TimeScapes

Multi-scale analytics package for MD trajectories

TimeScapes Version 1.5 / Document Version 0.6

May 2017

biomachina.org

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 described software. Biomachina.org 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.

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

Copyright

Trademarks

All trademarks are the property of their respective owners.

1. Overview

We recommend to obtain copies of the papers [1,2,4] and any supporting information as listed in appendix B. These documents are complementary to the following instructions and are required for understanding the program workflow.

TimeScapes consists of ten Python programs**:**

Seven original programs for event detection and activity monitoring [1]:

- 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

Three newer programs for heat map [2] and allosteric network analysis:

- tagging.py: performs a mapping of functionally important residues based e.g. on the correlations of absolute differentials of pairwise residue contact distances with an external activity function (such as returned by agility.py or terrain.py) [2]
- turning.py: performs a mapping of functionally important residues based on correlations of absolute differentials of backbone pivot angles with an external activity function (such as returned by agility.py or terrain.py) [2]
- signals.py: performs an allosteric network analysis based on pair contact correlations or a Cartesian covariance matrix [unpublished]

More details of the purpose, input, and output of each program are given on the following pages. These ten programs are supported by five library 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 compact and readable.

The name of the support modules corresponds to their functionality:

- mod_pio.py: PDB (atoms and bonds) 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

The ten 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@biomachina.org.

2. Integration into the MD Workflow

Through the use of VMD molfile plugins [3], 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.)

It is expected that the user has access to trajectory management software such as VMD for such basic editing tasks as stride (time-step) modification, least-squares alignment of trajectory frames, and unwrapping of coordinates (in case of periodic boundaries). TimeScapes does not support such trajectory editing.

When preparing trajectories with external software, we recommend removing the solvent and keeping only the non-hydrogen (heavy) protein atoms to reduce disk space and read access time. In the case of periodic boundaries, coordinates must be unwrapped prior to using TimeScapes (there should be no atoms wrapped around the periodic box, please inspect your trajectory if necessary). TimeScapes results have been tested to be robust under different stride. Setting the stride with external software mainly affects computational efficiency of the TimeScapes runs. As a rule of thumb we recommend for testing purposes to stride the trajectories initially to a small number of frames (100-1000) and/or to select a subset of the structure, before carrying out slower and more detailed analyses with TimeScapes.

Finally, the user must prepare a corresponding PDB file from which TimeScapes programs obtain atom masses and the coarse side chain model (typically, one atom is representative for a full side chain, see [1]). The PDB file should be carefully inspected for non-standard amino acid or atom names, especially at the N and C termini. 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 modified 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 described in Section 3.

A macro file was provided with this user guide to facilitate rendering of the coarse grained model `mod_pwk_side` with the VMD molecular

graphics program. After sourcing (from the VMD command menu) `representative_atoms_macro_source_me.vmd`, the new selection "representative" is available within VMD's Graphical Representations menu (under "Selections", "Singlewords") which selects the atoms of the coarse-grained side-chain model (see appendix C).

3. User Guide (alphabetically by program name)

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 [-full] [-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 (FWHM = 2 sqrt(2 ln 2) 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
- [`-full`]: Optional full output incl. basin trajectory files (these can be large, so they must be requested explicitly)
- [`-lsq`]: Optional least-squares fitting of trajectory frames to the input PDB

Output

- *outname*_segmentation.dat: Raw data file with frame number, RMS fluctuations, first derivative, basin label (for plotting purposes or to provide an activity rate function for tagging.py and turning.py)
- *outname*_transitions.dcd: DCD trajectory file with frames of basin transitions (if `-full` option is specified)
- *outname*_transitions.log: Corresponding log file with program parameters and frame information
- *outname*_minima.dcd: DCD trajectory file with frames of basin minima (if `-full` option is specified)
- *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 might be slow; pre-process trajectory and leave `-lsq` off, if possible
- Log files contain detailed output information even when the corresponding trajectories are not explicitly written (in default mode)
- 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 (default: `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 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 CA 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

