

TimeScapes

**A package for event detection and
activity monitoring of MD trajectories**

TimeScapes Version 1.2.2 / Document Version 0.2

October 2009

D. E. Shaw Research

Notice

The TimeScapes User's Guide and the information it contains is offered solely for educational purposes, as a service to users. It is subject to change without notice, as is the software it describes. D. E. Shaw Research assumes no responsibility or liability regarding the correctness or completeness of the information provided herein, nor for damages or loss suffered as a result of actions taken in accordance with said information.

No part of this guide may be reproduced, displayed, transmitted, or otherwise copied in any form without written authorization from D. E. Shaw Research.

The software described in this guide is copyrighted and licensed by D. E. Shaw Research under separate agreement. This software may be used only according to the terms and conditions of such agreement.

Copyright

© 2009 by D. E. Shaw Research. All rights reserved.

Trademarks

All trademarks are the property of their respective owners.

1. Overview

First, we recommend to obtain a copy of the paper [1] (appendix B) and its supporting information. These documents are complementary to the following instructions and are required for understanding the program workflow.

TimeScapes consists of the seven Python programs:

- `agility.py`: performs a sliding window RMS fluctuation and segmentation
- `contact.py`: extracts contact time series based on distance cutoff
- `expanse.py`: computes the alpha carbon RMS deviation
- `gmdhist.py`: computes Generalized Masked Delaunay (GMD) pair distribution histograms
- `gmdshow.py`: computes data for rendering a 3D GMD graph,
- `pearson.py`: performs Pearson correlation analysis of raw time series data
- `terrain.py`: performs event detection and activity monitoring

More details of the purpose, input, and output of each program are given on the following pages. These seven programs are supported by five library-style modules identified by the “mod” prefix. The programming style is mainly procedural and modular, to facilitate future development. Due to the emphasis on code-reuse and use of Python-specific nesting the code is quite compact and well readable.

The name of the support modules corresponds to their functionality:

- `mod_pio.py`: PDB input / output
- `mod_pwk.py`: utility routines for working with PDB structure files
- `mod_tio.py`: trajectory input / output
- `mod_twk.py`: utility routines for working with trajectory frame files
- `mod_gen.py`: generic utility routines, i.e. routines that are not structure or frame based

None of the Python scripts require user modification, though, as described below, TimeScapes supports creating new functionality.

The seven programs are invoked and all required information is specified using the UNIX shell command line:

```
$ python program.py arg1 arg2 ... argn
```

or

```
$ program.py arg1 arg2 ... argn
```

If invoked without arguments, abridged documentation will be printed on the standard output. This information can also be seen by scrolling down to the `__main__` function in the respective Python files.

For information on software required by TimeScapes and instructions on building, testing, and installing the software, consult the file README.txt in the distribution bundle. To report bugs or request help with using TimeScapes, send a message to TimeScapes@DEShawResearch.com.

2. Integration into the MD Workflow

Through the use of VMD molfile plugins [2], TimeScapes supports a variety of trajectory formats including DCD (CHARMM, NAMD, XPLOR), DTR, DTRV (Desmond), LAMMPSTRJ (LAMMPS), NetCDF (Amber) and TRJ, TRR, XTC (Gromacs). Note: the Amber CRD format is not supported, please use NetCDF instead.

TimeScapes provides no trajectory editing and accepts coordinates at face value. Therefore, it is expected that the user has access to a trajectory management tool for such basic editing tasks as stride (time-step) modification, least-squares alignment of trajectory frames, unwrapping of coordinates (in case of periodic boundaries), etc. Due to the use of relative distance geometry by most of the programs in TimeScapes, an alignment of frames is not required. The one exception is for `agility.py`, whose algorithm makes use of Cartesian coordinates, and where alignment is provided as an option to the user. See the following documentation for more details.

