5* will set *--seqQC* to 5% instead of the default 10% and leave all the other predefined flags untouched.
   c. When a flag has no value (e.g., *--useSubsets*), it defaults to *True*.
   d. Most flags in SIMSApiper are set to *False* by default, ensuring users are explicitly aware of which steps will be executed. A full table of available flags and their default values can be found on GitHub.

14. Data reduction: Even if the global SI of a dataset is low, pairs of sequences can be mostly identical and can be represented by each other (redundant dataset). A data reduction step is recommended, e.g. retaining only sequences that have less than 90% SI *(--dropSimilar 90*). The remaining dataset contains less protein sequences but still encapsulates most of the sequence diversity. When studying protein families, *--dropSimilar 70* is recommended instead as it will provide a more general overview of the dataset.

Using subsets: To accelerate the alignment process (see

15. Introduction), SIMSApiper supports a clustering-based approach in which sequences are grouped into subsets, aligned independently, and subsequently merged into a single final alignment.
    a. The best practice for merging sub-alignments is to create the subsets based on a phylogenetic tree. SIMSApiper does not support such approach but allows the usage of custom subsets (see Note 6).

   b. SIMSApiper creates subsets based on SI, as sufficient and fast approximation. These subsets can be generated automatically using the *--createSubsets 50* option, which clusters sequences based on SI, here with a threshold of 50%.
   c. The SIMSApiper generated subsets can be joined or split to archive a relative equal distribution of sequences per subset *--minSubsetID "min"* (to collate small clusters and ensure a minimal number of subsets). *--maxSubsetSize* can be used to set the maximal number of sequences in a subset. We recommend a maximum cluster size of 100 sequences for proteins smaller than 400 amino acids while 50 is recommended for proteins that are larger.

16. Naming of output alignment files: SIMSApiper generates multiple alignment files during its run. Each filename reflects the step at which it was produced (e.g., merged_minimagicMSA.fasta, squeezed_merged_minimagicMSA.fasta). The sequence of file generation corresponds to the pipeline's processing order. For details on which file corresponds to which data or step, refer to the SIMSApiper's documentation.
17. Post-processing:
   a. With *--dssp*, the DSSP files of the 3D structures in *your_project/data/structures/* are generated. DSSP translates structure models into 2D secondary structure nomenclature using the DSSP codes. These files are collected in *your_project/data/dssp/*. Next, the DSSP codes of each amino acid are mapped onto their alignment.
   b. *--dssp* is required for *--squeeze* as this needs the MSA with mapped secondary structure to identify the columns with conserved secondary structure elements.
   c. When *--squeeze* is set to true, SIMSApiper automatically looks for columns with conserved sheets (DSSP code "E") and helices (DSSP code 'H,I,G") and

squeezes towards these. Instead, if for example --*squeeze "E"* is used, SIMSApiper will only look for conserved sheets and squeeze towards these.

d. We recommend analyzing the merged MSA files from before and after post-processing (squeezing) to become aware of possible problems the post-processing can create (see Example 2)

e. With *squeezePerc* the stringency of the post-processing step can be adjusted: a lower threshold (60%) is recommended for small dataset, while the default requires a secondary structure element to be conserved in 80% of sequences per MSA column.

18. Log files:

    a. Summary (*your_project/results/simsa_time /simsapiper_summary.md*): Reports settings of all used parameters and results of intermediate steps. Paths to all intermediate output files are also mentioned as well as MSA statistics (gap reduction after post-processing, average pairwise sequence identity, …).

    b. Plots (.pdf files) and related raw data (.csv files) in *your_project/results/simsa_time/msa_stats*: Illustrate for example Shannon entropy conservation along the MSA and Occurrence of secondary structure elements (Helix, Sheet, Loop) per MSA column. More details in the GitHub.

    c. Resource log *(your_project/results/simsa_time/resources*.txt)*: Overview of used resources during every SIMSApiper's step.

    d. Execution log (*your_project/results/simsa_time/run_report*.nflog)*: Captures terminal output of Nextflow pipeline which facilitates error tracing. Moreover, it captures unique execution hash and flags for resuming the specific job (see Note 12.d.ii).

