PyPAT Documentation

Release 1.0

Michael G. Lerner, Steven A. Spronk, Heather A. Carlson

October 14, 2008

CONTENTS

CHAPTER

ONE

PyPAT

PyPAT (Python-based Protein Analysis Tools) is a collection of tools that build upon the ptraj module of AMBER and the PyMOL visualization package to aid in the analysis of protein structures and molecular dynamics trajectories. They allow for the evaluation of the convergence of trajectories, as well as the examination of correlated dynamics, hydrogen bonds, and bridging-water molecules throughout a trajectory. Our tools are written in Python and released under an open-source license.

This document is intended to explain the usage of the PyPAT tools. For an understanding of the theory behind, and implementation of, these tools, users are refered to the paper:

Lerner, M.G., Spronk, S.A., and Carlson, H.A. (2008) PyPAT: a Python-based toolset to aid in the analysis of protein structures and trajectories. *submitted to Bioinformatics*.

CHAPTER

TWO

System Requirements

Our tools are primarily meant for use with the AMBER suite of programs, and they have been extensively tested on both OS X and Linux. Many may be installed on Windows systems, but this is generally unsupported.

Required software includes

- gnuplot 4.2 (www.gnuplot.info)
- ImageMagick 6.4.3 (www.imagemagick.org),
- matplotlib 0.98.3 (matplot-lib.sourceforge.net)
- numpy 1.1.1 (numpy.scipy.org)
- PyMOL 1.1 (www.pymol.org)
- Python 2.6 (www.python.org)

all of which are freely available and open-source.

CHAPTER

THREE

Installation

Installation on Linux and OS X follows standard Python proctocols:

```
prompt$ python setup.py build
prompt$ python setup.py install
```

Users may easily install into a local directory with the `-prefix` option:

```
prompt$ python setup.py install --prefix=/my/home/dir/software
```

CHAPTER

FOUR

Documentation

Documentation is provided in ReStructured Text formatted text files in the top-level directory. Sphinx has been used to generate html and pdf documentation from these text files. The html and pdf files live in the `Doc` subdirectory. If the need arises, documentation may be rebuilt:

```
prompt$ make clean
prompt$ make html
prompt$ make latex
prompt$ cd .build/latex
prompt$ make all-pdf
prompt$ cd ../..
prompt$ rm -rf Doc/html
prompt$ mv .build/html Doc
prompt$ mv .build/latex/PyPAT.pdf Doc
```

CHAPTER

FIVE

Graphical display of MD properties over time

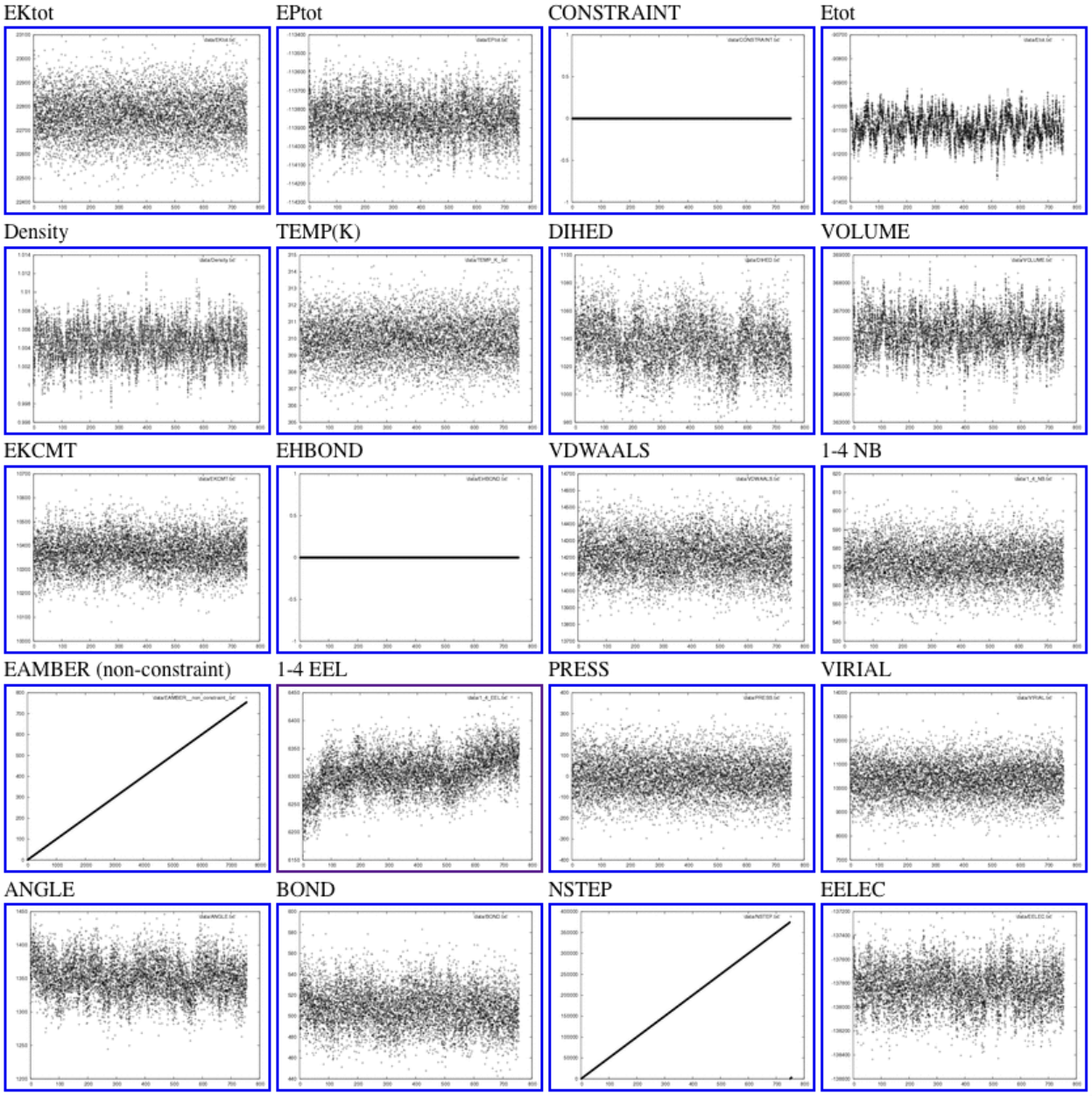
5.1 parse_sander_output.py

Analyzing MD simulations requires assessment of the convergence of dynamic properties, such as temperature, pressure, total energy, potential energy, kinetic energy, and various error estimates. This assessment is facilitated with PyPAT. The AMBER MD engine, sander, outputs the dynamic property information to a text file.

