MolTK

User's Manual

Release 0.4.0

November 4, 2011

Website: rotatingpenguin.com/moltk

Acknowledgments

Special thanks to Joy Ku who created the beautiful template for this manual. Thanks also to Christopher D. Putnam and Robert Edgar for helpful discussions about sequence alignment.

Table of Contents

List of Tables

List of Figures

List of Examples

1 Introduction to MolTK

1.1 Overview

1.1.1 What is MolTK?

There are three faces of MolTK; in order from easiest-to-use to most-powerful:

1. MolTK is a computer application that allows the user to view and align molecular sequences and structures.
2. Moltk is a Python programming language module that allows the user to align and otherwise compute on molecular sequences and structures.
3. Moltk is a C++ API that allow the programmer to compute with molecular sequences and structures with high efficiency.

1.1.2 Design Goals of MolTK

- Python programming language interactive environment for sequence/structure alignment that is just as easy to use as our (aging) dedicated alignment tool SEQUOIA.
- Flexible architecture that makes it easy to experiment with custom alignment methods and scoring systems.
- Consistent, well documented API for both Python and C++ programmers.
- Units-aware quantity type system. Thus alignment scores are not just numbers, they are information quantities with units of "bits". Atomic coordinates are not just numbers x, y, z, but are vector quantities with units of nanometers. Units-aware quantity types are an important part of scientific computing hygiene:
    - Converting a "quantity" to a raw number requires a "unit" to express the quantity in. This requires the user to pay attention to units at precisely the

moment when knowing the unit is most important. The rest of the time is "just works", even if you are wrong about what the current units are!
    - This sort of type safety in scientific computing might help prevent errors such as, say, crashing $100 million orbiters into planets.
    - Type safety: Adding a length to a volume makes no sense, and will result in an error. Dividing a length by a time results in a velocity.
- Next-generation molecular sequence/structure viewer that adheres to our user interface principles.

1.2 Installing MolTK

MolTK download site: http://code.google.com/p/moltk/downloads/list

1.2.1 Installing the MolTK Application

1.2.2 Installing the MolTK Python Module

1.2.3 Installing the MolTK C++ SDK

1.2.4 Building MolTK from Source Code

1.3 Software License

Commercial users should ask about our dual licensing model.

For questions contact: cmbruns@rotatingpenguin.com

1.4 Resources

Browse the MolTK home page at:
http://rotatingpenguin.com/moltk/

Download MolTK at:
http://code.google.com/p/moltk/downloads/list

Find questions and answers in the moltk-users forum at:
http://groups.google.com/group/moltk-users

Report issues and get the source code at:
http://code.google.com/p/moltk/

Study the MolTK Python API at:
http://www.rotatingpenguin.com/moltk/api_python/python_api.html

Study the MolTK C++ API at:
http://www.rotatingpenguin.com/moltk/api_cxx/cxx_api.html

2 The MolTK Application

3 Programming MolTK

3.1 Python

3.1.1 Tutorial

Download and install MolTK from the MolTK download site:
http://code.google.com/p/moltk/downloads/list

In the following examples text following the >>> prompt represents
commands typed by the user. Other lines represent the output of the program.

3.1.1.1 Scenario 1: Aligning 3 Sequences

In this example, the insulin_shark_v1.fasta, insulin_pig_v1.fasta, and insulin_human_v1.fasta are sequences in the FASTA format. Information about the FASTA format can be found at http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml. A file called test.fasta can be saved with the alignment. The alignment can be also saved as test.pretty which is a nicely formatted but not computer parsable version of the alignment. What follows is an example of performing an alignment of three sequences.

3.1.1.1.1 Step 1. Download FASTA Test Sequences

Go to download page http://code.google.com/p/moltk/downloads/list and download the three FASTA sequences: insulin_shark_v1.fasta, insulin_pig_v1.fasta, and insulin_human_v1.fasta.

3.1.1.1.2 Step 2. Start an Alignment Session

```
% python -i
>>> import moltk
```

3.1.1.1.3 Step 3. Load test1.fasta Sequence into Seq1

```
>>> seq1 = moltk.load_fasta("insulin_human_v1.fasta")
```

3.1.1.1.4 Step 4. Load test2.fasta Sequence into Seq2

```
>>> seq2 = moltk.load_fasta("insulin_pig_v1.fasta")
```

3.1.1.1.5 Step 5. Align Seq1 and Seq2

```
>>> align1 = moltk.align(seq1, seq2)
```

3.1.1.1.6 Step 6. Display the Alignment on the Screen

```
>>> print align1
```

3.1.1.1.7 Step 7. Load test3.fasta Sequence into Seq3

```
>>> seq3 = moltk.load_fasta("insulin_shark_v1.fasta")
```

3.1.1.1.8 Step 8. Align Previous Alignment of test1.fasta and test2.fasta with test3.fasta

```
>>> align2 = moltk.align(align1, seq3)
```

3.1.1.1.9 Step 9. Save the Alignment to a File

```
>>> align2.write_fasta("test.fasta")
```

You can also save as test.pretty for a nicely formatted but not computer parseable version.

```
>>> align2.write_pretty("test.pretty")
```