19. Problems at the CD-hit stage may be caused by one of these:

a. Permission errors can be fixed by running

chmod +x simsapiper/bin/psi-cd-hit.pl and

chmod +x simsapiper/bin/psi-cd-hit-local.pl

b. Check if you are trying to align <10 sequences. Do not use CD-Hit generated subsets in this case, choose *--useSubsets* instead.

20. Problems at T-Coffee stage:
    a. Error contains *sap_pair* and *--model* is used: possibly a ESMF prediction error, which can be solved by removing the ESMF model of protein in error message from *your_project/data/structures and* choosing one of the approaches below:
        i. Set *--model false* to add protein to the MSA based on sequence.
        ii. *--retrieve* AF2 model of the protein from AF2 DB.
        iii. Generate model with AlphaFold 3 server and add manually.
        iv. Remove protein from the *data/seqs/*.fasta* file to remove from the dataset entirely.
    b. Error contains *proba_pair*:
        i. T-Coffee could not find any structures and uses a sequence-based alignment method. Check the location of your structures (*your_project/data/structures/*) and ensure that their filename matches the corresponding sequence identifiers.
        ii. T-Coffee found identical sequences with the same identifier. Rename or remove those from your dataset.
        iii. Use complete path to (*your_project/data/structure* directory (Note 12.c).
21. Set up advice on shared servers/HPCs:

a. Ensure that Nextflow and Apptainer are available and loaded.

b. We observed that SIMSApiper and the *data* directory need to share a common root folder e.g.:

   i. *nextflow run /opt/software/simsapiper/simsapiper.nf -profile server,withsingularity --data /home/user/simsatest/toy_example/data --magic* does not work.

   ii. *nextflow run /home/user/simsapiper/simsapiper.nf -profile server,withsingularity --data /home/user/simsatest/toy_example/data --magic* works.

c. Provide a shared Apptainer cache location *(--apptainerPath*) to avoid keeping many copies of the GB-sized apptainer images.

Acknowledgements

This work was supported by the Research Foundation Flanders (FWO) SB PhD fellowship [1SE5925N] to C.C.; the FWO large infrastructure grant [I000323N] to A.D. and the Fonds de la Recherche Scientifique (FNRS) Aspirant fellowship to S.L.H. The resources used in this work were provided in part by the VSC (Flemish Supercomputer Center), funded by the FWO and the Flemish Government.

References

1. O'Sullivan O, Suhre K, Abergel C, et al (2004) 3DCoffee: Combining Protein Sequences and Structures within Multiple Sequence Alignments. Journal of Molecular Biology 340:385–395
2. Carpentier M and Chomilier J (2019) Protein multiple alignments: sequence-based versus structure-based programs. Bioinformatics 35:3970–3980
3. Rajapaksa S, Konagurthu AS, and Lesk AM (2023) Sequence and structure alignments in post-AlphaFold era. Current Opinion in Structural Biology 79:102539
4. Chatzou M, Magis C, Chang J-M, et al (2016) Multiple sequence alignment modeling: methods and applications. Briefings in Bioinformatics 17:1009–1023