`parse_sander_output.py` parses this file and compiles data of the relevant quantities vs. time in tab-delimited files. It then generates graphical images of the time-evolution of the properties and an html file that shows thumbnails of all these images, nested with links to larger versions.

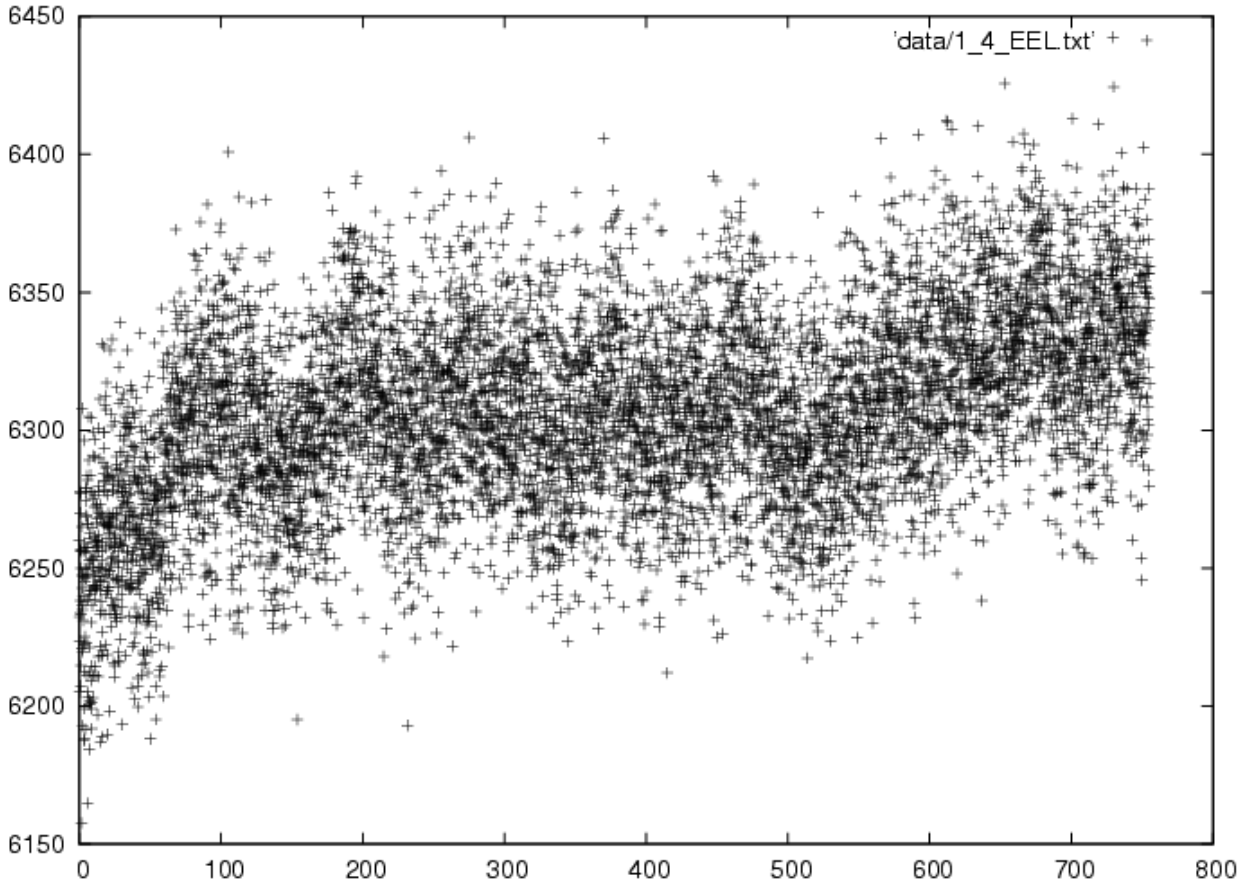
5.1.1 HTML overview

The HTML overview from a short MD simulation looks like:



5.1.2 Larger images

Clicking on individual thumbnails leads to larger images like:



5.1.3 Underlying data

One more click shows the underlying data in a tab-delimited format easily read by Excel and other scripts:

```
# TIME(PS)      1-4 EEL
0.002    6260.1304
0.1      6232.261
0.2      6205.0433
0.3      6223.5566
0.4      6233.5902
0.5      6277.6666
0.6      6270.031
0.7      6238.2656
0.8      6276.9254
0.9      6256.237
1.0      6207.3524
1.1      6214.5952
1.2      6256.6032
1.3      6250.0197
1.4      6233.2002
1.5      6157.5829
1.6      6256.233
```

5.2 Documentation from the script

```
Usage:
parseSanderOut.py -f file.out -d dirname

will put the data in file.out into nice files in dirname.
The directory called dirname must not exist when this script is run.
Among those files are:

  data/allout.txt      tab-delimited text file with all information
                       (suitable for gnumeric, koffice or excel)
  data/Etot.txt, etc.  individual files with different types of sander
                       output (two columns per file, one is time(ps))
  results.html         html file showing you lots graphs of your data
  images/*             postscript and gif graphs of your data

So, for the most part, you probably just want to point your favorite
webbrowser at dirname/results.html.

NOTE: this script assumes that you have gnuplot and convert installed
and in your path.


Options:
  -h, --help            show this help message and exit
  -f DATAFILENAME, --file=DATAFILENAME
                        The name of the sander output file to parse [default
                        sander.out]
  -d OUTDIR, --dir=OUTDIR
                        The name of the output directory (must not exist)
                        [default data]
```

CHAPTER

SIX

Bridging-water analysis

6.1 Executive Summary

An understanding of bridging-water molecules is critical in the study of structure, dynamics, and function of proteins and nucleic acids. While several programs (such as ptraj) allow for ane analysis of hydrogen bonds, the analysis of bridging waters is significantly more tedius. Typically, users must extract this information either through detailed examination of hydrogen bonds between pairs of protein atoms and water atoms, or through visual examination of MD trajectories.

PyPAT greatly simplifies this process, providing two scripts (`collect_water_bridges.py` and `display_bridging_interactions.py`) that extract and analyze bridging interactions throughout an MD trajectory.

