mDCC_tools Documentation

Release 0.91.f

Kota Kasahara

February 05, 2016

CHAPTER

ONE

INTRODUCTION

Author Kota Kasahara

1.1 What is mDCC?

The multi-modal dynamic cross correlation (mDCC) is a method for analyzing trajectories generated by molecular dynamics (MD) simulations. The mDCC was developed by KASAHARA Kota, FUKUDA Ikuo, and NAKAMURA Haruki, at Institute for Protein Reasearch, Osaka University. See the original manuscript for details: A Novel Approach of Dynamic Cross Correlation Analysis on Molecular Dynamics Simulations and Its Application to Ets1 Dimer-DNA Complex. 2014 PLoS ONE 9:e112419 *[Kasahara_2014]*.

1.2 mDCC tools

This tool kit includes ...

- Programs
  - mdcc_learn
    - Detection of modes of atomic motions
  - mdcc_assign
    - Calculation of probability density functions for each atom at each time step
  - python scripts
- This document
- Sample files for the tutorial
  - A trajectory file written in the Gromacs .trr format
  - .bash files to execute analysis programs
  - Configuration files as input of the programs

1.3 Installation

The path to the home directory of the mDCC tools should be set as the shell variable ${MDCCTOOLS}. For example,:

```
export MDCCTOOLS=${HOME}/local/mdcctools
```

This tool kit includes two C++ programs (*mdcc_learn* and *mdcc_assign*) and some python scripts. The C++ programs need to be compiled.

The additional information of installation of dependencies are described in the Appendix.

1.3.1 mdcc_learn

mdcc_learn program performs a pattern recognition on a spatial distribution of atomic coordinates in a trajectory.

The source codes of *mdcc_learn* are placed in ${MDCCTOOLS}/src/mdcc_learn directory.

mdcc_learn requires LAPACK library. The name of LAPACK library and path to the library file should be specified in the Makefile.

> PATH_LAPACKLIB = ${HOME}/lib
>
> LAPACKLIB = -llapack

To build *mdcc_learn*, execute the *make* command and move the generated binary to ${MDCCTOOLS}/bin directory:

```
make
mv mdcc_learn ../../bin
```

1.3.2 mdcc_assign

mdcc_assign program calculates the probability density for each data point of atomic coordinates in a trajectory on the basis of the results of *mdcc_learn* program.

This program requires LAPACK and BOOST libraries. The name and path of LAPACK library file and the path of BOOST include files should be specified in the Makefile:

```
PATH_LAPACKLIB = ${HOME}/lib
LAPACKLIB = -llapack
BOOSTINC = $(HOME)/include
```

To build *mdcc_assign*, execute the *make* command and move the generated binary to ${MDCCTOOLS}/bin directory:

```
make
mv mdcc_assign ../../bin
```

1.3.3 Python scripts

Many python scripts are located in ${MDCCTOOLS}/bin directory. They are written for python2.7 and requires the libraries:

- numpy
- scipy
- mdanalysis
- networkx

They should be installed in paths in ${PYTHONPATH} environment variable.

All these libraries can be obtained by using *easy_install* command.

1.3.4 Other programs for tutorial

mDCC tools output the results as tab-separated text or binary files. In order to visualize the data, some analyses tools are useful. In the tutorial, SQLite3, R, and Cytoscape are used. However, users can apply any other software, such as gnuplot and graphvis.

- https://www.sqlite.org/
- http://www.r-project.org/
- http://www.cytoscape.org/

For R software, the three libraries are used.

- reshape
- ggplot
- plyr

They can be installed with install.packages() command in the R shell.

1.4 License

mDCC_tools is distributed under GPL ver.3 liscense.

CHAPTER

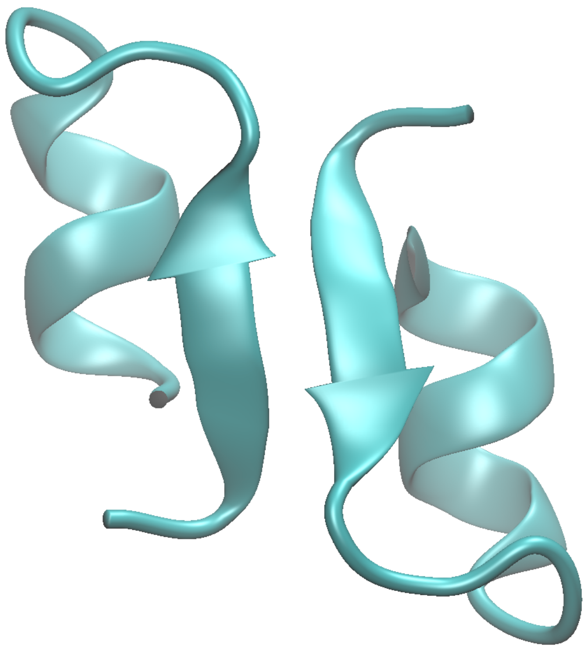
TWO

TUTORIAL 1: ENGENEERED ENDOTHELIN-1 PEPTIDE DIMER

Author Kota Kasahara

2.1 System

In this tutorial, we will analyze the trajectory of a MD simulation of Endothelin-1 dimer, based on PDB ID:1t7h *[Hoh_2004]* . The system is composed of two engineered Endothelin-1 peptides and 7,395 water molecules with 19 and 21 sodium and chrolide ions (150 mM), respectively. The total number of atoms is 22,787.



The MD simulation was perfomed using Gromacs *[Pronk_2013]* . The temperature and pressure of the system were kept at 300 K and 1 atm with Berendsen thermostat and barostat. The length of the trajectory is 100 ns with 10,000 snapshots.

The files for the tutorial are placed at ${MDCCTOOLS}/tutorial_files directory.

- initial.pdb
  - The original PDB file
- initial_md.pdb

  - The .pdb file prepared as the initial structure for the MD simulation.

- traj.trr

  - The trajctory file written in the Gromacs .trr format.

2.2 Preparation

2.2.1 Trajectory file

Our program can read the trajectory files written in Gromacs .trr format. If you have the trajectory file in another format, it has to be converted into the .trr format. *mDCC_tools* include file conversion script, powered by MDAnalysis library *[Michaud-Agrawal_2011]*. For your information VMD plug-in *[Humphrey_1996]* is also useful to convert the trajectory file.

For .ncdf (AMBER) and .dcd (CHARMM, NAMD, and LAMMPS) trajectory files, the following script convert it into .trr file:

```
python ${MDCCTOOLS}/bin/convert_trajectory_mda.py -i traj.dcd   --i-pdb reference.pdb -o traj.trr -f
python ${MDCCTOOLS}/bin/convert_trajectory_mda.py -i traj.ncdf  --i-pdb reference.pdb -o traj.trr -f
```

For trajectory files in the PRESTO format (generated by PRESTO v3, Cosgene and, Psygene-G *[Mashimo_2013]*), the other script is required:

```
python ${MDCCTOOLS}/bin/convert_trajectory_presto.py --i-crd traj.cod  --i-pdb reference.pdb -o traj
```

The RMS fitting should be performed for the trajecotry file because the mDCC analysis depends on the absolute position of atoms. If you use Gromacs tools,

```
trjconv -f traj_orig.trr -o traj_tmp.trr -pbc mol
trjconv -f traj_tmp.trr -o traj.trr -fit rot+trans
```

The fitted trajectory file (traj.trr) is distributed for the tutorial.