5. Varadi M, Anyango S, Deshpande M, et al (2022) AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. Nucleic Acids Research 50:D439–D444
6. Abramson J, Adler J, Dunger J, et al (2024) Accurate structure prediction of biomolecular interactions with AlphaFold 3. 630:493–500
7. Lin Z, Akin H, Rao R, et al (2023) Evolutionary-scale prediction of atomic-level protein structure with a language model. 379:1123–1130
8. Baltzis A, Mansouri L, Jin S, et al (2022) Highly significant improvement of protein sequence alignments with AlphaFold2. Bioinformatics 38:5007–5011
9. Rubio-Largo Á, Castelli M, Vanneschi L, et al (2018) A Parallel Multiobjective Metaheuristic for Multiple Sequence Alignment. J Comput Biol 25:1009–1022
10. Lladós J, Cores F, Guirado F, et al (2021) Accurate consistency-based MSA reducing the memory footprint. Computer Methods and Programs in Biomedicine 208:106237
11. Santus L, Garriga E, Deorowicz S, et al (2023) Towards the accurate alignment of over a million protein sequences: Current state of the art. Current Opinion in Structural Biology 80:102577
12. Crauwels C, Heidig S-L, Díaz A, et al (2024) Large-scale Structure-Informed multiple sequence alignment of proteins with SIMSApiper. Bioinformatics btae276
13. Di Tommaso P, Chatzou M, Floden EW, et al (2017) Nextflow enables reproducible computational workflows. Nat Biotechnol 35:316–319
14. Katoh K and Standley DM (2013) MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. Molecular Biology and Evolution 30:772–780
15. Taylor WR (2000) Protein structure comparison using SAP. Methods Mol Biol 143:19–32
16. Zhang Y and Skolnick J (2005) TM-align: a protein structure alignment algorithm based on the TM-score. Nucleic Acids Research 33:2302–2309

Figures

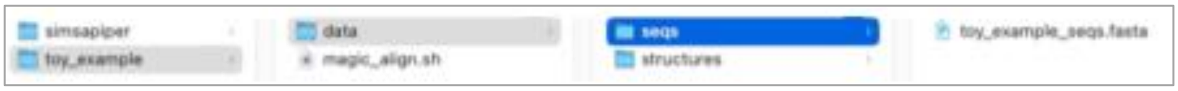

Figure 1: Recommended directory organization for an alignment project with SIMSApiper. The *toy_example* directory contains the launch file (.sh file) and the *data* directory, which includes subdirectories: *seqs* and *structures*. The *seqs* directory may contain one or more sequence files. The structures directory may contain 3D structure files in PDB format for none, some, or all sequences.