A typical invocation for a trajectory and topology named `mysim.trj` and `mysim.top` might look like:

```
prompt$ pymol -qcr collect_water_bridges.py -- --name=mysim
prompt$ display_bridging_interactions.py --name=mysim --resi-criteria 5,6,10-24
```

These scripts are able to analyze proteins, nucleic acids, and custom-defined ligands. In order to simplify the wording below, all of these will be referred to as "protein".

6.2 Phase 1 (collect_water_bridges.py)

6.2.1 Description

The first step is the most time consuming: extracting the bridging information from the MD simulation. Using PyMOL as a backend, all water molecules for which the Oxygen is within 4.0 Angstroms of the protein are examined. Information about the distance and angle of each interaction between these molecules and the protein is recorded. This information can be further refined in the next step, so we recommend using loose constraints; this helps avoid the need to rerun `collect_water_bridges.py`.

We note again that this script must be run through PyMOL:

```
pymol -qcr path/to/collect_water_bridges.py -- --name=mysimulation
```

etc. The – after the script name is required.

The default options for distance and angle cutoffs are usually correct. Users must explicitly specifiy the number of steps in the MD trajectory (`-num-steps`), as well as the name of the trajectory (`-name`). The trajectory and topology file must be named `<name>.top` and `<name>.trj` respectively.

PyMOL slows down if it processes too many frames at once. Therefore, the trajectory is analyzed in chunks of 500 frames at a time. We find this to be a generally useful chunk-size, but users can change it via the `chunk-size` command-line option.

6.2.2 Defining new ligands

The script comes with definitions for standard protein and nucleic acid hydrogen-bonding interactions. Users may wish to change these, or to add definitions for new ligands. This is easily accomplished by editing the file `hbond_definitions.py` (installed under `pypat/hbond/` when the scripts are installed). As an example, here is the function that defines donors and acceptors for NADPH:

```
def select_nap_donors_and_acceptors():
    """
    Ligand specific selections for NADPH (NAP)
    """
    #-- NADPH
    #acceptor mask :NAP@N6A  :NAP@H61
    cmd.select("prot_donors","prot_donors or (resn NAP and name H61)")
    #acceptor mask :NAP@N6A  :NAP@H62
    cmd.select("prot_donors","prot_donors or (resn NAP and name H62)")
    #acceptor mask :NAP@O'A3 :NAP@HOA3
    cmd.select("prot_donors","prot_donors or (resn NAP and name HOA3)")
    #acceptor mask :NAP@O'N3 :NAP@HON3
    cmd.select("prot_donors","prot_donors or (resn NAP and name HON3)")
    #acceptor mask :NAP@O'N2 :NAP@HON2
    cmd.select("prot_donors","prot_donors or (resn NAP and name HON2)")
    #acceptor mask :NAP@N7N  :NAP@H72
    cmd.select("prot_donors","prot_donors or (resn NAP and name H72)")
    #acceptor mask :NAP@N7N  :NAP@H71
    cmd.select("prot_donors","prot_donors or (resn NAP and name H71)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name N1A)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name N3A)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name N7A)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OA23)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OA22)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OA24)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'A2)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'A3)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'A4)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'A5)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OPA1)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OPA2)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OPN1)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O3P)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name OPN2)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'N5)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'N4)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'N3)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O'N2)")
    cmd.select("prot_acceptors","prot_acceptors or (resn NAP and name O7N)")
```

After defining such a function, it must be added to the `do_standard_selections` function at the bottom of `hbond_definitions.py`.

6.2.3 Documentation from the script

```
Usage:
    Run this like:

    pymol -qcr collect_water_bridges.py -- --name=myprotein --dist-cutoff=3.5

    Do not forget the double dashes after the script name.


Options:
  -h, --help            show this help message and exit
  -n NAME, --name=NAME  Trajectory and topology must be named name.trj and
                        name.top respectively. [default: nrna]
  -c CHUNKSIZE, --chunk-size=CHUNKSIZE
                        How many MD steps to process at a time. If you do too
                        many at a time, PyMOL will slow down. Too few, and
                        you're wasting time starting/stopping PyMOL. [default:
                        500]
  -s NUMSTEPS, --num-steps=NUMSTEPS
                        number of steps in your MD trajectory. [default: 5000]
  -d DISTCUTOFF, --dist-cutoff=DISTCUTOFF
                        Heavy atom to heavy atom distance cutoff. [default:
                        4.0]
  -a ANGLECUTOFF, --angle-cutoff=ANGLECUTOFF
                        Angle cutoff. If heavy:hydro:heavy angle must be
                        greater than this. [default: 0.0]
```

6.3 Phase 2 (display_bridging_interactions.py)

6.3.1 Description

Phase 1 records data in a water-centric format. That is, interactions are recorded and described for each water molecule. Phase 2 inverts this, and displays bridging interactions as protein-water-protein triplets. There are several subtleties involved in this process, and users are strongly advised to read the paper. Among the relevant options are:

minrequireddwelltime Bridging interactions that do not persist for at least this long are ignored

looseness This allows for gaps in the occupancy of a bridging interaction. Suppose a bridging interaction is present for 300 picoseconds, absent for 2 picoseconds, and present for another 200 picoseconds. If the looseness is greater than or equal to 2 picoseconds, this will be treated as a single 502 picosecond interaction.

minocc Bridging interactions that are absent for substantial percentages of the trajectory can be automatically filtered out.

resi-criteria A complete list of bridging interactions will contain an overwhelming amount of information, including many surface interactions in regions that may not be interesting to the user. With this option, a user can specify a list of residues of interest. Bridging interactions that do not involve at least one of these residues are ignored. The input format is fairly general (e.g. "5,6,7-12"). Users who are comfortable with Python can find examples in the code (`tool_utils.py`) of how to define residue groups such as "loop A", etc.

