# bitclust Documentation

***Release 0.0.6***

**Roy Gonzalez-Aleman**

**Jul 25, 2022**

# Contents

# CHAPTER 1

## Description

**BitClust** is a Python command-line interface (CLI) conceived for fast clustering of relatively long Molecular Dynamics trajectories following the Daura's algorithm[1]. Retrieved clusters are roughly equivalent to those reported by **VMD's** internal command **measure cluster** but they are computed in a much faster way (see benchmark section for more details).

What **BitClust** offers is a classical tradeoff; RAM for speed. It can calculate all pairwise distances between frames to run a clustering job and then store them in memory instead of recalculating them whenever a cluster is found.

It is worth noting that memory resources have been deeply optimized by encoding similarity distances as bits (0 if the distance is less equal than a specified threshold, 1 otherwise). This encoding result in a storage reduction of at least 32X/64X when compared to similar algorithms that save the same information as single/double-precision float values.

[1] Daura, X.; van Gunsteren, W. F.; Jaun, B.; Mark, A. E.; Gademann, K.; Seebach, D. Peptide Folding: When Simulation Meets Experiment. Angew. Chemie Int. Ed. 1999, 38 (1/2), 236–240.

CHAPTER 2

# Main Dependencies

**BitClust** is built on the shoulders of two giants:

- MDTraj software that allows a very fast calculation of RMSD pairwise distances between all frames of trajectories in a parallelized fashion **and**
- bitarray third-party python library which offers a memory-efficient data structure of bit-vectors (bit arrays) and a set of bitwise operations that are the very heart of our clustering implementation.

CHAPTER 3

# Citation

If you make use of **BitClust** in your scientific work, **BitCool** and cite it ;)

CHAPTER 4

# Licence

**BitClust** is licensed under GNU General Public License v3.0.

## 4.1 Installation

### 4.1.1 MDTraj

Before installing **BitClust**, please be sure that `mdtraj` is available in your system. It is recommended that you install `mdtraj` using conda.

```
$ conda install -c conda-forge mdtraj
```

`conda` is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies.

You can install `mdtraj` with `pip`, if you prefer.

```
$ pip install mdtraj
```

`pip` is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes.

Any of them will install `mdtraj` along with all dependencies from a pre-compiled binary. If you don't have Python or the `conda` package manager, you can start with the Anaconda Scientific Python distribution. `pip` homepage is available here

### 4.1.2 BitClust

Via **pip**:

After succesful installation of `mdtraj` you can easily proceed to install **BitClust** and the rest of its dependencies using pip.

```
$ pip install bitclust
```

Then, you should be able to see **BitClust** help by typing in a console:

```
$ bitclust -h
```

Via **GitHub**:

After succesful installation of `mdtraj` you can easily proceed to install **BitClust** and the rest of its dependencies from GitHub.

```
$ git clone https://github.com/rglez/bitclust
$ cd bitclust
$ python setup.py install
```

## 4.2 Usage

### 4.2.1 Valid formats

**BitClust** inherits valid trajectory and topology formats from **MDTraj**.

Valid trajectory extensions are: `.dcd`, `.dtr`, `.hdf5`, `.xyz`, `.binpos`, `.netcdf`, `.prmtop`, `.lh5`, `.pdb`, `.trr`, `.xtc`, `.xml`, `.arc`, `.lammpstrj` and `.hoomdxml`.

If trajectory format does not include topological information, user must pass a path to a topology file. Valid topology extensions are: `.pdb`, `.pdb.gz`, `.h5`, `.lh5`, `.prmtop`, `.parm7`, `.prm7`, `.psf`, `.mol2`, `.hoomdxml`, `.gro`, `.arc` and `.hdf5`.

### 4.2.2 Default usage

Once **BitClust** installed, you can have access to the program's help, which contains short descriptions of available arguments, by running

```
$ bitclust -h
```

Only one argument is always mandatory, `-traj`, which specifies the path to the trajectory file. If the trajectory format does not provide topological information of the system, it must be supplied from a topology file through the `-top` argument. All other arguments are always optional and if not explicitly specified they will take their default values commented below.

A minimal run like

```
$ bitclust -top tau_6K.pdb -traj tau_6K.dcd
```

loads the tau_6K.dcd trajectory into tau_6K.pdb coordinates (both present at current working directory as not path was provided) and performs a clustering job using Daura's algorithm with a cutoff of 1A (`-cutoff` 1) on the whole trajectory.

Arguments `-first`, `-last` and `-stride` can be used to select an interval (they default to 0, last frame and 1 respectively).

Default atom selection corresponds to all atoms (`-sel` all). **BitClust** will retrieve all clusters with at least 2 frames (`-size` 2).

Frame 0 will be used as reference (`-ref 0`) to make an RMSD graph. All produced output will be saved in the current working directory (`-odir .`).