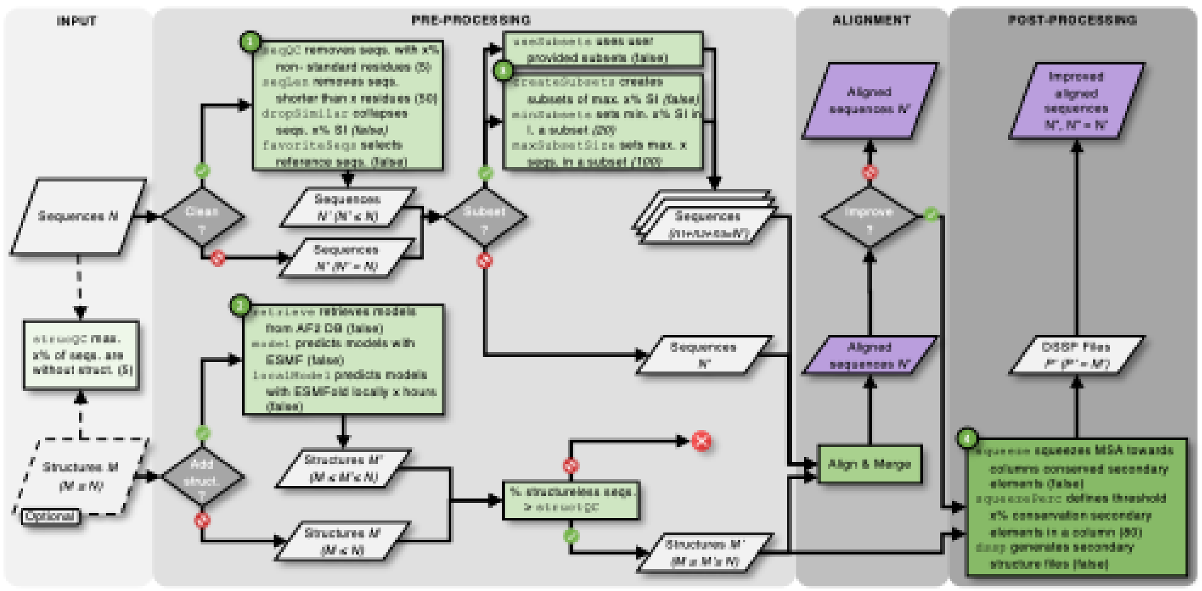
INPUT
PRE-PROCESSING
ALIGNMENT
POST-PROCESSING
Sequences N
Clean ?
seqQC removes seqs. with x% non- standard residues (5)
seqLen removes seqs. shorter than x residues (50)
dropSimilar collapses seqs. x% SI (false)
favoriteSeqs selects reference seqs. (false)
Sequences N' (N' < N)
Sequences N' (N' = N)
Subset ?
useSubsets uses user provided subsets (false)
createSubsets creates subsets of max. x% SI (false)
minSubsets sets min. x% SI in 1. a subset (20)
maxSubsetSize sets max. x seqs. in a subset (100)
Sequences (n1+n2+n3=N')
Sequences N'
structQC max. x% of seqs. are without struct. (5)
Structures M (M ≤ N)
Optional
Add struct. ?
retrieve retrieves models from AF2 DB (false)
model predicts models with ESMF (false)
localModel predicts models with ESMFold locally x hours (false)
Structures M' (M < M' ≤ N)
Structures M (M < N)
% structureless seqs. > structQC
Structures M' (M ≤ M' ≤ N)
Align & Merge
Aligned sequences N'
Improve ?
Aligned sequences N'
squeeze squeezes MSA towards columns conserved secondary elements (false)
squeezePerc defines threshold x% conservation secondary elements in a column (80)
dssp generates secondary structure files (false)
DSSP Files P' (P' = M')
Improved aligned sequences N'', N'' = N'

Figure 2: Flowchart to select the SIMSApiper's flags.

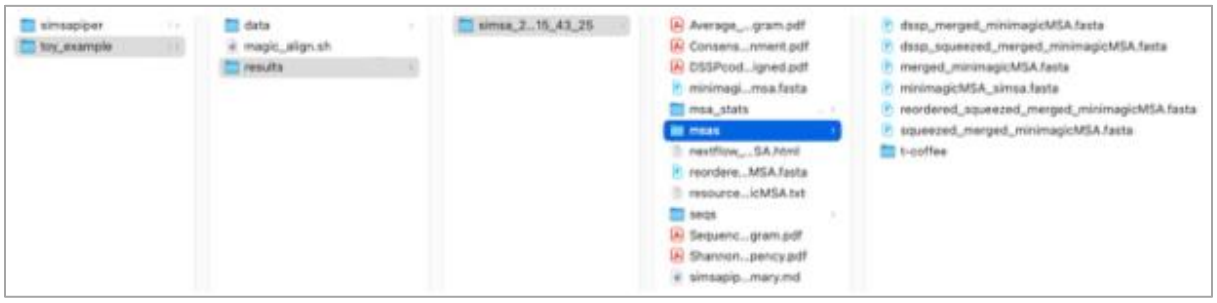

Figure 3: Example of the directory organization after running SIMSApiper. A results directory has been added and includes a directory per run. Such a directory includes all SIMSApiper's generated alignments and log files (Notes 16,18).