3.1.1.1.10 Step 10. Display a Table of Pairwise Sequence Identities

```
>>> print align2.id_table()
```

3.1.1.1.11 Step 11. End Your Alignment Session

```
>>> quit()
```

3.1.1.2 Scenario 2: Writing a Python Script to Help Automate the Alignment Process

3.1.1.3 Scenario 3: Aligning 2 Sequences and Assessing Significance

3.1.1.4 Scenario 4: Overlaying 2 tertiary structures

3.1.2 MolTK for SEQUOIA Users

Sequoia Command	**Corresponding MolTK Command(s)**	**Description**

`% sequoia`	`% python –i`
	`import moltk`
		Start an alignment session

`SEQUOIA> read SEQ1 test1.fasta`	`>>> seq1 = moltk.load_fasta("test1.fasta")`	Load a sequence or alignment into seq1
`SEQUOIA> read SEQ2 test2.fasta`	`>>> seq2 = moltk.load_fasta("test2.fasta")`	Load a sequence or alignment into seq2
`SEQUOIA> align`	`>>> align1 =  moltk.align(seq1, seq2)`	Align two sequences/alignments
`SEQUOIA> print ALIGN`	`>>> print align1`	Display alignment on screen
`SEQUOIA> set SEQ1 ALIGN`	`>>> seq1 = align1`	Copy alignment to seq1
`SEQUOIA> write ALIGN test.fasta`	`>>> align1.write_fasta("test.fasta")`	Save alignment to a file

Sequoia Command	**Corresponding MolTK Command(s)**	**Description**
`SEQUOIA> print ALIGN test.pretty`	`>>> align1.write_pretty(“test.pretty”)`	Save nicely formatted, but not computer parsable, version of alignment to a file
`SEQUOIA> print id ALIGN`	`>>> print align1.id_table()`	Display a table or pairwise sequence identities
`SEQUOIA> quit`	`>>> quit()`	End alignment session
`SEQUOIA> @test.inp`	`>>> execfile(“test.py”)`	Run a script file you wrote
`% sequoia < testp.inp`	`% python test.py`	Run a script you wrote directly from a command prompt
`SEQUOIA> COMMENT This is a comment`	`>>> # This is a comment`	Statement is ignored
`SEQUOIA> help <command>`	`>>> help(<command>)`	Access documentation about a command
`SEQUOIA> system <command>`	`>>> import os` `>>> os.system(“<commands>”)`	Issue an operating system command
`SEQUOIA> consensus <sigma>`		Create a consensus sequences with X’s at non-conserved positions
`SEQUOIA> optimize`		Improve an alignment by successively removing and realigning sequences
`SEQUOIA> print matrix`		Display the current scoring matrix
`SEQUOIA> read matrix test.mat`		Load a scoring matrix
`SEQUOIA> random 5`		Calculate 5 alignment scores with SEQ2 randomly shuffled
`SEQUOIA> split 5`		Remove sequence 5 from alignment and place it in seq1

Sequoia Command	**Corresponding MolTK Command(s)**	**Description**
`SEQUOIA> weight`		Apply sequence weights to alignment
`SEQUOIA> set epen 0.5`		Set gap extension penalty
`SEQUOIA> set gpen 10`		Set gap opening penalty
`SEQUOIA> set pretty_length 50`		Set sequence width of formatted alignments
`SEQUOIA> set random_seed 1`		Change seed for RANDOM command
`SEQUOIA> set suboptimal 0.1`		Parameter for allowing suboptimal alignments
`SEQUOIA> read STRUCT1 test1.pdb`		Load a structure into struct1
`SEQUOIA> salign`		Create a structure based alignment
`SEQUOIA> overlay`		Superpose two structures
`SEQUOIA> equivalence`		Assign equivalent residues between 2 structures
`SEQUOIA>set acutoff 45`		Max rotation difference for structure alignment in degrees
`SEQUOIA>set dcutoff 4.5`		Max distance difference for structure alignment in angstroms
`SEQUOIA> set runlength 4.5`		Min length required for a run of structurally equivalent residues
`SEQUOIA> set useangle 0`		Whether to use rotation similarity in structure alignment
`SEQUOIA> tabulate`	There is no tabulate command in MolTK	Populate pairwise residue scores before aligning (seldom used)

Sequoia Command	**Corresponding MolTK Command(s)**	**Description**
`SEQUOIA> stabulate`	There is no stabulate command in MolTK	Populate pairwise residue structure scores before aligning (seldom used)
`SEQUOIA> set echo 1`	Don't know how to do this in python	Output a copy of user's commands

3.2 C++

3.2.1 MolTK Coding Style Guidelines

Purpose: To provide a consistent look for both python and C++ MolTK code.

3.2.1.1 Indentation

Use four space characters per indent. No tabs.

3.2.1.2 Class names

Use CapitalizedWords

Use all caps for abbreviations, e.g. PDBStructure.

Class names should be nouns.

3.2.1.3 Macro, enum, and constant names

CAPITALIZED_WITH_UNDERSCORES

3.2.1.4 Method and function names

Use lower_case_with_underscores.

Begin method and function names with a verb.

3.2.1.5 Attribute and member names

lower_case_with_underscores

adjective_noun or noun

b_variable_name for boolean values

3.2.1.6 Namespace, package, and module names

lower case, single word, short names

3.2.1.7 Method parameters

lower_case_with_underscores

Remember that method parameter names are more important in python than in C++, because python allows named parameter use. In all C++ header files, every parameter to every exposed method should have an understandable name.

4 Bibliography

Dayhoff, M. O., 1969. *Atlas of Protein Sequence and Structure*. Silver Spring, MD: National Biomedical Research Foundation.

Edgar, R. C., 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research,* 32(5), pp. 1792-1797.

Gusfield, D., 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge: Cambridge University Press.

Henikoff, S. & Henikoff, J. G., 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences,* Volume 89, pp. 10915-10919.

Needleman, S. & Wunsch, C. D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology,* Volume 48, pp. 443-453.