Before using the TimeScapes programs, it is necessary in the case of periodic boundaries to ensure that coordinates in the trajectory frames are unwrapped. That is, make sure that the unwrapping, gluing, and tagging in your mapping tool perform as intended by inspection of the results with your graphics program (there should be no atoms wrapped around the periodic box). In addition, it often makes sense to strip solvent and other unwanted atoms from the trajectory frames beforehand, and to sample the trajectory to the desired stride since the stride cannot be modified from within TimeScapes. Since the algorithms in TimeScapes have a filtering and smoothing effect, setting an optimal stride is not critical: it mainly affects computational efficiency. But make sure there are sufficient sampling steps within the time window defined by the smoothing parameter `delta` (see the discussion of `agility.py` and `terrain.py`).

Finally, the user must prepare a corresponding PDB file from which TimeScapes programs obtain atom masses and the coarse side chain model. The PDB file should be carefully inspected for non-standard amino acid or atom names, especially at the N and C termini. The residue and atom name convention is most important for the tools `contact.py`, `gmdhist.py`, and `terrain.py` that require a coarse side chain model. Appendix C gives residue and atom names, taken from the CHARMM, Amber, and GROMACS force field parameter sets, that are recognized for representative side chain atom selection.

A warning will be issued if the number of representative side chain atoms in the coarse model does not match the number of alpha Carbon atoms. If this is the case, the easiest remedy is to edit the PDB file to conform to the above residue and atom names. If desired, the template function `mod_pwk_side` (in `mod_pwk.py`) can also be duplicated and edited to create a specialized coarse model specific to your system. See the modi-

fied function `mod_pwk_side_villin` as an example. To facilitate such modifications, the user may specify the name of a newly created coarse graining function as an optional argument to certain TimeScapes programs (see below).

3. User Guide

agility.py - sliding window RMS fluctuation and segmentation

Purpose

Performs an RMS fluctuation calculation in a Gaussian weighted sliding window and performs a time-dependent segmentation of the RMS fluctuation into “basins” and “transitions” for a temporal coarse-graining of the trajectory.

Usage

```
$ agility.py infile1 infile2 delta outname [-lsq]
```

Input (4 or 5 arguments)

- `infile1`: PDB file for mass assignment. Used also in the optional least-squares fit
- `infile2`: Trajectory file, for supported formats see Appendix C
- `delta`: Full Width at Half Maximum (FWHM) of Gaussian weighting within the sliding window in trajectory timestep units ($\text{FWHM} = 2 \sqrt{2 \ln 2} \text{ sigma}$). This parameter is important as it sets the desired time scale of analysis, and it is dependent on the problem. If you have no specific idea you can start with e.g. 5% of the total frame number and adjust up or down as desired
- `outname`: User-defined basename prefix for output file names
- `[-lsq]`: Option for least-squares fitting of trajectory frames to PDB

Output

- `outname_segmentation.dat`: Raw data file with frame number, RMS fluctuations, first derivative, basin label (for plotting purposes)
- `outname_transitions.dcd`: DCD trajectory file with frames of basin transitions
- `outname_transitions.log`: Corresponding log file with program parameters and frame information
- `outname_minima.dcd`: DCD trajectory file with frames of basin minima
- `outname_minima.log`: Corresponding log file with program parameters and frame information

Caveats and Notes

- Trajectory coordinates are taken at face value and must be unwrapped

- No atom selection mechanism is provided; mass-weighted fluctuations are computed for all atoms in the system. Eliminate all the atoms you don't want included beforehand
- Least-squares fitting is slow; pre-process trajectory and turn lsq off, if possible
- Log files contain detailed parameters
- If the program runs slowly or runs out of memory, try running with fewer frames or eliminate unwanted atoms

contact.py - extracting contact time series based on distance cutoff

Purpose

Facilitates plotting of distances between representative atoms in the coarse model as a function of simulation timestep

Usage

```
$ contact.py infile1 infile2 cut outname [csel]
```

Input (4 or 5 arguments)

- *infile1*: PDB file used for coarse model assignment
- *infile2*: Trajectory file, for supported formats see Section 2
- *cut*: Selection distance cutoff (in Å) A particular contact is selected if the separation of representative atoms falls to at or below the cut level in any trajectory frame
- *outname*: User-defined basename prefix for output file names
- [*csel*]: Optional user-provided coarse-graining function defined in *mod_pwk.py* (the default is *mod_pwk_side*)