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

---

gmdhist.py - computing GMD pair distribution histograms

Purpose

Facilitates plotting of pair distance distribution histograms for the evaluation of GMD orders as shown in [1]

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 (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 (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`)

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 in column format

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

- Mean and standard deviation for each input time series, correlation coefficients

---

signals.py – predicting allosteric signaling networks

Purpose

Prediction and visualization of allosteric signaling networks in proteins. Currently, the program supports two prediction modalities (based on pairwise residue distance geometry or based on the absolute Cartesian coordinate covariance). For testing purposes, a third network based on the inverse mean distances is also exported. The pairwise residue *distance* based approach is analogous to that described for pairwise residue interaction *energies* by Kong & Karplus (Proteins 2009, 74:145).

Usage

```
$ signals.py infile1 infile2 outname [-lsq] \
    [excl [ccut [dcut [csel]]]]
```

Input (3-8 arguments)

- `infile1`: PDB file used for coarse model assignment
- `infile2`: Trajectory file, for supported formats see Section 2
- `outname`: User-defined basename prefix for output file names
- [`-lsq`]: Optional least-squares fitting of trajectory frames to the input PDB (for Cartesian covariance analysis only)

The following optional parameters must be entered in order starting from top (argument list can be truncated):

- [`excl`]: Neighbor contact exclusion in coarse (residue) model (default: 1; excludes undesired correlation contributions from neighboring residues)
- [`ccut`]: Correlation cutoff (used only in distance geometry approach); allowable `ccut` range: [0 1]; absolute values <= ccut are set to zero (default value: 0.0)
- [`dcut`]: Distance cutoff > 0.0 in Å; only residue contacts with (mean-std.dev. ≤ `dcut`; statistics over time) will be considered (default: `'inf'`)
- [`csel`]: Optional user-provided coarse-graining function defined in mod_pwk.py (default: `mod_pwk_side`)

Output

- *outname*_cgcor.dat: Data file with distance-geometry based contact correlation matrix
- *outname*_cgcor.tcl: Corresponding VMD-sourceable (editable) Tcl script with weighted graph connectivity in descending order
- *outname*_cgcov.dat: Data file with (absolute) covariance matrix
- *outname*_cgcov.tcl: Corresponding VMD-sourceable (editable) Tcl script with weighted graph connectivity in descending order
- *outname*_cgcor.dat: Data file with (inverse, mean) distance matrix (for testing purposes)
- *outname*_cgcor.tcl: Corresponding VMD-sourceable (editable) Tcl script with weighted graph connectivity in descending order

Caveats and Notes

- Trajectory coordinates must be unwrapped
- Alignment of frames is required only for covariance analysis
- Some networks are very dense. The VMD-sourceable Tcl script can be truncated (with an editor or the UNIX `head` command) to limit the network to the most important interactions

tagging.py - mapping of functionally important residues

Purpose

Like its companion program turning.py, tagging.py performs a mapping of functionally important residues whose fast, local dynamics correlates with the slow, global dynamics. Here the analysis is based on pairwise residue contact distances (similar to signals.py) whose absolute time differentials are correlated with an external non-negative activity function (such as returned by agility.py or terrain.py). Alternatively, the `-raw` option may be specified for a direct correlation of plain pairwise residue contact distances with an external order parameter. The correlation analysis can be projected back to residue space, yielding a localization of functional hotspots on side chains.

Usage

```
$ tagging.py infile1 infile2 infile3 outname [-mi] [-raw] \
      [-norm] [colnr [excl [dcut [csel]]]]
```

Input (4-10 arguments)

- `infile1`: PDB file used for coarse model assignment
- `infile2`: Trajectory file, for supported formats see Section 2
- `infile3`: White space delimited time series data for ranking in column format (typically the *segmentation.dat file returned by agility.py or terrain.py, or an external order parameter for `-raw`)
- `outname`: User-defined basename prefix for output file names
- [`-mi`]: Option to turn on mutual information (fast information matching with BADE [4]), otherwise Pearson cross correlation will be used in the ranking (see [2] for details).
- [`-raw`]: Option to directly use raw time series for ranking (instead of absolute time differentials)
- [`-norm`]: Option to normalize intermediate correlations to the range [0,1] (useful for analyzing weak correlations)