6.3.2 Documentation from the script

```
Usage: display_bridging_interactions.py [options]

Options:
  -h, --help            show this help message and exit
  -n NAME, --name=NAME  Trajectory and topology must be named name.trj and
                        name.top respectively. [default: nrna]
  -t TIMESTEP, --timestep=TIMESTEP
                        trajectory timestep in picoseconds. [default: 5]
  -m MINREQUIREDDWELLTIME, --min-dwell-time=MINREQUIREDDWELLTIME
                        minimum required dwell time. [default: 3]
  -l LOOSENESS, --looseness=LOOSENESS
                        looseness. [default: 2]
  -d DISTCUTOFF, --dist-cutoff=DISTCUTOFF
                        Heavy atom to heavy atom distance cutoff. [default:
                        3.5]
  -a ANGLECUTOFF, --angle-cutoff=ANGLECUTOFF
                        Angle cutoff. If heavy:hydro:heavy angle must be
                        greater than this. [default: 0.0]
  -o MINOCC, --min-occ=MINOCC
                        Minimum percentage of the trajectory for which this
                        interaction must be occupied. [default: 0.4]
  -r DIR, --dir=DIR     Directory in which the name_hbond_*_*.txt files are
                        located. [default: .]
  -R RESI_CRITERIA, --resi-criteria=RESI_CRITERIA
                        Restrict the output to BWIs where at least one side
                        involves a residue in this list. [default: none]
```

CHAPTER

SEVEN

Hydrogen Bonding

7.1 Executive Summary

A commonly performed analysis of molecular dynamics trajectories is hydrogen bond analysis, in which statistics of all the hydrogen bonds in a system (occupancy, lifetime, distance, etc.) are monitored throughout the simulation. However, this type of analysis is difficult for two reasons: it requires very large amounts of computer memory, and the abundance of data obtained can be overwhelming. The PyPAT H-bond analysis tools provide solutions to these issues.

The tools come as a suite of four programs: `setup_hbond_ptraj.py`, `combine_hbonds.py`, `subset_hbonds.py`, and `compare_hbonds.py`. The first two programs provide a workaround to the memory issue, and the last two provide an easy way to comb through and sort the data.

The programs make use of the trajectory analysis program of the AMBER software package, ptraj. Because of memory requirements, ptraj can only be used to perform a global hydrogen-bonding analysis on a reasonably-sized system for short lengths (several hundred ps) of a trajectory. In order to analyze a long trajectory, we first divide it into short segments, use ptraj to analyze each one, and then recompile the data into a unified set. These tasks are accomplished by `setup_hbond_ptraj.py` and `combine_hbonds.py`. The other two programs provide methods to select a desired subset of data, sort it, and output it in a clean-looking format.

Typically, a series of commands similar to the following will be used in the analysis:

```
$ setup_hbond_ptraj.py  -x 1sgz.mdcrd -p 1sgz.prmtop -b 1 -e 5000 -g 500 -n 389

$ ./run_hbond_ptraj

$ combine_hbonds.py segment*.out -p 1sgz.prmtop -r 4 -o combined1.out

$ subset_hbonds.py combined1.out -R 2-6,9-12 -O 60 -y -c

$ compare_hbonds.py combined1.out combined2.out -i 1sgz,1w50 -R 9-27 -A bb_only -O 20 -s donor -y -c
```

The operation of each of these programs is detailed below. We note that most of these scripts take the `-D` option, which tells the scripts that the data files live in a different directory than the current one.

7.2 setup_hbond_ptraj.py

This program prepares a series of input scripts that can be run by ptraj as well as a script that will automatically perform the execution of ptraj with each one.

Documentation from the script:

```
Usage: setup_hbond_ptraj.py [options]
This program sets up ptraj input files to perform a series of H-bond
analyses on a trajectory.  The trajectory is broken into segments of a
specified size.  Files written out include a ptraj input file for each
segment and a file "run_hbond_ptraj" that will sequentially execute
ptraj with each one.

The only required option is --mdcrd-file (-x), which specifies the
coordinate file.

Important notes:
* A file called "mask" can be created to include hydrogen-bonding
  residues other than the standard amino acids and water.  The lines in
  mask will be copied "as is" into each ptraj input file, so follow the
  format for specifying hydrogen bond donors and acceptors in the ptraj
  documentation.

Options:
  -h, --help            show this help message and exit
  -x MDCRD_FILE, --mdcrd-file=MDCRD_FILE
                        Coordinate file to use for H-bond analysis (REQUIRED).
                        [default: none]
  -p PRMTOP_FILE, --prmtop-file=PRMTOP_FILE
                        Amber parameter/topology file to use for H-bond
                        analysis.  If None, the name will be guessed by
                        replacing the extension of the coordinate file name
                        with ".prmtop"  [default: none]
  -D MDCRD_PRMTOP_DIR, --mdcrd-prmtop-dir=MDCRD_PRMTOP_DIR
                        The directory that contains the coordinate and prmtop
                        file.  If None, the names of the files will be used
                        without modification.  [default: none]
  -o OUTPUT_FILE_BASE, --output-file-base=OUTPUT_FILE_BASE
                        The first part of the name of the ptraj input files
                        that will be created.  The files will be named
                        OUTPUT_FILE_BASEnn.in, where nn is the two-digit
                        segment number.  Once ptraj is run (with
                        run_hbond_ptraj), the output files containing the
                        H-bond data will be named OUTPUT_FILE_BASEnn.out.
                        [default: segment]
  -B HBOND_DATA_DIR, --hbond-data-dir=HBOND_DATA_DIR
                        The directory where the ptraj input files will be
                        placed.  [default: .]
  -b BEGIN_FRAME, --begin-frame=BEGIN_FRAME
                        The number of the first frame to use in the H-bond
                        analysis.  [default: 1]
  -e END_FRAME, --end-frame=END_FRAME
                        The number of the last frame to use in the H-bond
                        analysis.  [default: 10000]
  -g SEGMENT_SIZE, --segment-size=SEGMENT_SIZE
                        The number of frames included in each segment of the
                        trajectory.  [default: 1000]
  -n NUM_RESI, --num-resi=NUM_RESI
                        The number of residues in the protein.  [default:
                        10000]
  -d DIST_CUTOFF, --dist-cutoff=DIST_CUTOFF
                        The distance cutoff that defines whether or not an
                        interaction is an H-bond.  [default: 3.0]
  -a ANGLE_CUTOFF, --angle-cutoff=ANGLE_CUTOFF
```

```
                        The angle cutoff that defines whether or not an
                        interaction is an H-bond.  [default: 120.0]
  -s, --no-self         Flag to turn off the inclusion of H-bonds between
                        atoms within the same residue.  [default: True]
  -S, --solvent         Flag to include solvent-protein interactions in the
                        analysis.  [default: False]
  -m MASK_FILE, --mask-file=MASK_FILE
                        File that contains extra lines to include in the ptraj
                        input files, primarily to include masks for ligands.
                        [default: mask]
```

The only required input to `setup_hbond_ptraj.py` is the coordinate file given with the `-x` (`-mdcrd-file`) option. The program will terminate with an error if this input is not given. In addition, the program will terminate with an error if the specified coordinate or prmtop file does not exist.