Output

- *outname_raw.dat*: Raw contact distance time series stored column-wise
- *outname_contacts.log*: Representative atom details row-wise for each contact; the contacts are ordered by residue number of the second participating residue

Caveat and Note

- Trajectory coordinates are taken at face value and must be unwrapped
- Based on relative distance geometry so no alignment of frames is required

expanse.py - computing the alpha Carbon RMS deviation

Purpose

Facilitates plotting of the alpha Carbon (CA) RMSD from a PDB as a function of simulation timestep

Usage

```
$ expanse.py infile1 infile2 outfile
```

Input (3 arguments)

- `infile1`: PDB file for alpha carbon assignment and for the least-squares fit
- `infile2`: Trajectory file, for supported formats see Section 2
- `outfile`: Name of output file for CA RMSD time series

Caveat and Note

- Trajectory coordinates are taken at face value and must be unwrapped
- Based on relative distance geometry so no alignment of frames is required

gmdhist.py - computing GMD pair distribution histograms

Purpose

Facilitates plotting of pair distance distribution histograms for the evaluation of GMD orders

Usage

```
$ gmdhist.py infile1 infile2 nb cut m outname [csel [msel]]
```

Input (6, 7, or 8 arguments)

- `infile1`: PDB file used for coarse model assignment
- `infile2`: Trajectory file, for supported formats see Section 2
- `nb`: Number of histogram bins
- `cut`: Maximum distance in Å, e.g. 30
- `m`: Maximum GMD order > 1
- `outname`: User-defined basename prefix for output file names
- `[csel]`: Optional user-provided coarse-graining function defined in `mod_pwk.py` (the default `mod_pwk_side`)
- `[msel]`: If 'csel' is defined, optional user-provided mask sampling selection for GMD defined in `mod_pwk.py` (the default is `mod_pwk_all`)

Output

- *outname_hist1.dat*: Full pair distribution of coarse model
- *outname_hist2.dat*: Order-2 GMD pair distribution
- *outname_hist3.dat*: Order-3 GMD pair distribution
- ...
- *outname_histm.dat*: Order-m GMD pair distribution

Caveats and Notes

- Trajectory coordinates are taken at face value and must be unwrapped
- Based on relative distance geometry, so no alignment of frames is required
- If 7 arguments are given, the program assumes that the last argument is *csel*
- You can specify your own coarse graining and masking functions via the *csel* and *msel* parameters. The intended use of this functionality is mainly to select a region of interest for the analysis. It is recommended that the *csel* function picks one representative atom per side chain in the region of interest, whereas the *msel* function should sample the masking region at full atomic detail

gmdshow.py - rendering a 3D GMD graph

Purpose

Facilitates visualization of a GMD graph with the molecular graphics program VMD

Usage

```
$ gmdshow.py infile order outfile [csel [msel]]
```

Input (3, 4, or 5 arguments)

- *infile*: PDB file used for GMD calculation
- *order*: GMD order > 1
- *outfile*: filename of VMD-sourcable Tcl script that will contain graph connectivity
- [*csel*]: Optional user-provided coarse-graining function defined in *mod_pwk.py* (the default is *mod_pwk_side*)
- [*msel*]: If *csel* is defined, optional user-provided mask sampling selection for GMD defined in *mod_pwk.py* (the default is *mod_pwk_all*)

Caveats and Note

- PDB coordinates are taken at face value and must be unwrapped

- If 4 arguments are given, the program assumes the last argument is `csel`
- You can specify your own coarse graining and masking functions via the `csel` and `msel` parameters. The intended use of this functionality is mainly to select a region of interest for the analysis. It is recommended that the `csel` function picks one representative atom per side chain in the region of interest, whereas the `msel` function should sample the masking region at full atomic detail

pearson.py - Pearson correlation analysis of data columns

Purpose

Statistical correlation analysis of raw time series data

Usage

```
$ pearson.py infile1 colnr1 infile2 colnr2
```

Input (4 arguments)

- `infile1`: White space delimited time series data in column format
- `colnr1`: Column number that will be analyzed (counting from 1)
- `infile2`: White space delimited time series data in column format
- `colnr2`: Column number that will be analyzed (counting from 1)

Output

- Statistical variables in command window

terrain.py - event detection and activity monitoring

Purpose

Performs a detailed event and activity analysis and in addition performs a time-dependent segmentation of the total activity into “basins” and “transitions” for a temporal coarse-graining of the trajectory.