2.2.2 Converting the Gromacs trajectory into the mDCC original format

mDCC tools use an original file format of trajectory in order to improve file I/O in the programs. The following command converts the trajectory file, traj.trr

```
cp ${MDCCTOOLS}/tutorial_files/traj.trr .

python ${MDCCTOOLS}/bin/convert_trajectory.py \
  -i traj.trr \
  -o traj.trrmdcc
```

The *–double* option is required for trajectory files recording the real values in double precision.

- -i option specifies input .trr file in Gromacs format
- -o option specifies the name of output trajecotry file

The new file *traj.trrmdcc* will be generated.

2.2.3 Generating the data table

In this process, data tables for information of atoms and residues are generated. This process is required for the data visualizations with R and SQLite in the last part of this tutorial.:

```
cp ${MDCCTOOLS}/tutorial_files/initial_md.pdb .
cp ${MDCCTOOLS}/tutorial_files/initial.pdb .
cp ${MDCCTOOLS}/tutorial_files/segments.txt .

python ${MDCCTOOLS}/bin/get_structure_info.py \
  -i            initial_md.pdb \
  --i-orig      initial.pdb    \
  --i-mdconf    segments.txt   \
  -o            reference.pdb                  \
  --o-pdb-canoresid reference_cano.pdb         \
  --o-atom      ref_atoms.txt                  \
  --o-res       ref_res.txt                    \
  --o-chain     ref_chain.txt                  \
  --o-atom-woh ref_atoms_woh.txt               \
  --o-res-woh  ref_res_woh.txt
```

–i-orig and –i-mdconf are optional.

- *-i initial_md.pdb* is the structure of simulation model in .pdb format.
- *–i-orig initial.pdb* is the struture of the original .pdb file, without hydrogen atoms. In the case that the residue-IDs in the MD initial structure are modified from the original IDs, this setting . The residue numbers are extracted and integrated with the information in *-i initial_md.pdb*. This input file is optional.
- *–i-mdconf segments.txt* is user-defined labels for segments in the structure. In this tutorial, it used for labels of the secondary structures. This file is optional.

  segments.txt:

```
    --segment ANT    0   48   LOOP
    --segment AS1   49   79  SHEET
    --segment AL1   80  137   LOOP
    --segment AH1  138  281  HELIX
    --segment BNT  282  339   LOOP
    --segment BS1  340  360  SHEET
    --segment BL1  361  418   LOOP
    --segment BH1  419  562  HELIX
```

  1. The reserved keyword, "–segment".
  2. Name for this segment (users can arbitraliry define it).
  3. The first atom-ID of this segment.
  4. The last atom-ID of this segment.
  5. Type of this segment (users can arbitrarily define it).

- *-o reference.pdb* is an output .pdb file name.
- *–o-atom ref_atoms.txt* is a table of information about atoms in the structure, written in the tab separated text.

  ref_atoms.txt:

```
    atom_id.int atom_name.string res_name.string res_id.int res_num.int \
        res_num_auth.int chain_id.string segment.string seg_type.string
    0       N       LYS     0       1       1       A       ANT     LOOP
    1       H1      LYS     0       1       1       A       ANT     LOOP
    2       H2      LYS     0       1       1       A       ANT     LOOP
    3       H3      LYS     0       1       1       A       ANT     LOOP
```

The first line indicates the header of each column. From the second line, each line corresponds to each atom.

1. atom_id.int: Automatically assigned atom-ID beginning with zero.
2. atom_name.string: The name of atom taken from initial_md.pdb.
3. res_name.string: The name of residue taken from initial_md.pdb.
4. res_id.int: Automatically assigned residue-ID beginning with zero.
5. res_num.int: Residue number taken from initial_md.pdb.
6. res_num_auth.int: Residue number taken from initial.pdb.
7. chain_id.string: Chain-ID taken from initial_md.pdb
8. segment.string: Segment name defined in segments.txt
9. seg_type.string: Segment type defined in segments.txt

- *–o-res ref_res.txt* is a table of information about residues in the structure, written in the tab separated text.

  ref_res.txt:

```
res_id.int res_num.int res_num_auth.int res_name.string first_atom_id.int \
    last_atom_id.int res_label.string \
    chain_id.string segment.string seg_type.string
0       1       1       LYS     0       23      Lys1    A       ANT     LOOP
1       2       2       ARG     24      47      Arg2    A       ANT     LOOP
2       3       3       CYS     48      57      Cys3    A       ANT     LOOP
```

The first line indicates the header of each column. From the second line, each line corresponds to each residue.

1. res_id.int: Automatically assigned residue-ID beginning with zero.
2. res_num.int: Residue number taken from initial_md.pdb.
3. res_num_auth.int: Residue number taken from initial.pdb.
4. res_name.string: The name of residue taken from initial_md.pdb.
5. first_atom_id.int: The first atom-ID of this residue.
6. last_atom_id.int: The last atom-ID of this residue.
7. res_label.string: The label for this residue.
8. chain_id.string: Chain-ID taken from initial_md.pdb
9. segment.string: Segment name defined in segments.txt
10. seg_type.string: Segment type defined in segments.txt

- *–o-chain ref_chain.txt* is a table of information about chains in the structure, written in the tab separated text.

  ref_chain.txt:

```
chain_id.string chain_type.string first_atom_id.int last_atom_id.int \
    first_res_id.int last_res_id.int
A       PEPTIDE 0       280     0       17
B       PEPTIDE 281     561     18       35
```

The first line indicates the header of each column. From the second line, each line corresponds to each chain.

1. chain_id.string: Chain-ID taken from initial_md.pdb

2. chain_type.string: Type of the chain.
3. first_atom_id.int: The first atom-ID of this chain.
4. last_atom_id.int: The last atom-ID of this chain.
5. first_res_id.int: The first residue-ID of this chain.
6. last_res_id.int: The last residue-ID of this chain.

- *–o-atoms-woh ref_atoms_woh.txt* is the same as *–o-atom ref_atoms.txt* except for absence of the first header line.
- *–o-res-woh ref_res_woh.txt* is the same as *–o-res ref_res.txt* except for absence of the first header line.
- *–o-chain-woh ref_chain_woh.txt* is the same as *–o-chain ref_chain.txt* except for absence of the first header line.

2.3 Pattern recognition

2.3.1 Execute the mdcc_learn program

mdcc_learn program performs a pattern recognition for a spatial distribution of atomic coordinates. Here, we will apply *mdcc_learn* for all heavy atoms. For the system consisting of N_h heavy atoms, *mdcc_learn* should be repeatedly executed N_h times.