### 4.2.3 Default outputs

**BitClust** outputs basic graphics for fast inspection of the clustering job results (see figure below). All these graphs and others can be constructed from two generated text files: **clusters_statistics.txt** and **frames_statistics.txt**.

The first one contains as columns every `cluster ID`` (starting from 0, -1 corresponding to unclustered frames), their `size`, the `percent` this size represents from the total of frames and the `center` frame of every cluster.

The second file contains as columns every `frame ID` (starting from 0), the `cluster ID` where every frame belongs to and the `RMSD` value of every frame respect to the specified reference (default reference is frame 0).

---

**BitClust's basic graph outputs**

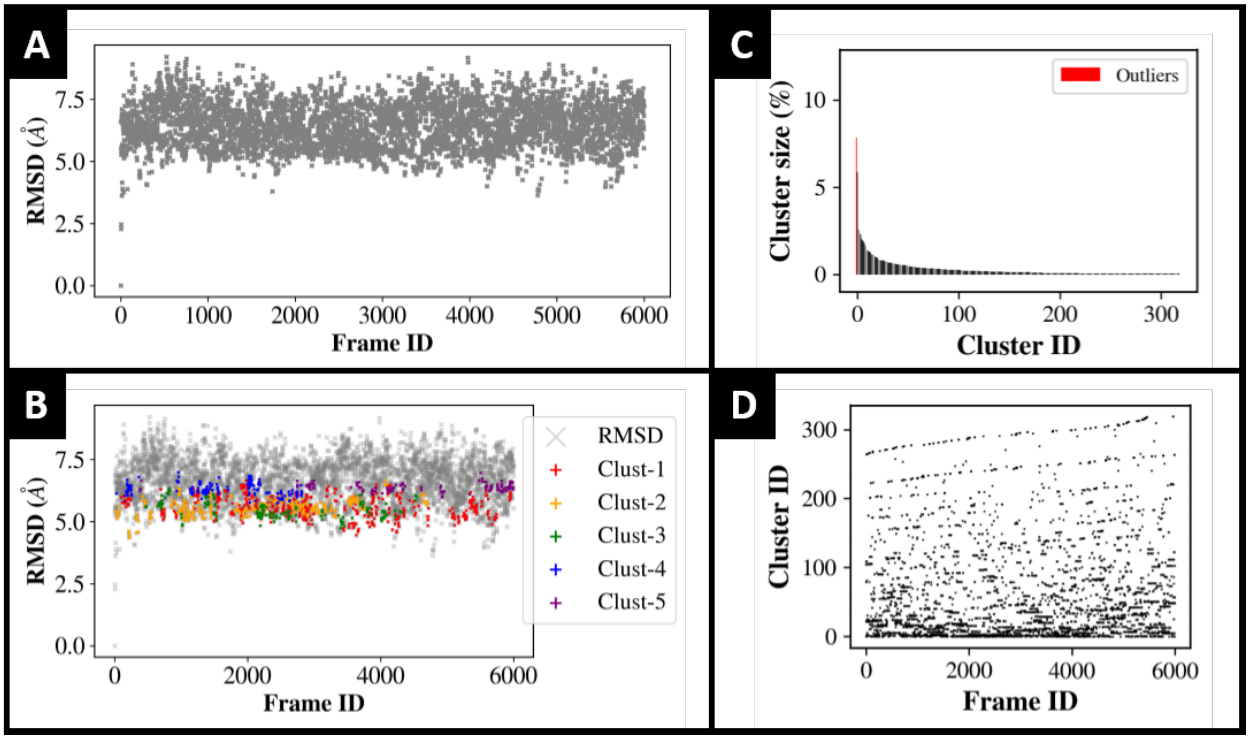
**Figure A:** RMSD of all frames in trajectory versus reference frame passed to argument `-ref`. Useful for fast visualization of trajectory's geometrical dispersion.

**Figure B:** Superposition of first five clusters onto **Figure A**. Useful for fast visualization of most populated clusters location along the trajectory.

**Figure C:** Clusters (including outliers in red) size. Useful for inspection of clusters relative population.

**Figure D:** Cluster lines. Useful to qualitatively assessment on temporal distribution of clusters.

---




### 4.2.4 Selection syntax

**BitClust** inherits atom selection syntax from **MDTraj** which is similar to that in VMD. We reproduce below some of the **MDTraj** examples. Note that in **BitClust** all keywords (or their synonyms) string are passed directly to `-sel`

argument as it is illustrated in the Usage Examples section. For more details on possible syntax, please refer to MDTraj original documentation.