Usage

```
$ terrain.py infile1 infile2 cut1 cut2 \  
    delta gtype outname [csel [msel]]
```

Input (7, 8, or 9 arguments)

- `infile1`: PDB file used for coarse model assignment
- `infile2`: Trajectory file, for supported formats see Section 2

- `cut1`: Inclusive upper bound of contact. Values up to `cut1` are considered contacts in the recrossing filter. Recommended values are 6.0-7.5 (Å) for Cutoff graphs, and 2 for GMD graphs
- `cut2`: Inclusive upper bound of crossing buffer. Values larger than `cut1` up to `cut2` define the buffer zone in the recrossing filter. Recommended values are 7.0-8.5 (Å) for Cutoff graphs, and 3 for GMD graphs
- `delta`: Smoothing parameter in discrete trajectory timestep units (window half width or Gaussian kernel FWHM); this parameter is important as it sets the desired time scale of the filtering and activity analysis, and it is dependent on the problem. If you have no specific idea you can start with e.g. 5% of the total frame number and adjust up or down as desired
- `gtype`: User-selected graph type, either `GMD` or `Cutoff`
- `outname`: User-defined basename prefix for output file names
- `[cse1]`: Optional user-provided coarse-graining function defined in `mod_pwk.py` (the default is `mod_pwk_side`)
- `[mse1]`: For GMD graphs only, if `cse1` is defined, optional user-provided mask sampling selection for GMD defined in `mod_pwk.py` (the default is `mod_pwk_all`)

Output

- `outname_graphs`: Directory containing VMD-sourceable Tcl scripts with graph connectivity for each frame
- `outname_events.log`: Log file of individual contact forming or breaking events
- `outname_events.dcd`: DCD trajectory containing only event frames
- `outname_activity.dat`: Time series of total, forming, and breaking activity
- `outname_segmentation.dat`: Data file with frame number, total activity, first derivative, basin label
- `outname_transitions.dcd`: DCD trajectory file with frames of basin transitions
- `outname_transitions.log`: Corresponding log file with program parameters and frame information
- `outname_minima.dcd`: DCD trajectory file with frames of basin minima
- `outname_minima.log`: Corresponding log file with program parameters and frame information

Caveats and Notes

- End effects: There may be a surplus of breaking contacts at the end of a trajectory due to the termination of the recrossing filter

- Trajectory coordinates are taken at face value and must be unwrapped
- If 8 arguments are given, the program assumes the last argument is `csel`
- You can specify your own coarse graining and masking functions via the `csel` and `mset` parameters. The intended use of this functionality is mainly to select a region of interest for the analysis. It is recommended that the `csel` function picks one representative atom per side chain in the region of interest, whereas the `mset` function should sample the masking region at full atomic detail
- Log files contain filenames and detailed parameter summaries
- If the program runs slowly or runs out of memory try first on `Cutoff` graphs and / or try fewer frames
- It is also a good practice to fine tune parameters first on faster `Cutoff` graphs before selecting the slower GMD

4. Usage Ideas and Examples

The following brief tutorials provide usage ideas and workflow examples.

Evaluating contacts, events, and activities

The paper [1] provides the best reference for the originally intended application of event detection and activity monitoring. The activity curves are most useful for plotting figures, whereas the event logs (see Supplementary Materials and Methods of the paper) give detailed time-dependent information on the significant contact changes in the structure. The workflow of this application is as follows. The RMS deviations in Figures 1, 7, and 8 were created with `expanse.py`. The 3D GMD graphs in Figure 3 were created with `gmdshow.py`. The histograms in Figure 4 were created with `gmdhist.py`. The contact time series in Figure 5 was extracted with `contact.py`. The RMS fluctuations in Figures 7 and 8 were created with `agility.py`. The Cutoff and GMD activities in Figures 7, 8, and 9 were created with `terrain.py`. The correlation values between the curves (discussed in [1]) were computed with `pearson.py`.