The program creates a number of files in the directory specified with the `-B` (`-hbond-data-dir`) option. The trajectory is broken into several segments, beginning with the frame specified with the `-b` (`-begin-frame`) option, ending with the frame specified with the `-e` (`-end-frame`) options, and with each segment containing a number of frames specified by the `-g` (`-segment-size`) option. The number of segments (and the number of ptraj input files created) is:

```
(END_FRAME - BEGIN_FRAME + 1) / SEGMENT_SIZE.
```

For statistical purposes, the segments must all have the same size, so if `SEGMENT_SIZE` initially does not divide the numerator equally, `END_FRAME` is decreased so that it does. A warning will be presented to the user to indicate that frames are being removed from consideration. The different ptraj input files that are created are identical except for the particular frames of the trajectory that they will be used to analyze. These files are named based on the `-o` (`-output-file-base`) option; the names are `OUTPUT_FILE_BASEnn.in`, where nn is the two-digit segment number.

An additional file called `run_hbond_ptraj` is also created. This is an executable file that should be run immediately after `setup_hbond_ptraj.py` is finished.

The program can handle all of the amino acid residues recognized by the Amber programs tLEaP and xLEaP (one of which, presumably, was used to prepare the system for MD). These include the twenty natural amino acids and the following additional residues: His in its delta-, epsilon-, and doubly-protonated forms (named HID, HIE, and HIP, respectively); neutralized Asp, Glu, and Lys (named ASH, GLH, and LYN); and Cys in its disulfide and deprotonated forms (named CYX and CYM). If the system of interest contains residues that are not among these amino acids, such as ligands, the additional hydrogen-bond donors and acceptors can be specified in a file named mask [or an alternate name specified by the `-m` (`-mask-file`) option]. The mask file will be read line for line into each ptraj input file, so it must contain the appropriate syntax for specifying the donors and acceptors to ptraj. In ptraj, hydrogen bond "donors" are defined as the heavy atoms that are not covalently bound to the hydrogen atom, and the "acceptors" are the heavy atoms that are covalently bound to the hydrogen. This is the opposite definition of the normal usage of the words, but it is the convention that ptraj has adopted. The file mask should contain one line for each potential H-bond donor and acceptor according to the syntax:

```
donor mask :lig@atom-name
acceptor mask :lig@heavy-atom-name :lig@H-atom-name
```

where `lig` is the ligand residue name or number and the atom-names are the names of the participating atoms.

The `-n` (`-num-resi`) option to specify the number of residues is of no consequence if there is no solvent present in the system of interest, but it is important if there is solvent. With solvent molecules present, failing to provide the correct number of residues will result in some of the water oxygen atoms being treated explicitly as hydrogen bond acceptors. This will not affect the analysis of protein-protein hydrogen bonds, but it will result in the compilation of more data than is necessary and may result in the memory issues during the ptraj execution. If the `-n` option is

not given, a warning will be printed. In some cases, the inclusion of H-bonds between protein and solvent may be desirable, but the `-S` (`-solvent`) flag should be used instead of increasing the number of residues. See the ptraj documentation for details on the way solvent donors and solvent acceptors are handled.

The `-d` (`-dist-cutoff`), `-a` (`-angle-cutoff`), `-s` (`-no-self`), and `-S` (`-solvent`) options can be used to control what interactions ptraj considers to be hydrogen bonds. The default distance and angle cutoffs are the same as those of ptraj, but in contrast to ptraj, hydrogen bonds between atoms of the same residue *will* be reported unless this behavior is turned off with the `-s` flag.

7.3 run_hbond_ptraj

This program is created by `setup_hbond_ptraj.py` to perform the ptraj executions. It takes no arguments. `run_hbond_ptraj` sequentially performs the ptraj execution for each segment. The resulting files output by ptraj have the name `OUTPUT_FILE_BASEnn.out`, again where `nn` is the two-digit segment number.

The output files contain H-bond data. For each H-bond, ptraj reports the occupancy percentage, average heavy atom-heavy atom distance, average heavy atom-H-heavy atom angle, average lifetime, and the maximum number of continuous frames the H-bond is populated. Standard deviations of the distance, angle, and lifetime are also included. In addition, a 10-character "graph" that displays the occupancy in each tenth of the trajectory is reported. Each character indicates a different level of occupancy: a space (0-5%), . (5-20%), - (20-40%), o (40-60%), x (60-80%), * (80-95%), or @ (95-100%).

7.4 combine_hbonds.py

Once the files with the H-bond data have been generated by ptraj, combine_hbonds.py is used to compile the data into a single file.

