|  |  |
| --- | --- |
| MolTK | |
|  | moltk1 |
|  | User’s Manual |
|  | Release 0.4.0  November 4, 2011 |
|  | Website: rotatingpenguin.com/moltk |

**Copyright and Permission Notice**

Copyright (c) 2011 Christopher M. Bruns

Contributors: Cami K. Bruns

Permission is hereby granted, free of charge, to any person obtaining a copy of this document (the "Document"), to deal in the Document without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Document, and to permit persons to whom the Document is furnished to do so, subject to the following conditions:

This copyright and permission notice shall be included in all copies or substantial portions of the Document.

THE DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS, CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENT OR THE USE OR OTHER DEALINGS IN THE DOCUMENT.

**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**

[1 Introduction to MolTK 1](#_Toc307944242)

[1.1 Overview 1](#_Toc307944243)

[1.1.1 What is MolTK? 1](#_Toc307944244)

[1.1.2 Design Goals of MolTK 1](#_Toc307944245)

[1.2 Installing MolTK 2](#_Toc307944246)

[1.2.1 Installing the MolTK Application 2](#_Toc307944247)

[1.2.2 Installing the MolTK Python Module 2](#_Toc307944248)

[1.2.3 Installing the MolTK C++ SDK 2](#_Toc307944249)

[1.2.4 Building MolTK from Source Code 2](#_Toc307944250)

[1.3 Software License 2](#_Toc307944251)

[1.4 Resources 3](#_Toc307944252)

[2 The MolTK Application 5](#_Toc307944253)

[3 Programming MolTK 7](#_Toc307944254)

[3.1 Python 7](#_Toc307944255)

[3.1.1 Tutorial 7](#_Toc307944256)

[3.1.2 MolTK for SEQUOIA Users 9](#_Toc307944257)

[3.2 C++ 12](#_Toc307944258)

[3.2.1 MolTK Coding Style Guidelines 12](#_Toc307944259)

[4 Bibliography 15](#_Toc307944260)

**List of Tables**

No table of figures entries found.

**List of Figures**

No table of figures entries found.

**List of Examples**

# 

# Introduction to MolTK

## Overview

### 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.

### 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.

## Installing MolTK

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

### Installing the MolTK Application

The MolTK application won’t be released until MolTK version 0.7.

### Installing the MolTK Python Module

#### Windows

##### Install 32-bit Python 2.7

http://www.python.org/download/

Python 2.7.2 Windows Installer (current version on November 5, 2011)

##### Put Python on your Path

Start -> Control Panel -> (System and Security) -> System -> Advanced System Settings ->

Environment Variables (in Advanced tab) -> Edit "Path" in System variables ->

Add ";C:\Python27" (without the quotation marks) onto the end of the current text in the

Variable value field -> Click OK

##### Install the MolTK Python module for windows

http://code.google.com/p/moltk/downloads/list

MolTK-0.4.1.win32-py2.7.msi (curren version on November 5, 2011)

#### Macintosh

TODO in MolTK version 0.6

Install Python 2.6

#### Linux

TODO in MolTK version 0.6

### Installing the MolTK C++ SDK

TODO in MolTK version 0.5

### Building MolTK from Source Code

Build from source if you cannot find a binary installer for your system.

#### Additional Prerequisites for Building MolTK from Source

* C++ compiler
  + Windows

Use Visual Studio 9 (2008), if you intend to use the standard python 2.7 install.

http://www.microsoft.com/express/downloads/

* + Mac

Use gcc.

You might need to install XCode developer tools to get gcc.

* + Linux

Use gcc.

* Boost, including iostreams and python libraries
  + Windows

Get it from http://www.boostpro.com/download/ to get prebuilt binary libraries.

Be sure to select the iostream and python libraries, and to select Visual Studio 9. It might take several tries for the install to complete successfully.