Execute the commands from the shell:

```
mkdir mdcclearn_out
mkdir mdcclearn_bash
cp ${MDCCTOOLS}/tutorial_files/mdcclearn_conf_template.txt .
cp ${MDCCTOOLS}/tutorial_files/mdcclearn_crd.bash .
```

mdcc_learn_crd.bash script repeats executions of *mdcc_learn* program.

For example,

```
bash mdcclearn_crd.bash 3 1
bash mdcclearn_crd.bash 3 2
bash mdcclearn_crd.bash 3 3
```

Here we have 307 heavy atoms in the system, and the *mdcc_learn* jobs are divided into three jobs, each of them processes 102 or 103 atoms. The first argument means the total number of divided jobs, and the second means the ID of a job. They can be executed in parallel.

mdcc_learn_crd.bash:

```
#!/bin/bash

n_cal=${1}
id_cal=${2}

echo "${id_cal} / ${n_cal}"

fn_mdcclearn_conf="mdcclearn_conf_template.txt"
fn_mdcclearn_sh="mdcclearn_${id_cal}.bash"
fn_pdb="reference.pdb"

python ${MDCCTOOLS}/run_mdcc_tool.py \
    --mode mdcclearn \
    --pdb ${fn_pdb} \
    --select "not type H" \
    --n-div ${n_cal} \
    --task-id ${id_cal} \
    --mdcc-conf ${fn_mdcclearn_conf} \
    --mdcc-bin "${MDCCTOOLS}/bin/mdcc_learn" \
    --fn-sh mdcclearn_bash/${fn_mdcclern_sh}

cd mdcclearn_bash
bash ./${fn_mdcclearn_sh}
```

- *–select* specifies atoms to be analyzed. The syntax is defined in MDAnalysis library. See the document of MDAnalysis.

For each *mdcc_learn* job, the configure file is loaded.

mdcclearn_conf_template.txt:

```
-atom #{COLUMN}
-n-mixed-element 5
-skip-data 1
-skip-header 10
-fn-data-table  ../traj.trrmdcc
```

- *-n-mixed-element 5* means the number of Gaussian functions to model the distribution.
- *-data-skip 1* means the every frames in the trajectory are sampled. When it is 2, one of every two frames are sampled.
- *-header-skip 10* means the first 10 frames are skipped. They considered as relaxaition steps.
- *-fn-data-table* specifies the input trajectory file in our original format.
- *-fn-out-gaussian* specifies the output file name. The keyword #{COLUMN} is replaced into atom-ID by the python script.

The results of *mdcc_learn* jobs are output at *mdcclearn_out* direcoty. The number at the tail of file name indicates the atom-ID.

mdcclearn_out.txt.0:

```
0       2.0036e-07      0       0       0       0.2     0       0       0       0.2     0       0
1       2.0036e-07      0       0       0       0.2     0       0       0       0.2     0       0
2       2.0036e-07      0       0       0       0.2     0       0       0       0.2     0       0
3       0.00184653      21.1787 36.237  25.1603 35.3337 59.9602 41.5524 59.9602 102.601 71.1896 41.5
4       0.998153        25.2273 39.9415 30.0973 0.893441        0.432562        0.228942        0.43
```

Each column indicate the parameters for each Gaussian function.

- The 1st column is ID of the Gaussian functions.
- The 2nd column is the probability.
- The 3rd-5th columns are the mean of the Gaussian in x, y, and z coordinates.
- The remaining 9 columns indicate the covariance matrix.

2.3.2 Integrating the results of all heavy atoms

After finishing all jobs, all the results are concatenated and global-ID for all Gaussian functions are assigned.

Execute the command from the shell:

```
python ${MDCCTOOLS}/bin/mdcclearn_result_summary.py \
  --dir-mdcclearn mdcclearn_out --pref-mdcclearn mdcclearn_out.txt. \
  -o crd_mdcclearn_gauss.txt \
  --min-pi 0.01
```

- *–min-pi 0.01* means that the Gaussian funcitons probability of which is less than 0.01 will be eliminated

The files named *mdcclearn_out.txt.** in the directory *mdcclearn_out* are merged to a single file *crd_mdcclearn_gauss.txt*.

2.4 Assigning the trajectory on the patterns

In the similar way to the *mdcc_learn* case, *mdcc_assign* program will be executed with dividing into some jobs.

Execute the commands from the shell:

```
mkdir -p mdccassign_bash/assign
cp ${MDCCTOOLS}/tutorial_files/mdccassign_conf_template.txt .
cp ${MDCCTOOLS}/tutorial_files/mdccassign_crd.bash .
bash mdccassign_crd.bash 3 1
bash mdccassign_crd.bash 3 2
bash mdccassign_crd.bash 3 3
```

mdccassign_crd.bash:

```
#!/bin/bash
n_cal=${1}
id_cal=${2}

fn_mdccassign_conf="mdccassign_conf_template.txt"
fn_mdccassign_sh="mdccassign_${id_cal}.bash"
fn_pdb="reference.pdb"

echo "${id_cal} / ${n_cal}"

python2.7 ${MDCCTOOLS}/bin/run_mdcc_tool.py \
  --mode mdccassign \
  --pdb ${fn_pdb} \
  --select "not type H" \
  --n-div ${n_cal} \
  --task-id ${id_cal} \
  --mdcc-conf ${fn_mdccassign_conf} \
  --mdcc-bin "${MDCCTOOLS}/bin/mdccassign" \
  --fn-sh mdccassign_bash/${fn_mdccassign_sh}

cd mdccassign_bash
bash ./${fn_mdccassign_sh}
```

- *–select* specifies atoms to be analyzed.

mdccassign_conf_template.txt:

```
-mode assign-trajtrans
-target-column #{COLUMN}
-skip-data 1
-skip-header 0
-skip-header-gaussian 1
-fn-gaussians ../crd_mdcclearn_gauss.txt
```

```
-fn-interactions  ../traj.trrmdcc
-fn-result assign/assign.dat.#{COLUMN}
-gmm-type #{COLUMN}
```

- The string *#{COLUMN}* will be replaced into the atom-ID by the python script
- *-mode assign-trajtrans* is a reserved keyword. It should not be changed.
- *-target-column* specifies the atom-ID to be processed in the job.
- *-skip-header-gaussian 1* means the first line in *../crd_mdcclearn_gauss.txt* will be omitted.

We will get many binary files *assign.txt.** in the directory *mdccassign_bash/assign*.