Documentaiton from the script:

```
Usage: combine_hbonds.py FILE1 [ FILE2 [ ... ] ] [options]
FILE1 and additional optional FILEs are files containing hbond data that
were produced by the ptraj hbond command.  At least one such file is
required.  The data in the files are spliced together to created a
unified data set.

Important notes:
* An AMBER prmtop or a PDB file may be input with the -p option.  The
  file will be used to determine the residue name associated with each
  residue number in the system.  If the file name does not end with
  '.pdb', the file will be assumed to be an AMBER prmtop file.  The
  offset and amino acid code are also used in the residue name
  generation.
* The resi_criteria option takes a comma-separated string containing any
  or all of the following:
        - individual residue numbers
        - a range of numbers, separated by a '-'
        - strings associated with valid residue lists in the
            standard file 'residue_lists'
* The atom_criteria option takes a comma-separated string containing
  the atom names to report.  In addition, the string can contain any
  of these strings:
        - 'bb_only': only H-bonds between two backbone atoms
        - 'not_bb': no H-bonds between two backbone atoms
        - 'protein_only': no H-bonds involving water
```

```
Options:
  -h, --help            show this help message and exit
  -o OUTPUT_FILE, --output-file=OUTPUT_FILE
                        The name of the output file.  If None, the results
                        will be written to stdout.  Supplying a name is
                        recommended.  [default: none]
  -B HBOND_DATA_DIR, --hbond-data-dir=HBOND_DATA_DIR
                        The directory that contains the H-bond data files.  If
                        None, the file names will be used without
                        modification, and output will be written to the
                        current directory.  [default: none]
  -g SEGMENT_SIZE, --segment-size=SEGMENT_SIZE
                        The number of frames included in each segment of the
                        trajectory.  [default: 1000]
  -p PRMTOP_FILE, --prmtop-file=PRMTOP_FILE
                        Amber parameter/topology file or PDB file for the
                        system.  [default: none]
  -D PRMTOP_DIR, --prmtop-dir=PRMTOP_DIR
                        The directory that contains the prmtop (or PDB) file.
                        If None, the name of the file will be used without
                        modification.  [default: none]
  -r RESI_OFFSET, --resi-offset=RESI_OFFSET
                        The offset between the residue numbers in the prmtop
                        (or PDB) file and the actual residue numbers.
                        [default: 0]
  -a AA_CODE, --aa-code=AA_CODE
                        Must be 1 or 3.  Indicates the use of 1- or 3-letter
                        amino acid codes in residue names.  [default: 3]
  -y, --occ-graph-only  Flag to report only the occupancy and graph data (no
                        distance or angle data).  [default: False]
  -R RESI_CRITERIA, --resi-criteria=RESI_CRITERIA
                        A comma- and dash-separated list of residue numbers to
                        include in the analysis.  [default: all]
  -A ATOM_CRITERIA, --atom-criteria=ATOM_CRITERIA
                        A comma-separated list of atom names to include in the
                        analysis.  [default: all]
  -O OCC_THRESH, --occ-thresh=OCC_THRESH
                        The minimum occupancy threshold that the H-bonds must
                        have to be reported.  [default: 0.0]
```

`FILE1` and additional optional `FILE` s are the files containing H-bond data that were produced by the ptraj hbond command. At least one such file is required. If the data files are not contained in the current directory, the `-B` (`-hbond-data-dir`) option can be used to indicate where they are located. The name of the output file can be specified by the `-o` (`-output-file`) option. Because a file output from `combine_hbonds.py` is required for the analysis aids subset_"hbonds.py" and `compare_hbonds.py` (below), it is recommended that a name be supplied. The output file will also be located in directory `HBOND_DATA_DIR`.

A line is output providing information for each H-bond in the system. As shown in the diagram below, this information includes the participating atoms, the occupancy percentage, the total number of frames included in the analysis, the average distance and angle of the H-bond (and their standard deviations), and the graph. Lifetime and maximum continuous occupancy data are not included, because splitting the trajectory into segments artificially shortens these values. Consequently, they are not reliable indicators of the data over the trajectory as a whole.

```
Thr376  OG1--HG1 ... OG   Ser295   35.20(  250) 2.854(0.17) 22.49(12.08) |*x@.    x@|@@.--.  . |
----- participating atoms ------   -occ- -num-  -dist -SD-  angle --SD-   ------- graph -------
                                    pct  frames
```

Several options control the display of the output. An AMBER prmtop or a PDB file may be input with the `-p` (`-prmtop-file`) option. (If the file name does not have an extension of `.pdb`, the program will assume it is a prmtop file.) The file will be used to determine the residue type of each residue number in the system. The residue type is prepended to the residue number to create the full residue name displayed in the output file. Whether to use a three- or one-letter amino acid code is determined by the input to the `-a` (`-aa-code`) option. Additionally, an offset can be supplied to change the numbering of the residues using the "r" (–resi-offset) option. This may be useful in systems with an unusual numbering convention. For example, the residues of the 1SGZ crystal structure are numbered from -3 to 385, but the Amber setup shifts them to 1-389. Using an offset of -4 brings the numbering back in line with convention. If a prmtop file for 1SGZ and this offset are given to the program, the residues in the output are labeled, for instance, "Tyr71" instead of "75." Also, the `-y` (`-occ-graph-only`) flag will affect the output presentation of the H-bond data. Using this flag will prevent the reporting of the distance and angle data, so that only the occupancy percentage and graph will be displayed.

The data in the input files can be filtered prior to output by specifying residue or atom criteria or an occupancy threshold. The residue criteria (`-R`, `-resi-criteria`) option takes a comma-separated string containing the residues numbers to report. It may contain any or all of the following: individual residue numbers, a range of numbers separated by a dash, or strings associated with valid residue lists in the standard file residue_lists as described below. The atom criteria (`-A`, `-atom-criteria`) option takes a comma-separated string containing the atom names to report. In addition, the program understands these strings: "`bb_only`", which will include only H-bonds between two backbone atoms; "`not_bb`", which will exclude H-bonds between two backbone atoms; or "`protein_only`", which will exclude any H-bonds involving water. In addition, both the residue and atom criteria can take the word "`all`". Lastly, the occupancy threshold (`-O`, `-occ-thresh`) option will exclude all H-bonds with an occupancy percentage below the threshold.

7.4.1 Note on the `residue_lists` file

The lines of the '`residue_lists file`' can be used to associate a list of residues with a particular name according to the following syntax:

```
name:  residue_string
```

where `name` is any string and `residue_string` is any comma- and dash-separated list of residues. For example, the line

```
loops: 9-12,65-67
```

will associate the string `loops` with the residue list 9, 10, 11, 12, 65, 66, and 67. A residue list is even allowed to include strings that were defined above it. For example, if the following lines are contained in the residue_lists file:

```
loops     : 9-12,65-67
more_loops: loops,100,104
```

the name `more_loops` is associated with residues 9, 10, 11, 12, 65, 66, 67, 100, and 104.

7.5 subset_hbonds.py

Once a file of combined H-bond data is created with `combine_hbonds.py`, `subset_hbonds.py` allows the user to select and sort a subset of the data, which greatly facilitates analysis.