Time-dependent segmentation (clustering)

The idea of segmenting the trajectory into basins and their constituent minima and transitions was also described in the paper [1]. We found this functionality to be useful in applications where a meaningful temporal coarse-graining of the trajectory is required. For example, certain clustering algorithms require a full all-to-all comparison of frames. It makes sense for such applications to reduce the computational complexity by picking only meaningful frames, e.g. those corresponding to the basin minima. Although this is not a true clustering in the sense that the temporal sequence of the trajectory is retained in the ordering of the minima, the saved minima could be used as seeds for a full clustering with a separate program. TimeScapes supports such a “segmentation” of the trajectory in the tools `agility.py` (for Cartesian RMS fluctuations) and `terrain.py` (Cutoff and GMD graphs). In both of these programs minima are saved as DCD trajectory files.

Variable stride time compression

The output DCD trajectory “events.dcd” contains frames where “something happens” in the trajectory. Compared to trajectories with fixed stride access pattern, this events trajectory is able to bridge between a wider range of time scales: the system time is compressed during times of inactivity, but it is stretched during times of detected conformational changes. Variable stride movie animations will appear more vibrant and

active compared to those with a fixed stride, and they emphasize particular changes a user may be interested in. For example, in `terrain.py` (Cutoff and GMD graphs) a user may specify their own coarse graining and masking functions via the `cse1` and `mse1` arguments to focus on a region of interest. The events trajectory will then record changes within this region only. Thereby, a user can visualize fast processes in this region of interest against the background of overall slower change in the global structure. Note that this trajectory may contain more frames than the original trajectory due to duplication in the case of multiple events per frame, so you may want to compress your dynamics sufficiently by selecting a long smoothing parameter `delta` and/or a large recrossing buffer.

A. Version History:

- A. TimeScapes 1.0 was based on the `generictrajectory` trajectory access library and was used for the Figures and plots described in the paper [1]. Due to the reliance on parts of Desmond it was designed mainly for in-house use at D. E. Shaw Research.
- B. TimeScapes 1.1 was the first version based on the free `molfile` plugin library. It was designed to give results consistent with version 1.0 but it was not optimized for memory use or efficiency. As an intermediate version it was designed mainly for in-house use at D. E. Shaw Research.
- C. TimeScapes 1.2, based on the `molfile` plugin library, was optimized for memory and efficiency. Also, `terrain.py` was converted to distance geometry to eliminate the need for trajectory frame alignment and to conform to sequential processing of frames required by `molfile`. This means that the median filter is applied after extracting the distance time series (in versions 1.0 and 1.1 that order of median filtering and distance calculation is reversed). For these reasons, the output of `terrain.py` is similar, but not identical, to that of versions 1.0 and 1.1.

B. References

- [1] Willy Wriggers, Kate A. Stafford, Yibing Shan, Stefano Piana, Paul Maragakis, Kresten Lindorff-Larsen, Patrick J. Miller, Justin Gullingsrud, Charles A. Rendleman, Michael P. Eastwood, Ron O. Dror, and David E. Shaw, "Automated Event Detection and Activity Monitoring in Long Molecular Dynamics Simulations," *J. Chem. Theory Comput.*, 2009, 5 (10), pp 2595–2605, DOI: 10.1021/ct900229u, <http://pubs.acs.org/action/showLargeCover?issue=346432824>

Supporting information:

<http://pubs.acs.org/doi/suppl/10.1021/ct900229u>

- [2] Molfile Plugin Documentation, Theoretical and Computational Biophysics Group, University of Illinois at Urbana Champaign, <http://www.ks.uiuc.edu/Research/vmd/plugins/molfile/>