2.5 Calculating the mDCC

The correlations of all pairs of modes, which defined with *mdcc_learn* program, will be calculated by using a python script *cal_mdcc.py*. In the same way as execution of *mdcc_learn* and *mdcc_assign*, execute the commands from the shell as following commands:

```
mkdir mdcc
cp ${MDCCTOOLS}/tutorial_files/cal_mdcc.bash .
bash cal_mdcc.bash 3 1
bash cal_mdcc.bash 3 2
bash cal_mdcc.bash 3 3
cat mdcc/corr.txt.* > corr_mdcc.txt
```

Note that this process calculate all pairs of modes, and thus the calculation time depends O(N^2).

cal_mdcc.bash:

```
#!/bin/bash
n_cal=${1}
id_cal=${2}

python2.7 ${MDCCTOOLS}/bin/cal_mdcc.py \
  --fn-ref reference_cano.pdb \
  --gaussian crd_mdcclearn_gauss.txt \
  --pref-assign mdccassign_bash/assign/assign.dat. \
  --fn-crd-bin  traj.trrmdcc \
  --select "not type H" \
  --o-mdcc mdcc/corr.txt.${id_cal} \
  --n-div ${n_cal} \
  --task-id ${id_cal} \
  --assign-binary \
  --range-time-begin 10
```

- *–select “not type H”* specifies the atoms to be considered. All the pairs in these atoms will be calculated. The syntax of this selection string follows MDAnalysis library (http://pythonhosted.org/MDAnalysis).
- *–range-time-begin 10* means the first nine frames of the trajectory will be omitted.

We will obtain *mdcc.txt.** files.

corr_mdcc.txt:

```
0   421 0   512 0.0672200182733 1.0 17.0161954061
0   423 0   514 0.0541268903286 1.0 15.8070676053
421 423 512 514 0.763646415298  1.0 1.29259511836
0   1   0   4   0.873601422704  1.0 1.35943704893
```

1. The 1st and 2nd columns: IDs of modes corresponding to *crd_mdcclearn_gauss.txt*.
2. The 3rd and 4th columns: atom-IDs (zero-origin)
3. The 5th column: the mDCC correlation coefficient
4. The 6th column: simultaneous probability for the two modes.
5. The 7th column: the distance between ceters of the two modes.

2.6 Calculating the conventional DCC

For comparison, the conventional DCC is calculated by using the same script.

Execute the commands from the shell:

```
mkdir dcc
cp ${MDCCTOOLS}/tutorial_files/cal_dcc.bash .
bash cal_dcc.bash 1 1
cat dcc/corr.txt.* > corr_dcc.txt
```

2.7 Calculating the interatomic distance

The initial and averaged distance between atoms are calculated.

Execute the commands from the shell:

```
mkdir dist
cp ${MDCCTOOLS}/tutorial_files/cal_dist.bash .
bash cal_dist.bash 1 1
cat dist/dist.txt.* > dist.txt
cp ${MDCCTOOLS}/tutorial_files/dist_init.bash .
bash dist_init.bash
```

2.8 Gathering the data into SQLite database

The data generated in the previous processes is gatered into a relational database.

First, remove the first line (column headers) from the *crd_mdcclearn_gauss.txt* file.

```
vi crd_mdcclearn_gauss.txt
```

SQL queries are in the files *sqlquery_*.sql*. The bash file *exe_sql.bash* executes the queries and ouput some files for following analyses.

```
mkdir sqlite
cd sqlite
cp ${MDCCTOOLS}/tutorial_files/*sql* .
bash exe_sql.bash
```

The four tab-separated tables are obtained.

- atom_atom.txt
- atom_atom_d5_c50.txt

- res_res.txt
- res_res_d5_c50.txt

Each row records information on a pair of atoms or residues. The files with "_d5_c50" is a subset of the other file, including only pairs that their distance is less than 5.0 Å and their correlation is higher than 0.5.

- *res_num1.int* The ID of first residue.
- *res_num2.int* The ID of sedond residue.
- *atom_id1.int* The ID of first atom.
- *atom_id2.int* The ID of second atom.
- *gauss_id1.int* The ID of the first Gaussian function.
- *gauss_id2.int* The ID of the second Gaussian function.
- *correlation.float* mDCC values between gauss_id1 and gauss_id2.
- *coef.float* The joint probability.
- *dist.float* The distance between centers of gauss_id1 and gauss_id2.
- *corr_dcc.float* The conventional DCC value between atom_id1 and atom_id2.
- *dist_ave.float* The averaged distance between the atoms.
- *dist_sd.float* The standard deviation of the interatomic distance.
- *dist_min.float* The minimum distance between these atoms.
- *dist_max.float* The maximum distance between these atoms.
- *dist_init.float* The initial distance between these atoms.

2.9 Visualizing the data with R

The R software is used for generating figures of the correlation map.

Execute the commands from the shell in the *sqlite* directory:

```
cp ${MDCCTOOLS}/tutorial_files/r_mdcc.r .
R --vanilla --slave < r_mdcc.r
```

The file *mdcc_diff_heatmap.png* will be generated.

2.10 Network analysis

The betweenness values can be calculated by the python script.

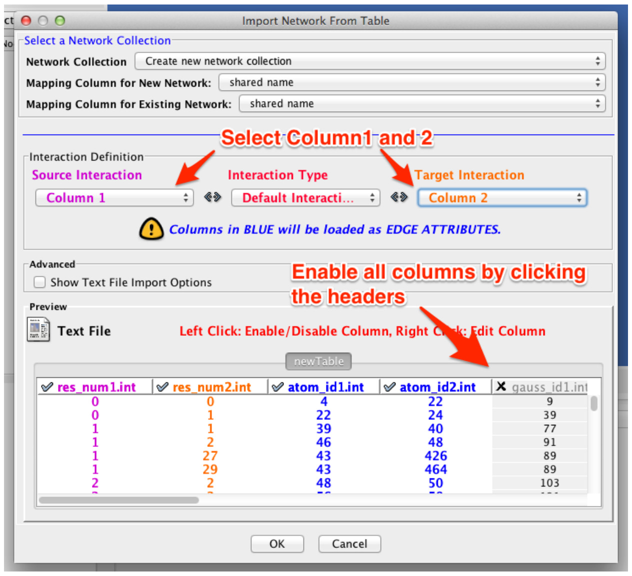
Execute the commands from the shell in the sqlite/ directory:

```
python ${MDCCTOOLS}/bin/nx_centrality.py \
  -i res_res_d5_c50.txt \
  --i-elem ../ref_res.txt \
  --key-elem res_id \
  -o res_cent_btw_d5_c50.txt \
  --btw
```

res_cent_btw_d5_c50.txt is a tab-separated table. Each row indicates each residue.

The files *res_res_d5_c50.txt* and *res_cent_btw_d5_c50.txt* can be loaded with Cytoscape software for analysis.

- Open the file *res_res_d5_c50.txt* from the menu [File] => [Import] => [Network] => [File]



Import Network From Table

Select a Network Collection

Network Collection: Create new network collection
Mapping Column for New Network: shared name
Mapping Column for Existing Network: shared name

Interaction Definition

Source Interaction	Interaction Type	Target Interaction
Column 1	Default Interacti...	Column 2

Columns in BLUE will be loaded as EDGE ATTRIBUTES.