**MDTraj** recognizes the following keywords.

| Keyword | Synonyms | Type | Description |
|---|---|---|---|
| `all` | `everything` | `bool` | Matches everything |
| `none` | `nothing` | `bool` | Matches nothing |
| `backbone` | `is_backbone` | `bool` | Whether atom is in the backbone of a protein residue |
| `sidechain` | `is_sidechain` | `bool` | Whether atom is in the sidechain of a protein residue |
| `protein` | `is_protein` | `bool` | Whether atom is part of a protein residue |
| `water` | `is_water`, `waters` | `bool` | Whether atom is part of a water residue |
| `name` | | `str` | Atom name |
| `index` | | `int` | Atom index (0-based) |
| `type` | `element`, `symbol` | `str` | 1 or 2-letter chemical symbols from the periodic table |
| `mass` | | `float` | Element atomic mass (daltons) |
| `residue` | `resSeq` | `int` | Residue Sequence record (generally 1-based, but depends on topology) |
| `resid` | `resi` | `int` | Residue index (0-based) |
| `resname` | `resn` | `str` | Residue name |
| `rescode` | `code`, `resc` | `str` | 1-letter residue code |
| `chainid` | | `int` | Chain index (0-based) |

#### Operators

Standard boolean operations (`and`, `or`, and `not`) as well as their C-style aliases (`&&`, `||`, `!`) are supported. The expected logical operators (`<`, `<=`, `==`, `!=`, `>=`, `>`) are also available, as along with their FORTRAN-style synonyms (`lt`, `le`, `eq`, `ne`, `ge`, `gt`).

#### Range queries

Range queries are also supported. The range condition is an expression of the form `<expression> <low> to <high>`, which resolves to `<low> <= <expression> <= <high>`. For example

```
# The following queries are equivalent
-sel "resid 10 to 30"
-sel "(10 <= resid) and (resid <= 30)"
```

### 4.2.5 Usage examples

Next, you will find some usage examples of **BitClust**.

```
# An interval (1000 frames) of tau_6K.pdb trajectory (no topology file is needed)
# will be clustered with default values for all other arguments (see help section).

$ bitclust  -traj tau_6K.pdb -first 0 -last 100000 -stride 100
```

```
# Clustering all atoms but hydrogen´ ones.

$ bitclust -top tau_6K.pdb -traj tau_6K.dcd -sel "\"name =~ '[^H.*]'\""
```

```
# Backbone atoms of trajectory tau_6K.dcd will be clustered using a cutoff of 4 A.
# Retrieved clusters will have at least 15 frames and output RMSD graphs will use
# frame 2580 (counting from 0) as a reference structure.

$ bitclust -top tau_6K.pdb -traj tau_6K.dcd -sel "backbone" -cutoff 4 -minsize 15 -
↪ref 2580
```

```
# Default run saving results to local/test/run1 (relative path to current working␣
↪directory)

$ bitclust  -traj tau_6K.pdb -odir "local/test/run1"
```

## 4.3 Help

### 4.3.1 Basic help

**BitClust** help is displayed in the console when typing **bitclust -h**

```
$ bitclust -h

usage: bitclust [-h] [-top TOPOLOGY] [-traj TRAJECTORY] [-first FIRST]
                [-last LAST] [-stride STRIDE] [-sel SELECTION]
                [-cutoff CUTOFF] [-minsize MINSIZE] [-max_clust MAX_CLUST]
                [-ref REFERENCE] [-numouts NUM_OUTS] [-odir OUTDIR]

BitClust: Fast & memory efficient clustering of long MD trajectories

optional arguments:
  -h, --help            show this help message and exit
  -top TOPOLOGY         path to topology file (psf/pdb)
  -traj TRAJECTORY      path to trajectory file
  -first FIRST          first frame to analyze (starting from 0)
  -last LAST            last frame to analyze (starting from 0)
  -stride STRIDE        stride of frames to analyze
  -sel SELECTION        atom selection (MDTraj syntax)
  -cutoff CUTOFF        RMSD cutoff for pairwise comparisons in A
  -minsize MINSIZE      minimum number of frames inside returned clusters
  -max_clust MAX_CLUST  maximum number of returned clusters
  -ref REFERENCE        reference frame to align trajectory
  -numouts NUM_OUTS     number of clusters and leaders to print as pdb
  -odir OUTDIR          output directory to store analysis
```

### 4.3.2 Arguments details

`-traj (str)`: This is the only argument that is **always** required. Valid extensions for trajectories are `.dcd`, `.dtr`, `.hdf5`, `.xyz`, `.binpos`, `.netcdf`, `.prmtop`, `.lh5`, `.pdb`, `.trr`, `.xtc`, `.xml`, `.arc`, `.lammpstrj` and `.hoomdxml`.

`-top (str)`: If trajectory format includes topological information this argument is not required. Otherwise, user must pass a path to a topology file. Valid topology extensions are `.pdb`, `.pdb.gz`, `.h5`, `.lh5`, `.prmtop`, `.parm7`, `.prm7`, `.psf`, `.mol2`, `.hoomdxml`, `.gro`, `.arc` and `.hdf5`.