C. PDB residue names

PDB residue name	comment	representative atom
ALA	alanine	CA
ARG	arginine	CZ
ARGN	GROMACS deprotonated arginine	CZ
ASP	aspartate	CG
ASPH	GROMACS protonated aspartate	CG
ASPP	CHARMM protonated aspartate	CG
ASH	AMBER protonated aspartate	CG
ASN	asparagine	CG
ASN1	GROMACS alternate asparagine	CG
CYS	cysteine	CB
CYM	AMBER protonated cysteine	CB
CYSH	GROMACS protonated cysteine	CB
CYN	AMBER cysteine not protonated	CB
CYX	AMBER disulfide bonded cysteine	CB
CYS1	GROMACS cysteine	CB
CYS2	GROMACS cysteine	CB
GLN	glutamine	CD
GLU	glutamate	CD
GLH	AMBER protonated glutamate	CD
GLUH	GROMACS protonated glutamate	CD
GLUP	CHARMM protonated glutamate	CD
GLY	glycine	CA
HIS	histidine	CG
HIP	AMBER doubly protonated histidine	CG
HIE	AMBER epsilon-2 protonated histidine	CG
HID	AMBER delta-1 protonated protonated histidine	CG
HISH	GROMACS doubly protonated histidine	CG
HISB	GROMACS epsilon-2 protonated histidine	CG
HIS1	GROMACS epsilon-2 protonated histidine	CG
HISA	GROMACS delta-1 protonated histidine	CG
HISP	CHARMM doubly protonated histidine	CG
HISE	CHARMM epsilon-2 protonated histidine	CG
HISD	CHARMM delta-1 protonated histidine	CG
HSC	CHARMM doubly protonated histidine	CG
HSP	CHARMM doubly protonated histidine	CG
HSE	CHARMM epsilon-2 protonated histidine	CG
HS2	CHARMM epsilon-2 protonated histidine	CG
HSD	CHARMM delta-1 protonated histidine	CG
ILE	isoleucine	CG1

LEU	leucine	CG
LYS	lysine	CE
LYN	AMBER lysine neutral (not protonated)	CE
LSN	CHARMM lysine neutral (not protonated)	CE
LYP	AMBER lysine protonated	CE
LYSH	GROMACS lysine protonated	CE
MET	methionine	SD
PHE	phenylalanine	CG
PHEU	GROMACS alternate phenylalanine	CG
PRO	proline	CG
SER	serine	CB
THR	threonine	CB
TRP	tryptophane	CE2
TRPU	GROMACS alternate tryptophane	CE2
TYR	tyrosine	CG
TYRU	GROMACS alternate tyrosine	CG
VAL	valine	CB

D. Licenses

TimeScapes

TIMESCAPES LICENSE AGREEMENT

1. License Grant. Subject to the terms and conditions of this license agreement (the "Agreement"), D. E. Shaw Research, LLC ("DESRES") grants to LICENSEE a limited, royalty-free license, on a non-exclusive, non-transferable, non-assignable, and non-sublicensable basis, to install and use the analysis package for biomolecular simulations known as TimeScapes Version 1 (including any subsequent version of such program whose version number begins with "1.") and any associated documentation (any such documentation and any such version collectively referred to herein as the "SOFTWARE"). The SOFTWARE may be accessed, held, or otherwise used only with a valid license. Any other parties (including, without limitation, any collaborators of LICENSEE) wishing to install or use the SOFTWARE may do so only if such parties have executed a separate license agreement with DESRES giving such parties the right to do so. DESRES reserves all rights not expressly granted herein.
2. Representations and Warranties. LICENSEE hereby represents and warrants that:
 - a. LICENSEE has the necessary authority to enter into this Agreement;
 - b. all information that LICENSEE has provided or will hereafter provide in connection with this Agreement is and will be correct and complete;
 - c. LICENSEE will abide by the terms and conditions set forth in this Agreement.
3. Restrictions. LICENSEE may make copies of the SOFTWARE only as necessary for bona fide backup or archival purposes. LICENSEE shall not: (a) modify, translate, adapt, create derivative works from (except in accordance with the Derivative Work Permissions set forth in this paragraph), or decompile the SOFTWARE, or any portion thereof, or create or attempt to create, by reverse engineering or otherwise, the source code ("Source Code") from the object code supplied hereunder; (b) rent, lease, loan, sell, transfer, publish, display, or distribute the SOFTWARE, or make the SOFTWARE available to third parties, or use the SOFTWARE, or any portion thereof, in a service bureau, time-sharing, or outsourcing service, or otherwise use the SOFTWARE for the benefit of third parties; (c) remove or alter any proprietary rights notices on the SOFTWARE; (d) export, import, or re-export the SOFTWARE in violation of any applicable law, rule, or regulation of any jurisdiction; (e) disclose, without DESRES's prior written approval, the SOFTWARE or any code, information, or materials contained in or related to the SOFTWARE ("RELATED MATERIALS") other than as expressly authorized hereunder. LICENSEE shall notify DESRES immediately of any actual or imminent unauthorized access to, or use or disclosure of, the SOFTWARE and/or any RELATED MATERIALS. LICENSEE recognizes that the unauthorized use or disclosure of any of the foregoing will give rise to irrepara-