Advanced

Show Text File Import Options

Preview

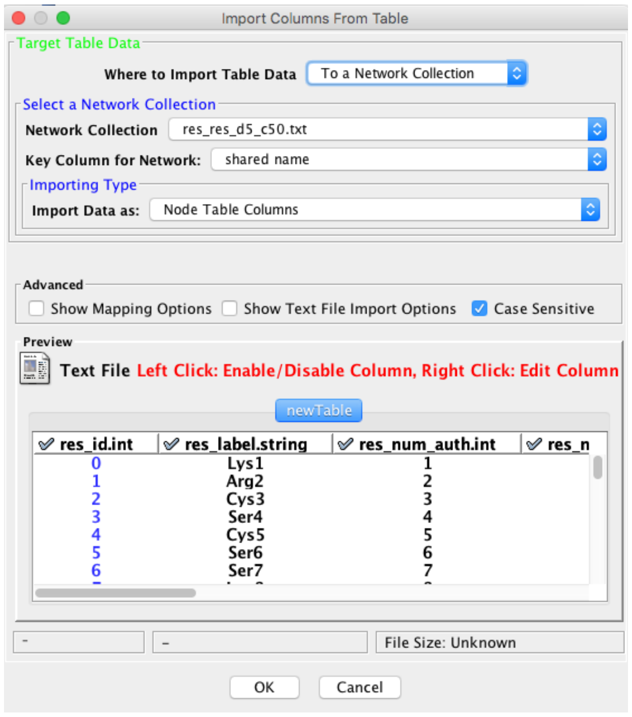
Text File Left Click: Enable/Disable Column, Right Click: Edit Column

newTable

✓ res_num1.int	✓ res_num2.int	✓ atom_id1.int	✓ atom_id2.int	✗ gauss_id1.int
0	0	4	22	9
0	1	22	24	39
1	1	39	40	77
1	2	46	48	91
1	27	43	426	89
1	29	43	464	89
2	2	48	50	103

- Open the file *res_cent_btw_c5_d50.txt* from the menu [File] => [Import] => [Table] => [File]

If you get the error message "Types of keys selected for tables are not matching", specify the type of the first column "res_id.int" as "String" by right-click on the column header.



Import Columns From Table

Target Table Data

Where to Import Table Data: To a Network Collection

Select a Network Collection

Network Collection: res_res_d5_c50.txt
Key Column for Network: shared name

Importing Type

Import Data as: Node Table Columns

Advanced

Show Mapping Options [ ] Show Text File Import Options [ ] Case Sensitive [✓]

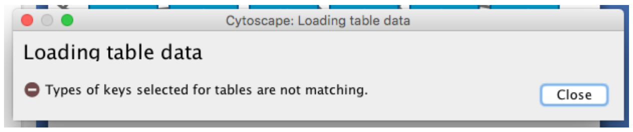
Preview

Text File Left Click: Enable/Disable Column, Right Click: Edit Column

newTable

✓ res_id.int	✓ res_label.string	✓ res_num_auth.int	✓ res_n
0	Lys1	1	
1	Arg2	2	
2	Cys3	3	
3	Ser4	4	
4	Cys5	5	
5	Ser6	6	
6	Ser7	7	

File Size: Unknown

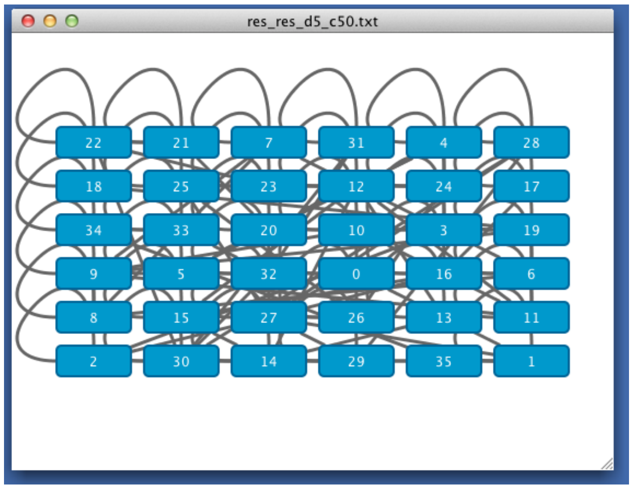


Cytoscape: Loading table data

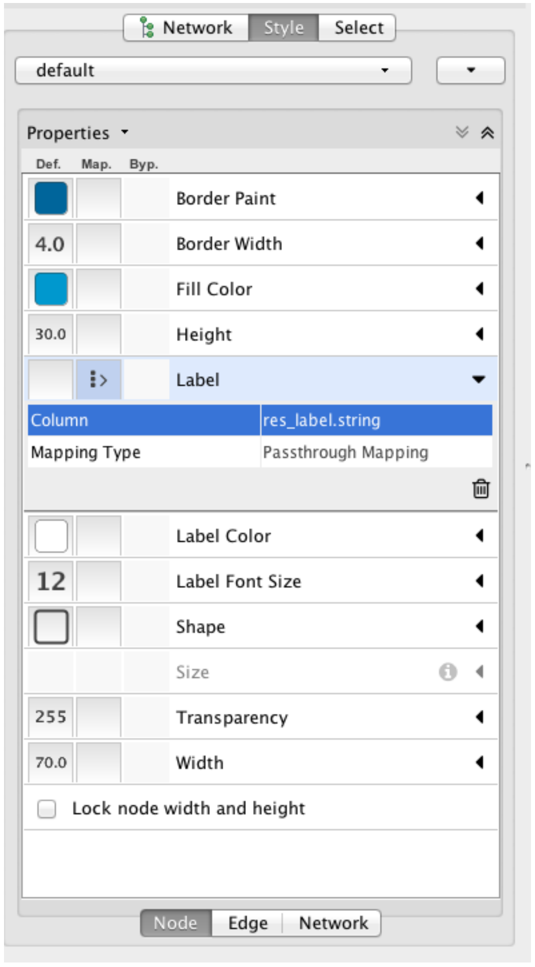
Loading table data

Types of keys selected for tables are not matching.

- Remove the self-loops from the menu [Edit] => [Remove Self-loops]
- Representation of the graph can be modified from the left panel.

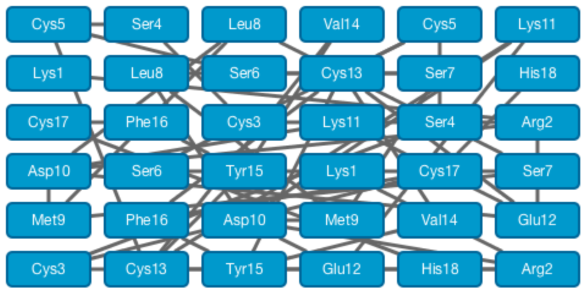


- Choose "Style" tab at the top of the panel.
- In "Node" pane, label on each node can be specified by clicking the row "Label". Here, we choose "res_label.string" as Column, "Passthrough Mapping" as "Mapping Type". See the document of Cytoscape for details.