`-first (int, default=0)`: First frame to analyze (starting count from 0)

`-last (int, default=-1)`: Last frame to analyze (starting count from 0). Value -1 indicates that last frame will be used.

`-stride (int, default=1)`: Stride of frames to analyze. Use this argument to reduce trajectory size when performing exploratory analysis of cutoff value.

`-sel (str, default='all')`: Atom selection. **BitClust** inherits `MDtraj` syntax selection which is very flexible. For a deeper insight please refer to the `MDTraj atom selection reference` original documentation. Common cases are listed at usage examples section.

`-cutoff (int, default=1)`: RMSD cutoff for similarity measures given in Angstroms (1 A = 0.1 nm).

`-minsize (int, default=2)`: Minimum number of frames inside returned clusters. 0 is not a meaningful value and 1 implies an unclustered frame (no other frame is similar to it). Greater values of this parameter can speed up the algorithm.

`-max_clust (int, default=all)`: Maximum number of calculated clusters. Change the default for a better performance whenever you need only the first clusters.

`-ref (int, default=0)`: Reference frame to align trajectory.

`-numouts (int, default=5)`: Number of clusters and their corresponding representative frames (leaders) to print as pdb.

`-odir (str, default=".")`: Output directory to store analysis. If not specified, files and figs will be stored in current working directory.

## 4.4 Benchmark

A set of commonly used options for clustering MD simulations has been chosen for performance comparison between BitClust and other clustering software. Run time and memory consumption of each method is reported in Table 2 for the three trajectories 6K, 100K, and 500K described in the Computational Details section (having 6000, 100000 and 500000 frames respectively).

The clustering method selected for each software was as follows; **Daura** for BitClust and GROMACS (through the gromos option), **quality threshold** for py-MS and VMD, **qt-like** for Wordom and **median-linkage** for TTClust. We would like to stress out that despite their native denomination in their original software, the chosen algorithms correspond all to **Daura**, as it has been recently reported[1]

In the case of VMD, we decided to show the performance of processing five (VMD-5, the default value) and all (VMD-ALL) clusters as a way to evaluate the usefulness of its implementation, which is specially conceived for preserving memory resources.

**Table 2** Run time ([hh:]mm:ss) and memory peaks (GB) comparison of several clustering algorithms. A crash was declared if the job consumed more than 64 GB of RAM or last for more than 60 h. (For an extensive discussion of this benchmark, you can refer to the academic article[2])

[1] Roy González-Alemán, David Hernández-Castillo, Julio Caballero, and Luis A. Montero-Cabrera. Journal of Chemical Information and Modeling. DOI: 10.1021/acs.jcim.9b00558

[2] Roy González-Alemán, David Hernández-Castillo, Alejandro Rodríguez-Serradet, Julio Caballero, Erix W. Hernández-Rodríguez, and Luis Alberto Montero-Cabrera. Journal of Chemical Information and Modeling. DOI: 10.1021/acs.jcim.9b00828.

| Software | Trajectory 6K | | Trajectory 100K | | Trajectory 500K | |
|---|---|---|---|---|---|---|
| | Run time | Memory peak | Run time | Memory peak | Run time | Memory peak |
| BitClust | 00:04 | 0.15 | 01:25:28 | 9.41 | 06:00:08 | 33.84 |
| py-MS | 00:04 | 0.15 | 01:46:02 | 61.54 | 03:03:49 | CRASH |
| Wordom | 00:04 | 0.15 | 01:26:13 | CRASH | 02:54:23 | CRASH |
| VMD-5 | 00:04 | 0.15 | 04:59:22 | 6.97 | 29:34:43 | 4.13 |
| VMD-ALL | 00:04 | 0.15 | 05:08:10 | 6.97 | CRASH | 4.14 |
| TTClust | 00:04 | 0.15 | 00:01:59 | CRASH | 00:00:37 | CRASH |
| GROMACS | 00:04 | 0.15 | 26:13:29 | 52.22 | 01:09:03 | CRASH |

### 4.4.1 References

## 4.5 Changelog

### 4.5.1 Version 0.0.11 (10/05/2020)

- Change of contact mail to academic adresses
- Function of bitarray_ph4 library replaced by new versioof equivalents in bitarray library
- Bug in the argument "-last" is fixed. Now it defaults to None instead of -1
- New argument "-max_clust" lets user specify the maximum number of cluster to calculate
- New argument "-numouts" lets users specify a number of clusters and leaders to be printed as independent pdb files
- The function "load_trajectory" now raise errors if selections are incorrect or contain no atoms
- The function "load_trajectory" now treat selections in a way that improve RAM savings

### 4.5.2 Version 0.0.6 (October, 2019)

- Initial stable public release after publication.

CHAPTER 5

# References