Documentation from the script:

```
Usage: subset_hbonds.py FILE1 [options]
FILE1 is a file created by combine_hbonds.py that contains a dataset of
H-bonds.  Only a subset of all the data is presented according to the
criteria presented by the user.  The data will also be sorted according
to the metric specified by the user.

Important notes:
* The resi_criteria option takes a comma-separated string containing any
  or all of the following:
        - individual residue numbers
        - a range of numbers, separated by a '-'
        - strings associated with valid residue lists in the
            standard file 'residue_lists'
* The atom_criteria option takes a comma-separated string containing
  the atom names to report.  In addition, the string can contain any
  of these strings:
        - 'bb_only': only H-bonds between two backbone atoms
        - 'not_bb': no H-bonds between two backbone atoms
        - 'protein_only': no H-bonds involving water
* Note that ptraj uses a definition of H-bond donor and acceptor that is
  opposite of the normal convention.  This program follows the
  definitions of ptraj, in which the acceptor is the atom covalently
  bonded to the hydrogen atom.

Options:
  -h, --help            show this help message and exit
  -o OUTPUT_FILE, --output-file=OUTPUT_FILE
                        The name of the output file.  If None, the results
                        will be written to stdout.  [default: none]
  -B HBOND_DATA_DIR, --hbond-data-dir=HBOND_DATA_DIR
                        The directory that contains the H-bond data files.  If
                        None, the file names will be used without
                        modification, and output will be written to the
                        current directory.  [default: none]
  -s SORT, --sort=SORT  The quantity used to sort the results.  Must be one of
                        "occ_pct" (occupancy percentage), "donor", "acceptor",
                        "dist", or "angle".   [default: occ_pct]
  -c, --compress        Flag to compress the H-bond graph.  [default: False]
  -y, --occ-graph-only  Flag to report only the occupancy and graph data (no
                        distance or angle data).  [default: False]
  -R RESI_CRITERIA, --resi-criteria=RESI_CRITERIA
                        A comma- and dash-separated list of residue numbers to
                        include in the analysis.  [default: all]
  -A ATOM_CRITERIA, --atom-criteria=ATOM_CRITERIA
                        A comma-separated list of atom names to include in the
                        analysis.  [default: all]
  -O OCC_THRESH, --occ-thresh=OCC_THRESH
                        The minimum occupancy threshold that the H-bonds must
                        have to be reported.  [default: 0.0]
```

The only requirement for this program is a single argument `FILE1`, which is a file created by `combine_hbonds.py`. If more than one file is given to the program, it will use only the first one.

Most of the options are the same as those for `combine_hbonds.py`: `-o` (`-output-file`), `-B` (`-hbond-data-dir`), `-y` (`-occ-graph-only`), `-R` (`-resi-criteria`), `-A` (`-atom-criteria`), and `-O` (`-occ-thresh`). Two options differ: the sorting (`-s`, `-sort`) and graph compression (`-c`, `-compress`) options. The sorting option is self-explanatory, but the other requires some explanation. The graph compression flag will shorten the length of the graph by combining every pair of characters into a single character representing one fifth

(instead of one tenth) of the segment. It should be noted that the compressed graph is not as precise as the uncompressed graphs. Each character in the compressed graph represents the occupancy percentage over a pair of segments. However, the exact occupancy cannot always be determined solely from the ranges specified by the characters from each segment. For example, the actual occupancy of the pair of segments represented by "x*" could correspond to either "x" or "*", depending on the underlying percentages, which are unknown. In these cases, the character that is output is the one corresponding to the most probable occupancy percentage.

7.6 compare_hbonds.py

If H-bond data from multiple trajectories of the same system have been processed by `combine_hbonds.py`, the data from the different trajectories can be compared with `compare_hbonds.py`.

Documentation from the script:

```
Usage: compare_hbonds.py FILE1 [ FILE2 [ ... ] ] [options]
FILE1 and additional optional files are files created by
combine_hbonds.py that contain datasets of H-bonds from different
trajectories of the same system.  The particular subset of the H-bonds
and the metric for sorting can be specified by the user.

Important notes:
* Use the -i option to provide meaningful identifiers for the different
  trajectories.
* The resi_criteria option takes a comma-separated string containing any
  or all of the following:
        - individual residue numbers
        - a range of numbers, separated by a '-'
        - strings associated with valid residue lists in the
            standard file 'residue_lists'
* The atom_criteria option takes a comma-separated string containing
  the atom names to report.  In addition, the string can contain any
  of these strings:
        - 'bb_only': only H-bonds between two backbone atoms
        - 'not_bb': no H-bonds between two backbone atoms
        - 'protein_only': no H-bonds involving water
* If the H-bonds are sorted by occ_pct (the occupancy percentage), any
  H-bond that has an occupancy greater than the occ_thresh value will
  be retained.  If the H-bonds are sorted by occ_diff (the difference
  between the largest and smallest occupancy percentages for the
  systems), donor, or acceptor, only those H-bonds with occ_diff
  greater than the occ_thresh value will be retained.
* Note that ptraj uses a definition of H-bond donor and acceptor that is
  opposite of the normal convention.  This program follows the
  definitions of ptraj, in which the acceptor is the atom covalently
  bonded to the hydrogen atom.

Options:
  -h, --help            show this help message and exit
  -o OUTPUT_FILE, --output-file=OUTPUT_FILE
                        The name of the output file.  If None, the results
                        will be written to stdout.  [default: none]
  -B HBOND_DATA_DIR, --hbond-data-dir=HBOND_DATA_DIR
                        The directory that contains the H-bond data files.  If
                        None, the file names will be used without
                        modification, and output will be written to the
                        current directory.  [default: none]
```

```
-i IDENTIFIERS, --identifiers=IDENTIFIERS
                      Comma-separated list of identifying strings for the
                      trajectories to be compared.  If None, the
                      trajectories will simply be numbered.  [default: none]
-s SORT, --sort=SORT  The quantity used to sort the results.  Must be one of
                      "occ_diff" (occupancy difference), "occ_pct"
                      (occupancy percentage), "donor", or "acceptor".  The
                      occupancy difference is the difference between the
                      highest and lowest occupancy percentages for a
                      particular H-bond in the different trajectories.
                      [default: occ_diff]
-c, --compress        Flag to compress the H-bond graph.  [default: False]
-y, --occ-graph-only  Flag to report only the occupancy and graph data (no
                      distance or angle data).  [default: False]
-R RESI_CRITERIA, --resi-criteria=RESI_CRITERIA
                      A comma- and dash-separated list of residue numbers to
                      include in the analysis.  [default: all]
-A ATOM_CRITERIA, --atom-criteria=ATOM_CRITERIA
                      A comma-separated list of atom names to include in the
                      analysis.  [default: all]
-O OCC_THRESH, --occ-thresh=OCC_THRESH
                      The minimum occupancy threshold that the H-bonds must
                      have to be reported.  [default: 0.0]
```