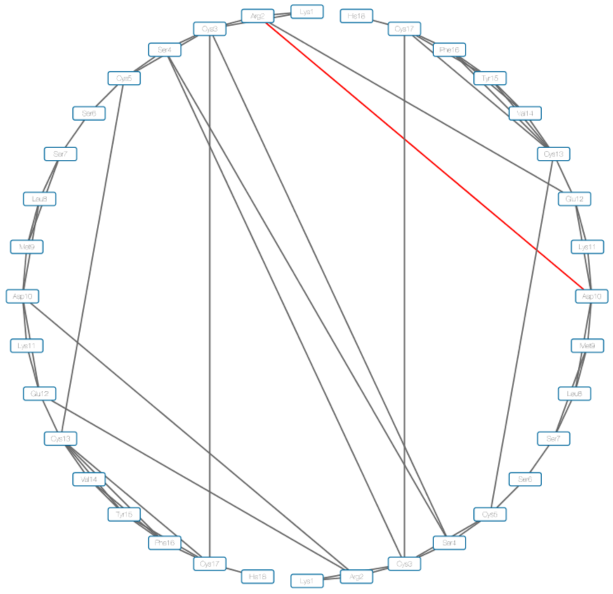


Column	res_label.string
Mapping Type	Passthrough Mapping

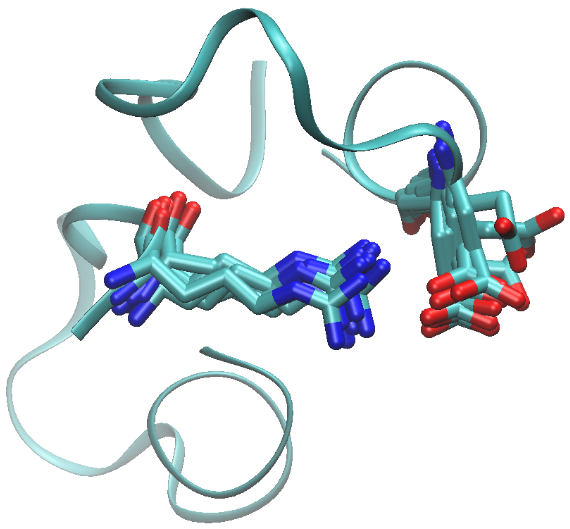
- Layout can be modified from the menu [Layout]



- Here we choose the circle layout.



The edge between Asp10 of A chain and Arg2 in B chain is drawn as red in the figure. The DCC and mDCC values of this edge are 0.31 and 0.55, respectively. mDCC value indicates the highest correlation value in all pairs of modes between these residues. This result means that at least one of these residues shows multi-modal motion, and the interaction between them is transient. The structures of the system along the trajectory show side-chain flipping of Asp10 of A chain.



2.11 About computation times

For your information, the computation times required for this tutorial, with 290 heavy atoms, are,

- mdcc_learn ... 1,213 seconds
- mdcc_assign ... 13 seconds
- cal_mdcc ... 2,575 seconds

Only 1 core of Intel(R) Xeon(R) CPU E5-2670 was used.

CHAPTER

THREE

TUTORIAL 2: ARTIFICIAL 1D DATA

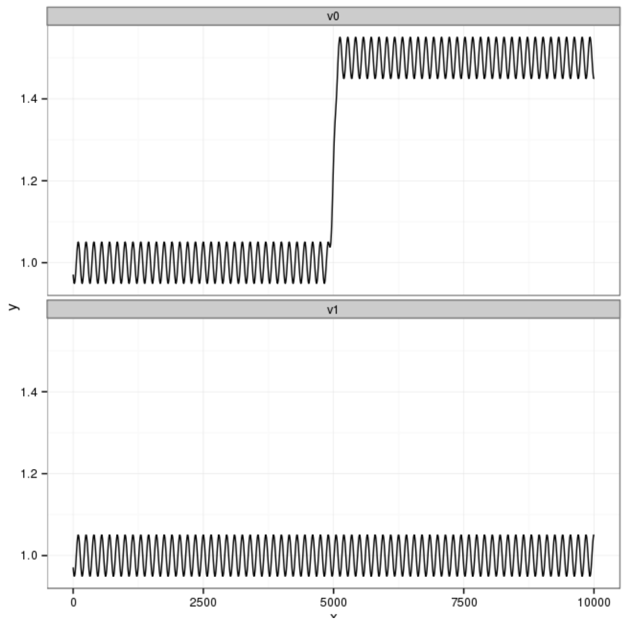
Author Kota Kasahara

The mDCC analysis method can be generalize for analyses of any multi-dimensional numerical data. In this tutorial, we will re-purpose the mDCC analysis method for non-MD application, by using a simple, artificial data.

3.1 Preparation

3.1.1 Artificial data

The two time series of 1D data, v0 and v1, will be generated, by using R.



The data v0 is composed of three sin curves: 1) the first half, 2) the last half, and 3) the middle of them. The phases of the sin functions 1 and 2 are opposite. The function 3 is needed for smoothly concatenating these two functions. The data v1 is a single sin function, which is same as the first half of v0. Thus, the first half of v0 is positively correlated with v1, and the last half is negatively correlated with v1.

To generate the data table, execute the following commands on R:

```
gen.sin <- function(A, B, L, x){
  A * sin( B + 2*pi/L * x)
}
```

```
steps <- 1:10000
v01 <- gen.sin(0.05, 10, 150, steps) + 1
v02 <- gen.sin(-0.05, 10, 150, steps) + 1.5
grad <- (gen.sin(1,  0, 400, 1:400) + 1)[101:300]/2
w01 <- c(rep(1,4900), grad, rep(0,4900))
w02 <- 1-w01
v0 <- v01*w01 + v02 * w02
v1 <- gen.sin(0.05, 10, 150, steps) + 1
dat <- data.frame(v0, v1)
write.table(dat, "dat.txt", quote=F)
```

dat.txt is tab separated table:

```
v0 v1
1 0.971065970962453 0.971065970962453
2 0.969383757588155 0.969383757588155
3 0.967755255506516 0.967755255506516
4 0.966183321663557 0.966183321663557
..
```

3.1.2 Converting the tab separated table into the mDCC original format

The data table should be converted into the original binary file:

```
python ${MDCCTOOLS}/bin/convert_trajectory.py \
  -i dat.txt \
  --i-format tsv \
  -o traj.trrmdcc \
  --ignore-col 1 \
  --ignore-row 1 \
  --dim 1
```

- *-i* option specifies the input data table.
- *–i-format* option indicates the data format of -i file. It should be “tsv” or “trr”.
- *-o* option indicates the output file name.
- *–ignore-col, –ignore-row* options indicate how many columns and rows are skipped in the -i tsv table, respectively.
- *–dim* option specifies dimension of the data.

3.2 Pattern recognition

3.2.1 Execute the mdcc_learn program

mdcc_learn program will be executed two times. One is for the entity v0, and the other is for v1.

The input files for v0 is:

```
-feature 1 0 x
-n-mixed-element 5
-fn-data-table  traj.trrmdcc
-format-data-table mdcc
-fn-out-gaussian mdcclearn_out.txt.0
```

The other is:

```
-feature 1 1 x
-n-mixed-element 5
-fn-data-table  traj.trrmdcc
-format-data-table mdcc
-fn-out-gaussian mdcclearn_out.txt.1
```

- *-fn-data-table* specifies the file name of the trajectory data.
- *-format-data-table* specifies the file format of –fn-data-table. It should be "mdcc" or "tsv"
- *-fn-out-gaussian* specifies the output file name. This file describes the inferred Gaussian mixture parameters.
- *-feature* requires three arguments.
  - The first is always 1. This field will be used for future developments.
  - The second specifies 0-origin ID of entity (atom) in the data file.
  - The third is the name of this feature. Any string is acceptable.
- *-n-mixed-element* is the number of Gaussian functions in the Gaussian mixture model.