* + Mac

I like to build fat binaries supporting both 32-bit and 64-bit architectures. So I build boost like:

./bootstrap.sh --prefix=/usr/local

sudo ./b2 address-model=32\_64 architecture=x86 variant=release link=shared,static install

<or to include PowerPC, which Python still does>

sudo ./b2 address-model=32\_64 architecture=combined variant=release link=shared,static install

* CMake

<http://www.cmake.org>

### Extend MolTK/Regenerate Python Bindings

Only regenerate python bindings if you have changed the moltk API.

#### Additional Prerequisites for Regenerating Python Bindings

* gccxml

(you might need to install cvs first)

cvs -d :pserver:anonymous@www.gccxml.org:/cvsroot/GCC\_XML login

(respond with password gccxml)

cvs -d :pserver:anonymous@www.gccxml.org:/cvsroot/GCC\_XML co gccxml

(install using cmake)

(add location of gccxml.exe to your PATH)

(on windows: run gccxml\_vcconfig.bat from install area)

On Windows, install to a directory without any space characters, i.e. NOT c:/Program Files/whatever

* pygccxml

svn checkout:

https://pygccxml.svn.sourceforge.net/svnroot/pygccxml/pygccxml\_dev pygccxml

cd pygccxml

python setup.py build

python setup.py install

On windows, patch pygccxml/parser/source\_reader.py method

\_\_create\_command\_line to NOT wrap command with double quotes.

* pyplusplus

svn checkout:

https://pygccxml.svn.sourceforge.net/svnroot/pygccxml/pyplusplus\_dev pyplusplus

cd pyplusplus

python setup.py build

python setup.py install

#### Generate API documentation(optional)

* epydoc (for generating python API documentation)

http://sourceforge.net/projects/epydoc/files/epydoc/

* doxygen (C++ API documentation)
* dot (graphs and figures in API documentation)

## Software License

Copyright (C) 2011 Christopher M. Bruns

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Commercial users should ask about our dual licensing model.

For questions contact: cmbruns@rotatingpenguin.com

## 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

# The MolTK Application

# Programming MolTK

## Python

### 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.

#### 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.

##### 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.

##### Step 2. Start an Alignment Session

Open a command prompt window. Navigate to the location of your downloaded FASTA files. In this step “%” represents your command prompt.

% python -i

>>> **import** moltk

##### Step 3. Load test1.fasta Sequence into Seq1

>>> seq1 = moltk.load\_fasta**(**"insulin\_human\_v1.fasta"**)**

##### Step 4. Load test2.fasta Sequence into Seq2

>>> seq2 = moltk.load\_fasta**(**"insulin\_pig\_v1.fasta"**)**

##### Step 5. Align Seq1 and Seq2

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

##### Step 6. Display the Alignment on the Screen

>>> **print** align1

Note: Homo sapiens (human) and Sus scrofa (pig)

##### Step 7. Load test3.fasta Sequence into Seq3

>>> seq3 = moltk.load\_fasta**(**"insulin\_shark\_v1.fasta"**)**

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

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

##### 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"**)**

##### Step 10. Display a Table of Pairwise Sequence Identities

>>> **print** align2.id\_table**()**

##### Step 11. End Your Alignment Session

>>> quit**()**

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

#### Scenario 3: Aligning 2 Sequences and Assessing Significance

#### Scenario 4: Overlaying 2 tertiary structures

### 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> 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> 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> 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 |

## C++

### MolTK Coding Style Guidelines

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

#### Indentation

Use four space characters per indent. No tabs.

#### Class names

Use CapitalizedWords

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

Class names should be nouns.

#### Macro, enum, and constant names

CAPITALIZED\_WITH\_UNDERSCORES

#### Method and function names

Use lower\_case\_with\_underscores.

Begin method and function names with a verb.

#### Attribute and member names

lower\_case\_with\_underscores

adjective\_noun or noun

b\_variable\_name for boolean values

#### Namespace, package, and module names

lower case, single word, short names

#### 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.

# 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.