The following optional parameters must be entered in order starting from top (argument list can be truncated):

- [`colnr`]: Column number of `infile3` that will be used for ranking (default: 2; counting columns from 1)
- [`excl`]: Neighbor contact exclusion in coarse (residue) model (default: 0; may be used to exclude correlation contributions from neighboring residues if desired)
- [`dcut`]: Distance cutoff > 0.0 in Å; only residue contacts with (mean-std.dev. ≤ `dcut`; statistics over time) will be considered (default: `'inf'`)
- [`csel`]: Optional user-provided coarse-graining function defined in mod_pwk.py (default: `mod_pwk_side`)

Output

- *outname*_log: Log file with parameters
- *outname*_pairwise.dat: Data file with coarse grained (residue-ID based) pairwise Pearson correlation matrix
- *outname*_pairwise_resname.dat: Data file with 20x20 matrix of correlations mapped to resname-resname space in cumulative fashion. Resnames consistent with appendix C are assigned to row and column indices in the order ARDNCQEGHILKMFPSTWYV
- *outname*_pairwise_resname_count.dat: Data file with corresponding number of projections to each resname-resname bin
- *outname*_pairwise_resname_normalized.dat: Data file with normalized resname-resname correlations (i.e. cumulative results divided by count)
- *outname*_pairwise_resname_max.dat: Data file with maximum resname-resname correlations (winner-take-all projection)
- *outname*_dump.dat: Data file with resname indices (0-19 corresponding to the order ARDNCQEGHILKMFPSTWYV) and correlation values that can be used for an external meta-analysis across many trajectories
- *outname*_projected.dat: Data file with correlation values projected to sequence
- *outname*_projected.pdb: PDB file with projected correlation values in B-factor column (for visualization purposes)

Caveats and Notes

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

---

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 [-full] [-lsq] [csel [msel]]
```

Input (7-11 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
- [`-full`]: Optional full output incl. graph and event trajectory files (these can be large, so they must be requested explicitly)
- [`-lsq`]: Optional least-squares fitting of output trajectory frames to the input PDB (if `-full` option is specified).
- [`csel`]: Optional user-provided coarse-graining function defined in mod_pwk.py (the default is `mod_pwk_side`)
- [`msel`]: For GMD graphs only, 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*_graphs: Directory containing VMD-sourceable Tcl scripts with graph connectivity for each frame (if `-full` option is specified)
- *outname*_events.log: Log file of individual contact forming or breaking events
- *outname*_events.dcd: DCD trajectory containing only event frames (if `-full` option is specified)
- *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 (for plotting purposes or to provide an activity rate function for tagging.py and turning.py)
- *outname*_transitions.dcd: DCD trajectory file with frames of basin transitions (if `-full` option is specified)
- *outname*_transitions.log: Corresponding log file with program parameters and frame information

- *outname*_minima.dcd: DCD trajectory file with frames of basin minima (if `-full` option is specified)
- *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
- You can specify your own coarse graining and masking functions via the `csel` and `msel` parameters (`csel` must be set before `msel` can be set). The intended use of this functionality is 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
- Log files contain detailed output information even when the corresponding trajectories are not explicitly written (in default mode)
- If the program runs slowly or runs out of memory try first to use `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`

---

turning.py - mapping of functionally important residues

Purpose

Like its companion program tagging.py, turning.py performs a mapping of functionally important residues whose fast, local *turning* motion (hinge-bending) correlates with the slow, global dynamics. Here the analysis is based on 'pivot residue' dihedral angles (four consecutive alpha carbon dihedral: Yan et al., J. Protein Chem. 1999, 18:807) whose absolute time differentials are correlated with an external non-negative activity function (such as returned by agility.py or terrain.py). The correlation analysis can be projected back to residue space, yielding a localization of functional hotspots on the protein backbone.

Usage