Then, *mdcc_learn* is executed from the linux shell:

```
${MDCCTOOLS}/bin/mdcc_learn -fn-cfg mdcclearn_v0.cfg
${MDCCTOOLS}/bin/mdcc_learn -fn-cfg mdcclearn_v1.cfg
```

3.2.2 Integrating the results of all heavy entities

After that, the results of these two entities are concatenated and global-ID for all Gaussian functions are assigned.

Execute the command from the shell:

```
python ${MDCCTOOLS}/bin/mdcclearn_result_summary.py \
  --dir-mdcclearn ./ \
  --pref-mdcclearn mdcclearn_out.txt. \
  -o crd_mdcclearn_gauss.txt \
  --dim 1 --min-pi 0.01
```

- *–min-pi 0.01* means that the Gaussian funcitons probability of which is less than 0.01 will be eliminated

The files named *mdcclearn_out.txt.** in the directory *mdcclearn_out* are merged to a single file *crd_mdcclearn_gauss.txt*.

3.3 Assigning the trajectory on the patterns

Next, *mdcc_assign* is executed for the two entity, by using the following settings

mdccassign_v0.cfg:

```
-mode assign-mdcctraj
-target-column 0
-skip-header-gaussian 1
-fn-gaussians crd_mdcclearn_gauss.txt
-fn-data-table  traj.trrmdcc
-fn-result assign.dat.0
-gmm-type 0
-format-output binary
```

mdccassign_v1.cfg:

```
-mode assign-mdcctraj
-target-column 1
-skip-header-gaussian 1
-fn-gaussians crd_mdcclearn_gauss.txt
-fn-data-table  traj.trrmdcc
-fn-result assign.dat.1
-gmm-type 1
-format-output binary
```

- *-mode* specifies "assign-mdcctraj" or "assign-table". The latter is for .tsv input file.
- *-target-column* indicates the 0-origin ID of entity (atom) in the input data file.
- *-skip-header-gaussian* specifies the number of lines to be skipped in the Gaussian definition file (-fn-gaussians)
- *-fn-gaussians* specifies the file name of Gaussian definition file obtained from the previous step.
- *-fn-result* specifies the file name of output.
- *-gmm-type* specifies the ID of Gaussian mixture, corresponding to the second column in -fn-gaussians.

Execute the commands from the shell:

```
${MDCCTOOLS}/bin/mdcc_assign -fn-cfg mdccassign_v0.cfg
${MDCCTOOLS}/bin/mdcc_assign -fn-cfg mdccassign_v1.cfg
```

3.4 Calculating the mDCC and DCC

mDCC and DCC should be calculated as following commands.

For mDCC:

```
python2.7 ${MDCCTOOLS}/bin/cal_mdcc.py \
  --gaussian crd_mdcclearn_gauss.txt \
  --pref-assign assign.dat. \
  --suff-assign "" \
  --o-mdcc mdcc.txt \
  --min-corr 0.0 \
  --select-id 0-1 \
  --fn-crd-bin traj.trrmdcc \
  --assign-binary
```

For DCC:

```
python2.7 ${MDCCTOOLS}/bin/cal_mdcc.py \
  --o-dcc dcc.txt \
  --min-corr 0.0 \
  --select-id 0-1 \
  --fn-crd-bin traj.trrmdcc
```

- *–select-id* specifies the 0-origin entity ID, which are analyzed. The range of IDs should be specified by concatenating the first and last IDs with '-'. Or, the IDs and the range of IDs can be enumerated by sepalation with ',', e.g., "1-10,12,14,16-18".

–pref-assign, –suff-assign indicates prefix and suffix of *mdcc_assign* output files. The file must be named with the prefix, IDs of elements, and suffix, e.g., "prefix.0.suffix", "prefix.1.suffix", ... * *–assign-binary* is required for the binary *mdcc_assign* output. * *–o-mdcc, –o-dcc* are the output file name. * *–fn-crd-bin* is the input binary file. * *–min-corr* indicates the minimum correlation coefficient for output.

3.5 Results

crd_mdcclearn_gauss.txt:

```
gc_id.int element_id.int pi.float mu1.float sigma11.float
0       0       0.496002        1.49895 0.00199557
1       0       0.49639 1.00054 0.00170489
2       1       1       0.99981 0.00145084
```

The *mdcc_learn* program found two and one Gaussian functions for the data v0 and v1, respectively.

mdcc.txt:

```
0       2       0       1       -0.90211646445  0.500272653972  0.499140297242
1       2       0       1       0.875547919424  0.499727346028  0.000772943337186
```

- The 1st and 2nd columns indicate the pair Gaussian IDs.
- The 3rd and 4th columns indicate the element IDs.
- The 5th column indicates the mDCC value.
- The 6th column indicates the simultaneous probability for the Gaussians.
- The 7th column indicates the distance between the means of the Gaussian functions.

dcc.txt:

```
0       1       0.00363216      0.24999
```

- The 1st and 2nd columns indicate the pair of element IDs.
- The 3rd column indicates the DCC value.
- The 4th column indicates the distance between the means.

As we expected, these results say that v1 positively correlated with the Gaussian 1 of v0, but it negatively correlated with the Gaussian 0 of v0. On the other hand, the conventional DCC shows no correlation between v0 and v1.

With this protocol, any kinds of multi-dimensional numerical data can be analyzed by using this tool kit.

CHAPTER

FOUR

DATA FORMAT

Author Kota Kasahara

4.1 .trrmdcc format

The original trajectory file format *.trrmdcc*, which can be converted from Gromacs .trr format by using *convert_trajectory.py* script, is used in the mDCC tools.

The first 16 bytes record four inter values:

```
* 1-4: the reserved number "1993"
* 5-8: the size of a real value, 4 or 8
* 9-12: the number of atoms : Na
* 13-16: the number of frames : Nf
* 17-20: the dimension of data vector : D (for MD trajectory, D=3)
```

The remaining part records D * Na * Nf real values indicating the x,y,z coordinates of atoms in each time.

The order of values are,

- x(0,0,0), x(0,0,1), x(0,0,2), ..., x(0,0,Nf),
- x(1,0,0), x(1,0,1), x(1,0,2), ..., x(0,0,Nf),
- ...
- x(D,0,0), x(D,0,1), x(D,0,2), ..., x(D,0,Nf),
- x(0,1,0), x(0,1,1), x(0,1,2), ..., x(0,1,Nf),
- ...
- x(D,Na,0), x(D,Na,1), x(D,Na,2), ..., x(D,Na,Nf)

where, x(d,i,j) means d-th dimension coordinate of i-th entity (atom) at j-th sample (time).