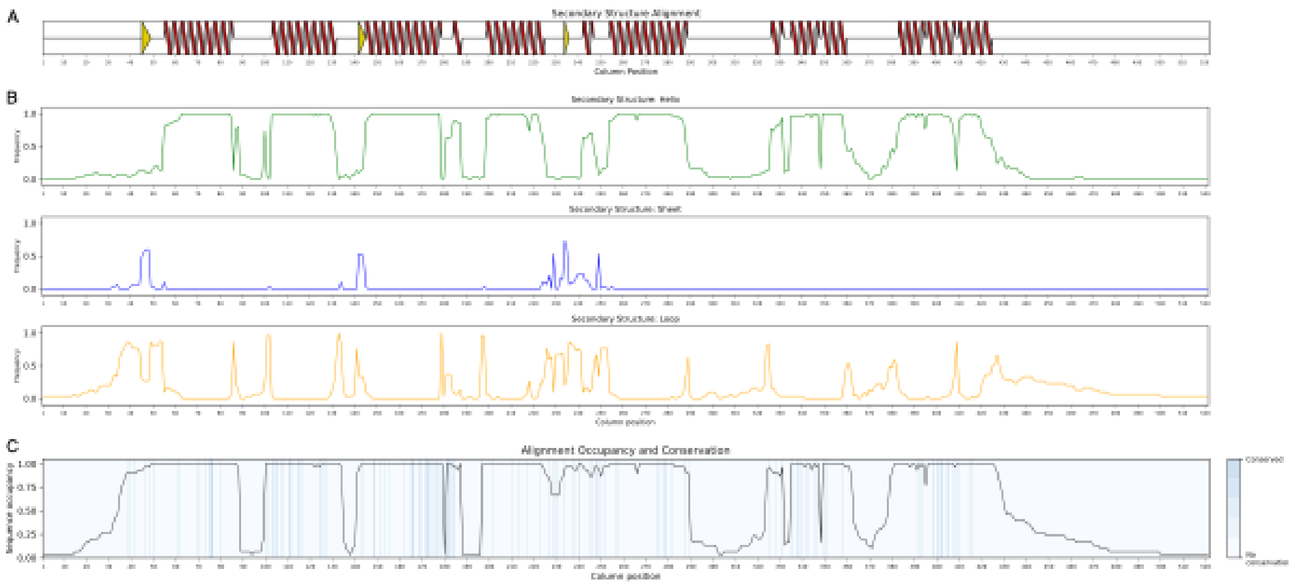

Figure 4: SIMSApiper's automatically generated plots to illustrate the obtained alignment. **A.** Consensus secondary structure plot. **B.** Distribution of the secondary structure elements along the alignment. **C.** Sequence conservation (Shannon entropy, heatmap) and sequence occupancy (occupancy of 1 for a given column means no gaps in that column, black line).