```
$ turning.py infile1 infile2 infile3 outname [-mi] \
      [-norm] [colnr]
```

Input (4-6 arguments)

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

- `infile3`: White space delimited time series data for ranking in column format (typically the *segmentation.dat file returned by agility.py or terrain.py)
- `outname`: User-defined basename prefix for output file names
- [`-mi`]: Option to turn on mutual information (fast information matching with BADE [4]), otherwise Pearson cross correlation will be used in the ranking (see [2] for details).
- [`-norm`]: Option to normalize intermediate correlations to the range [0,1] (useful for analyzing weak correlations)
- [`colnr`]: Optional column number of `infile3` that will be used for ranking (default: 2; counting columns from 1)

Output

- *outname*_log: Log file with parameters and dihedral residue numbers
- *outname*_dihedrals.dat: Data file with pivot residue dihedral time series
- *outname*_differentials.dat: Data file with correlation-based rankings of pivot residues
- *outname*_turning.dat: Data file with correlation based rankings of pivot residues projected to sequence
- *outname*_turning.pdb: PDB file with projected correlation values in B-factor column (for visualization purposes)
- *outname*_pairwise_resname.dat: Data file with 20x20 matrix of correlations mapped to resname-resname space in cumulative fashion. Resnames consistent with appendix C are assigned to row and column indices in the order ARDNCQEGHILKMFPSTWYV
- *outname*_pairwise_resname_count.dat: Data file with corresponding number of projections to each resname-resname bin
- *outname*_pairwise_resname_normalized.dat: Data file with normalized resname-resname correlations (i.e. cumulative results divided by count)
- *outname*_pairwise_resname_max.dat: Data file with maximum resname-resname correlations (winner-take-all projection)
- *outname*_dump.dat: Data file with resname indices (0-19 corresponding to the order ARDNCQEGHILKMFPSTWYV) and correlation values that can be used for an external meta-analysis across many trajectories

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

- Due to the cyclic angle variables, the program does not support direct correlation with an external order parameter as in tagging.py.
- No atom selection is supported, dihedrals are computed for all (consecutive) alpha carbon quadruplets

4. Usage Ideas and Examples

The following brief tutorials provide usage ideas and workflow examples.

Evaluating contacts, events, and activities

Reference [1] provides the best reference for the original event detection and activity monitoring applications. The activity curves are most useful for plotting figures and as input for subsequent mapping of functional residues (see below), 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 can be saved (via the `-full` option) as DCD trajectory files.

Variable stride time compression

Compared to trajectories with fixed stride access pattern, the output DCD trajectory *events.dcd (returned by terrain.py using the `-full` option) is able to bridge between wider ranges 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 `csel` and `msel` 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 (the requirement to specify DCD output explicitly with the `-full` option was introduced in version 1.3 to avoid filling up the disk space accidentally).

Prediction of signaling networks and mapping of hot spots

As described above, the programs signals.py, tagging.py, and turning.py perform the analysis of functionally relevant motion and its mapping onto a network or the protein structure.

Detailed usage examples for heat map analysis using tagging.py and turning.py are provided in [2]. It has already been noted that these tools make use of the activity rate functions (returned by agility.py or terrain.py) to perform a heat mapping of residues relevant for the global activity. It should be noted that the two programs are somewhat complementary. While the turning analysis in turning.py mainly is focused on the protein backbone, the distance geometry of tagging.py is focused on pairwise side-chain interactions. We have used both programs routinely for the following tasks: (i) to inspect functional hotspots in (pairwise) residue ID (sequence) space, (ii) to map the hotspots onto linear sequence (via B-factor fields in the resulting PDB files), and (iii) to map the hot spots to (pairwise) resname space. The last approach allows one to implement an external meta-analysis across different protein systems, which we have used e.g. in a stability analysis of MD force fields across a large trajectory database.