ble injury to DESRES, its affiliates, and/or its licensors for which monetary damages may be an inadequate remedy; and LICENSEE agrees that DESRES, its affiliates, and/or its licensors may seek and obtain injunctive relief against the breach or threatened breach of LICENSEE's obligations hereunder, in addition to any other legal and equitable remedies which may be available. The "Derivative Work Permissions" relate only to any Source Code provided by DESRES to LICENSEE and permit LICENSEE to create only the following types of derivative works: (i) any complementary code that interoperates with the SOFTWARE, provided that any such code is provided to users free of charge and distributed only with a disclaimer that conspicuously states that D. E. Shaw Research, LLC and its affiliates did not create, approve, or authorize such code, and (ii) any modification to the code comprising the SOFTWARE itself ("Software Modification"), provided that any such Software Modification may in no case be distributed by the LICENSEE.

4. **Acknowledgement and Citation.** LICENSEE agrees to acknowledge the use of the SOFTWARE in any reports or publications of results obtained with the SOFTWARE as follows:

"TimeScapes Analysis Package, version 1.X, D. E. Shaw Research, New York, NY, 2009."

Where 'X' is to be replaced with the minor-release number of the version used in the published research. If the published research is based on results obtained with any Software Modification or any complementary code not developed by DESRES, then those variants must be acknowledged as such. LICENSEE is also requested to include a citation to the following paper:

Willy Wriggers, Kate A. Stafford, Yibing Shan, Stefano Piana, Paul Maragakis, Kresten Lindorff-Larsen, Patrick J. Miller, Justin Gullingsrud, Charles A. Rendleman, Michael P. Eastwood, Ron O. Dror, and David E. Shaw, "Automated Event Detection and Activity Monitoring in Long Molecular Dynamics Simulations," *J. Chem. Theory Comput.*, 2009, 5 (10), pp 2595–2605, DOI: 10.1021/ct900229u, <http://pubs.acs.org/action/showLargeCover?issue=346432824>

5. **Disclaimer of Warranties and Liabilities.** LICENSEE acknowledges that the SOFTWARE is a research tool. The SOFTWARE is provided "as is." For the avoidance of doubt, DESRES and its licensors shall have no maintenance, upgrade, or support obligations with respect to the SOFTWARE. **DESRES, ITS AFFILIATES, AND ITS LICENSORS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, OR THAT THE SOFTWARE WILL OPERATE UNINTERRUPTED OR ERROR-FREE OR MEET LICENSEE'S PARTICULAR REQUIREMENTS. LICENSEE AGREES THAT DESRES AND ITS AFFILIATES SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, EXEMPLARY, PUNITIVE, OR INCIDENTAL DAMAGES WITH RESPECT TO ANY CLAIM BY LICENSEE OR ANY THIRD PARTY ARISING OUT OF OR RELATING TO THIS AGREEMENT OR USE OF THE SOFTWARE OR ANY DERIVATIVE WORK BASED ON THE SOFTWARE.**