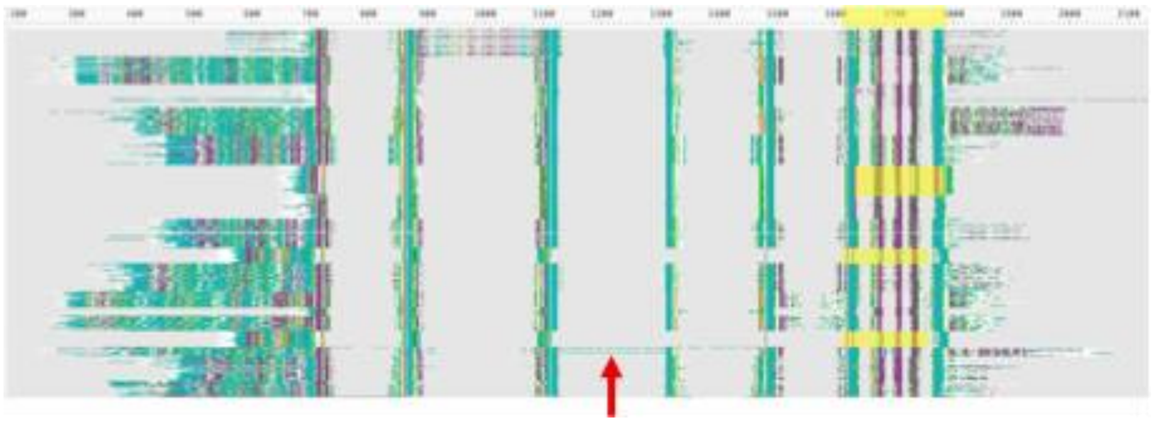

Figure 5: Secondary structure elements (Helix: green, Sheet: purple) mapped onto the MSA generated by SIMSApiper. Red arrow highlights sequences with long unalignable sections. Yellow bars highlight sequences with suboptimal alignment of highly conserved secondary structure region in positions 1600-1800. Visualized in AliView.

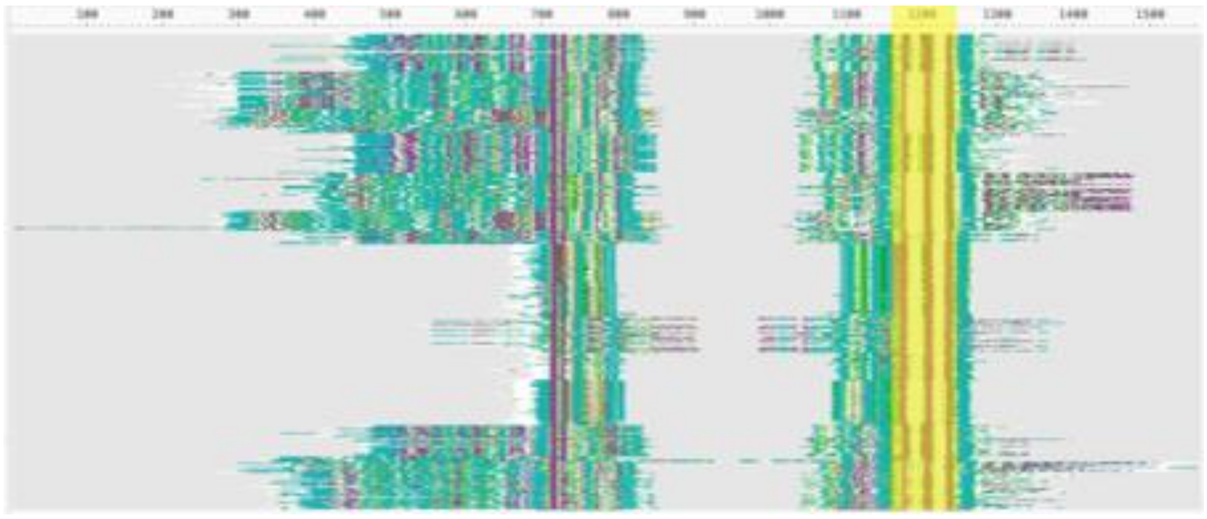

Figure 6: Secondary structure elements (Helix: green, Sheet: purple) mapped onto the MSA generated by SIMSApiper. Alignment of β-sheets in positions 1150-1250 (yellow highlight) now highly conserved across the whole MSA. Visualized in AliView.

Tables

Table 1: Selection helper describing the computational environment and dependency management assumed by the different example launch files, corresponding to local machines/laptops, servers and HPCs.

CPU cores / **Dependencies**	~8	~100	~1000
Docker	magic_local.sh	[custom profile recommended]	[custom profile recommended]
Apptainer	magic_apptainer.sh	magic_apptainer.sh	magic_hpc.sh (SLURM supported)

Table 2: Selection helper describing the computational environment and dependency management assumed by the different execution profiles, corresponding to local machines/laptops, servers and HPCs.

CPU cores / Dependencies	~8	~100	~1000
Local	-profile standard	-profile standard	[custom profile recommended]
Docker	-profile standard,withdocker	[custom profile recommended]	[custom profile recommended]
Apptainer	[custom profile recommended]	-profile server	-profile hpc (with SLURM)