The prediction of signaling networks with signals.py will be described in a future publication. It should be noted that the simple editing of the Tcl files returned by signals.py facilitates a basic “community analysis”. By truncating the ordered list of graph edges, one can reveal only the strongest edges that typically yield one or two connected communities.

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.

D. TimeScapes 1.3 introduced three new programs, signals.py, tagging.py, and turning.py (see above). The programs were augmented by several new functions added to the respective mod_* support modules (see documentation in the source code): mod_twk_stat_cutoff, mod_twk_average, mod_twk_covariance_flat, mod_twk_covariance, mod_pio_write, mod_pio_write_weighted_bonds, and mod_pwk_lsq_fit_no_mass. A VMD-sourceable macro was provided to facilitate rendering of coarse grained models: `representative_atoms_macro_source_me.vmd` (see Section 2). Some minor improvements were implemented in the existing programs: For example, the function mod_twk_kernel_estimator (used by agility.py and terrain.py) now prints progress information, a new `-full` option was added to agility.py and terrain.py to limit the output of large data files (that may not be needed by every user) in the default mode, and an option to align frames of the output trajectories was added to terrain.py. Due to the move of main author Willy Wriggers in July 2014, ownership was transferred from D. E. Shaw Research to Willy Wriggers, affecting the license terms in appendix D with version 1.3.1.

E. TimeScapes 1.4 introduced the optional use of mutual information instead of Pearson cross correlation in the tagging.py and turning.py tools [2].

F. TimeScapes 1.5 improved the mutual information calculation by introducing the BADE algorithm [4]. At the time of this writing the extension to lipids and solvent molecules in [5] is not yet implemented, but will be released in a future version 1.5.1. Readers of [5] who require the functionality should inquire about a beta version by contacting timescapes@biomachina.org .

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/doi/full/10.1021/ct900229u

Supporting information:

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

[2] Julio Kovacs and Willy Wriggers, "Spatial Heat Maps from Fast Information Matching of Fast and Slow Degrees of Freedom: Application to Molecular Dynamics Simulations," J. Phys. Chem. B., 2016, 120 (33), pp 8473–8484, DOI: 10.1021/acs.jpcb.6b02136, http://pubsdc3.acs.org/doi/full/10.1021/acs.jpcb.6b02136

Supporting information:

http://pubsdc3.acs.org/doi/suppl/10.1021/acs.jpcb.6b02136

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

[4] Julio Kovacs, Cailee Helmick, and Willy Wriggers, "A Balanced Approach to Adaptive Probability Density Estimation" Front. Mol. Biosci., 2017, volume 4 (article 25) DOI: 10.3389/fmolb.2017.00025, http://journal.frontiersin.org/article/10.3389/fmolb.2017.00025/full

[5] Willy Wriggers, Federica Castellani, Julio Kovacs and P. Thomas Vernier, "Computing Spatiotemporal Heat Maps of Lipid Electropore Formation: A Statistical Approach," Front. Mol. Biosci., 2017, volume 4 (article 22) DOI: 10.3389/fmolb.2017.00022, http://journal.frontiersin.org/article/10.3389/fmolb.2017.00022/full

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

License Grant. 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 Willy Wriggers, Biomachina.org, D. E. Shaw Research, nor the names of its contributors may be used to endorse or promote products derived from this Software without specific prior written permission.

Acknowledgement and Citation. Licensee agrees to acknowledge the use of the Software in any reports or publications of results obtained with the Software as

> "TimeScapes Analytics Package, version 1.X, http://timescapes.biomachina.org, 2016"

where 'X' is to be replaced with the minor release number of the version used in the published research. Licensee is also requested to include a citation to the following paper:

> Julio Kovacs and Willy Wriggers, "Spatial Heat Maps from Fast Information Matching of Fast and Slow Degrees of Freedom: Application to Molecular Dynamics Simulations," J. Phys. Chem. B., 2016, 120 (33), pp 8473–8484, DOI: 10.1021/acs.jpcb.6b02136, http://pubsdc3.acs.org/doi/full/10.1021/acs.jpcb.6b02136

If the published research is based on results obtained with any Software Modification or any complementary code, then those variants must be acknowledged as such.

Disclaimer of Warranties and Liabilities. 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.

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 pri-
or written permission.

THE SOFTWARE IS PROVIDED AS IS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IM-
PLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE CONTRIBU-
TORS 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 <akohlmey@cmm.chem.upenn.edu>
```