CHAPTER

FIVE

APPENDIX: ADDITIONAL INFORMATION FOR THE INSTALLATION

Author Neetha Mohan

5.1 Installation of various dependencies for mDCC

5.1.1 BLAS (Basic Linear Algebra Subprograms)

BLAS is a standard interface for operations like matrix multiplication. The software package is available for download from http://www.netlib.org/blas/ (direct link: http://www.netlib.org/blas/blas.tgz).

- Download the source code.
- Untar it into an empty directory:

  ```
  gunzip blas.tgz
  tar xf blas.tar
  cd BLAS
  ```

  Now you now have a bunch of .f files. With these Fortran files, you can create either a static library or a shared library.

- A shared library libblas.so is built with following command:

  ```
  gfortran -shared -O2 *.f -o libblas.so -fPIC
  # Replace -O3 with your favorite optimization options.
  ```

- A static library is built with the two following commands:

  ```
  su -c “cp libblas.a /usr/local/lib”
  gfortran -O2 -c *.f
  ar cr libblas.a *.o
  ```

- Configuration

  The library is in the BLAS directory, called ‘blas_LINUX.a’. Copy it where you prefer and set the correct PATH where you want to use it.

5.1.2 LAPACK (Linear Algebra PACKage)

LAPACK, is a standard collection of routines, built on BLAS, for more-complicated linear algebra operations like matrix inversion and diagonalization. LAPACK is available from http://www.netlib.org/lapack/.

- Download the source code.
- Untar the file

  tar zxf lapack.tgz
- Copy and edit the file LAPACK/make.inc.example to LAPACK/make.inc

  cp LAPACK/make.inc.example LAPACK/make.inc
- Edit the file LAPACK/Makefile and type 'make'.

  The LaPack library uses the BLAS library, so you need to tell where to find it. The result is a library 'lapack_LINUX.a'. This can be copied in a place of your choice.

5.1.3 MDAnalysis

MDAnalysis is an object-oriented python toolkit to analyze molecular dynamics trajectories in many popular formats. MDAnalysis allows one to read molecular dynamics trajectories and access the atomic coordinates through NumPy arrays. This provides a flexible and relatively fast framework for complex analysis tasks. MDAnalysis is available from http://www.mdanalysis.org/. MDAnalysis can be installed using:

- Using pip

  pip install MDAnalysis
- Installing from source

  It is also possible to build and install from source. The source packages can be downloaded either from PyPI repositories or Git repository. To download from PyPI repositories go to:

  https://pypi.python.org/pypi/MDAnalysis (main package)

  https://pypi.python.org/pypi/MDAnalysisTests (for the tests) and download the tar files from there.
- Alternatively, the source packages can be downloaded from the git repository at https://github.com/MDAnalysis/mdanalysis.

  In most cases simply do

  git clone https://github.com/MDAnalysis/mdanalysis

  cd mdanalysis
- Standard installation from source

  Steps for installing in your home directory:

  python setup.py build

  python setup.py install –user

  It is also possible to use –prefix, –home, or –user options for setup.py to install in a different (probably your private) python directory hierarchy.

5.1.4 NumPy and SciPy

NumPy is the fundamental package for scientific computing with Python. SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. NumPy is available from http://www.numpy.org and SciPy from http://www.scipy.org/.

- The source package for NumPy can be obtained from:

  git clone git://github.com/numpy/numpy.git numpy or

  git clone http://github.com/numpy/numpy.git numpy
- The source package for SciPy can be obtained from:

  git clone git://github.com/scipy/scipy.git scipy or

  git clone http://github.com/scipy/scipy.git scipy
- To build NumPy/SciPy from source, get the source package, unpack it, and:

  python setup.py install –user # installs to your home directory

  or

  python setup.py build

  python setup.py install –prefix=$HOME/local
- An altermate way to install the packages of the SciPy stack is to download one of these Python distributions, which includes all the key packages like,

  Anaconda: A free distribution for the SciPy stack.

5.1.5 NetworkX

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. NetworkX is available from http://networkx.github.io

- Before installing NetworkX, you need to have setuptools installed.

  Download ez_setup.py and run it using the target Python version. The script will download the appropriate version and install it for you:

  wget https://bootstrap.pypa.io/ez_setup.py -O - | python

  Note that you will may need to invoke the command with superuser privileges to install to the system Python:

  wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python
- Alternatively, Setuptools may be installed to a user-local path:

  wget https://bootstrap.pypa.io/ez_setup.py -O - | python - –user
- Get NetworkX from the Python Package Index at http://pypi.python.org/pypi/networkx or install it with

  pip install networkx

  You can install the development version (at github.com) with

  pip install git://github.com/networkx/networkx.git#egg=networkx
- Installing from source

  You can install from source by downloading a source archive file (tar.gz or zip) or by checking out the source files from the Mercurial source code repository.

  Download the source (tar.gz or zip file) from https://pypi.python.org/pypi/networkx/ or get the latest development version from https://github.com/networkx/networkx/

  Unpack and change directory to the source directory (it should have the files README.txt and setup.py).

Run python setup.py install to build and install

(Optional) Run nosetests to execute the tests if you have nose installed.

5.1.6 R

R is a free software environment for statistical computing and graphics.

- Download the program source code and install

  http://lib.stat.cmu.edu/R/CRAN/

  tar xzvf R-1.4.1.tgz

  ./configure

  make

  make check # to ensure that the program was actually built correctly.

- Install packages

  Eg: install.packages("ggplot2")

[Kasahara_2014] Kasahara, K., Fukuda, I., & Nakamura, H. (2014). A Novel Approach of Dynamic Cross Correlation Analysis on Molecular Dynamics Simulations and Its Application to Ets1 Dimer-DNA Complex. PLoS ONE, 9(11), e112419. http://doi.org/10.1371/journal.pone.0112419

[Hoh_2004] Hoh, F.,Cerdan, R.,Kaas, Q.,Nishi, Y.,Chiche, L.,Kubo, S.,Chino, N.,Kobayashi, Y.,Dumas, C.,Aumelas, A. High-resolution X-ray structure of the unexpectedly stable dimer of the [Lys(-2)-Arg(-1)-des(17-21)]endothelin-1 peptide, Biochemistry, 43:15154-15168, 2004

[Pronk_2013] Pronk S, Páll S, Schulz R, Larsson P, Bjelkmar P, et al. (2013) GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. Bioinformatics 29: 845–854. doi:10.1093/bioinformatics/btt055.

[Michaud-Agrawal_2011] Michaud-Agrawal N, Denning EJ, Woolf TB, Beckstein O (2011) MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. J Comput Chem 32: 2319–2327. doi:10.1002/jcc.21787.

[Humphrey_1996] Humphrey W, Dalke A, Schulten K (1996) VMD: visual molecular dynamics. J Mol Graph 14: 33–8–27–8.

[Mashimo_2013] Mashimo T, Fukunishi Y, Narutoshi K, Takano Y, Fukuda I, et al. (2013) Molecular Dynamics Simulations Accelerated by GPU for Biological Macromolecules with a Non-Ewald Scheme for Electrostatic Interactions. J Chem Theory Comput 9: 5599–5609. doi:10.1021/ct400342e.