6. Ownership Rights. LICENSEE acknowledges that the SOFTWARE is the sole and exclusive property of, and is valuable, confidential, and proprietary to, DESRES and its licensors, including, without limitation, all rights to patents, copyrights, trademarks, trade secrets, and any other intellectual property and proprietary rights inherent therein or appurtenant thereto, in all media now known or hereinafter developed, and LICENSEE shall protect the foregoing to at least the same extent that it protects its own confidential information, but using no less than a reasonable standard of care. LICENSEE is not purchasing title to the SOFTWARE or copies thereof, but rather is being granted only a limited license to use the SOFTWARE only in accordance with this Agreement. LICENSEE shall not use DESRES or its affiliates or licensors' names or marks or employee names, or adaptations thereof, in any advertising, promotional, sales, or other materials without the prior written consent of DESRES or, if and as applicable, of DESRES's affiliates or licensors. LICENSEE shall inform DESRES promptly in writing of any actual or alleged infringement of DESRES or its licensors' rights and of any available evidence thereof.
7. Term and Termination. LICENSEE's license with respect to the SOFTWARE shall be perpetual, subject to DESRES's rights to terminate this Agreement. Any and all rights granted to LICENSEE hereunder shall terminate immediately upon LICENSEE's breach of, or non-compliance with, any provisions of this Agreement. In the event of any termination of this Agreement for any reason, LICENSEE shall discontinue all use of the SOFTWARE and shall either (a) promptly return all copies of the SOFTWARE and any RELATED MATERIALS to DESRES, or (b) subject to DESRES's prior consent, provide DESRES with a certificate of destruction of all copies of the SOFTWARE and any RELATED MATERIALS. Notwithstanding the foregoing, only Paragraph 1 of this Agreement shall not survive the termination of this Agreement.
8. Government Use. The SOFTWARE and the accompanying documentation are "commercial items" as that term is defined in 48 C.F.R. 2.101 consisting of "commercial computer software" and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212. Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4, if LICENSEE hereunder is the U.S. Government or any agency or department thereof, the SOFTWARE and the documentation are licensed hereunder (i) only as commercial items, and (ii) with only those rights as granted to all other end users pursuant to the terms and conditions hereof.
9. General. This Agreement and its enforcement shall be governed by, and construed in accordance with, the laws of the State of New York, without regard to conflicts-of-law principles. LICENSEE acknowledges that (x) DESRES may enter into agreements with one or more third parties (each an "Independent Distributor") to distribute the SOFTWARE; and (y) any such Independent Distributor is a third-party beneficiary of this Agreement. The exclusive venue for any action relating to this Agreement shall be the state and federal courts situated in the State of New York, County of New York, and each party expressly consents to the jurisdiction of such courts. This Agreement constitutes the entire agreement between the parties and supersedes all prior agreements, written or oral, relating to the subject matter hereof. This Agreement may not be modified or

altered except by written instrument duly executed by both parties. If any provision of this Agreement is deemed to be unenforceable, that provision shall be enforced to the maximum extent permitted to effect the parties' intentions hereunder, and the remainder of this Agreement shall continue in full force and effect. The failure of either party to exercise any right provided for herein shall not be deemed a waiver of any right hereunder.

Additional Licenses

Portions of the enclosed software are made available under separate terms specified by the owners of that software.

Molfile

University of Illinois Open Source License
Copyright 2003 Theoretical and Computational Biophysics Group,
All rights reserved.

Developed by: Theoretical and Computational Biophysics Group
 University of Illinois at Urbana-Champaign
 <http://www.ks.uiuc.edu/>

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

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.

Neither the names of Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign, nor the names of its contributors may be used to endorse or promote products derived from this Software without specific prior written permission.

THE SOFTWARE 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 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 SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

Other Software

Included in molfile are three files that claim additional copyright over that in the above text:

ReadPARM7.h and ReadParm.h have the following notice:

* COPYRIGHT 1992, REGENTS OF THE UNIVERSITY OF CALIFORNIA

hoomdplugin.c contains the following notice:

* Copyright (c) 2009 Axel Kohlmeyer <akohlme@cmm.chem.upenn.edu>