This program will allow the user to compare side-by-side the H-bond data between two trajectories of the same system. At least one input file `FILE1`, containing the H-bond data from `combine_hbonds.py`, is required. In the output, the data for a given H-bond for all the trajectories are shown next to each other. The output has the same format as shown in the example in the `combine_hbond.py` description, except that an identifying string is placed at the beginning of each line to indicate the trajectory the data represents. These identifiers can be input with the `-i` (`-identifiers`) option.

```
1w50A:  Glu165      N--H ... OD1  Asn162   36.00(  250) | ..xo.xxo-|-oo*o.....|
1sgz :  Glu165      N--H ... OD1  Asn162   86.80(  250) |*o@@**x**@|*x@*x****@|
```

Most of the options are the same as those described for `combine_hbonds.py` and `subset_hbonds.py`. The only difference is in the interaction of the sorting (`-s`, `-sort`) and occupancy threshold (`-O`, `-occ-thresh`) options. If the H-bonds are sorted by occupancy difference, donor, or acceptor, the output will include only those H-bonds for which the occupancy *difference* is greater than the threshold. If they are sorted by occupancy percentage, the output will include only those H-bonds that have at least one trajectory with an occupancy *percentage* greater than the threshold.

CHAPTER

EIGHT

Correlated Dynamics

8.1 Executive Summary

Here's how to make plots for a simulation of 1rx1 with 1000ps windows:

8.1.1 Make the directories

```
ssh node6
cd /data/people/mlerner
mkdir correlated_dynamics
cd correlated_dynamics
mkdir ptraj_files
mkdir ptraj_files/images
mkdir ptraj_files/1rx1
```

8.1.2 Calculating the correlation matrices

write_ptraj_input_files.py –strip-water –strip-hydros –input-dir=. –mdcrd=1rx1.trj –structure-name=1rx1 –start=100 –stop=1000 –window-size=200 –window-spacing=100

```
#
# Standard defaults are windowsize of 1000ps (1NS) and windowspacing of 100ps.
#
write_ptraj_input_files.py --strip-water --strip-hydros --input-dir=. --mdcrd=1rx1.mdcrd
   --structure-name=1rx1 --start=500 --stop=5500 --ps=5 --window-size=1000 --window-spacing=500

nohup run_ptraj.py --input-dir=. --prmtop=1rx1.prmtop --structure=1rx1&

cd ptraj_files/1rx1
bzip2 *.dat #*
cd ../..
```

8.1.3 Extract the per-residue and per-atom information

```
nohup do_correlated_md_analysis.py --output-dir=ptraj_files --structure-name=1rx1 --start=500
   --stop=5500  --window-size=1000 --non-ca-resis=160,161-175&
```

8.1.4 Make the plots

```
nohup make_correlated_dynamics_plots.py --structure-name=1rx1 --start=500 --stop=5500
   --window-size=1000 --window-spacing=500&

make_movies.py --structure-name=1rx1 --plot-types=ca,avg,max,min,abs,straight,mainheavy

cd ptraj_files
tar cvf 1rx1_100ps_movies.tar 1rx1BigAnimatedMovies.html images/animated_*
mv 1rx1_100ps_movies.tar ~
cd ..
```

8.2 More detailed explanations

The scripts have many options, and we'll describe a standard setup here. You'll need a couple of things before you begin:

1. An MD trajectory from *sander*. In this example, it will be called *1rx1.mdcrd*.
2. The paramater/topology file corresponding to that trajectory. Ours will be called *1rx1.prmtop*
3. Sufficient disk space and processor power. This can easily eat up several gigs of disk space, and we usually run things with 2G of memory.

8.2.1 Making the directories

First of all, we need to set up a directory structure to store our files. We need a main directory which will contain all of our results (*correlated_dynamics*). Inside that, we need to store the images and the data. We'll store images and data in *ptraj_files*. Images will go in *ptraj_files/images*. We'll have different directories for each different structure that we study. These examples will be for the DHFR structure 1RX1, and we'll store data in *ptraj_files/1rx1*. Assuming we do all of this on node6 of our cluster, here's how to set up the directories:

```
ssh node6
cd /data/people/mlerner
mkdir correlated_dynamics
cd correlated_dynamics
mkdir ptraj_files
mkdir ptraj_files/images
mkdir ptraj_files/1rx1
```

8.2.2 Calculating the correlation matrices

We use *ptraj* to calculate the correlation matrices. So, we need to write out lots of ptraj input files and then we need to run them. This sets up all of the calculations that we'll use, so it's important to get the options right.

- *structure-name* should be the same thing you specified earlier. *1rx1* in our case.
- *input-dir* is the path to the directory that contains the trajectory and parameter/topology file.
- *strip-water* and *strip-hydros* tell us whether or not to include waters and hydrogens in our calculations. Waters are almost never worth including. Hydrogens can be interesting, but it's very easy to run out of memory on reasonably-sized systems, so we typically exclude them.

- *ps* tells us how often (in picoseconds) frames were written to the mdcrd file.
- *start* and *stop* tell us (in ps) when to start and stop the windows. *window-size* tells us how long each window should be (in ps). *window-spacing* tells us how often to write out windows (in ps). The defaults are to write out windows of length 1ns (1000ps) every 100ps.
- There are several other options, including the ability to insert user-specified commands directly to the ptraj input files. Please use the *help* option for more information.

Finally, these files can take up a lot of disk space. We typically compress things with bzip2. The scripts are smart enough to decompress things on the fly later on.

so, here's how we set up and run ptraj:

```
#
# Standard defaults are windowsize of 1000ps (1NS) and windowspacing of 100ps.
#
write_ptraj_input_files.py --strip-water --strip-hydros --input-dir=. --mdcrd=1rx1.mdcrd
   --structure-name=1rx1 --start=500 --stop=5500 --ps=5 --window-size=1000 --window-spacing=100

nohup run_ptraj.py --input-dir=. --prmtop=1rx1.prmtop --structure=1rx1&

cd ptraj_files/1rx1
bzip2 *.dat
cd ../..
```

8.2.3 Extract the per-residue and per-atom information

Ptraj calculates correlations between each atom. We also want to calculate the following quantities:

- Correlations between alpha-carbons.
- Correlations between main-chain heavy atoms
- The following quantities on a per-residue basis:
  - average
  - maximum
  - minimum
  - largest absolute value

If hydrogens are included, we will also calculate correlations between potential hydrogen-bond donors and acceptors.

Since we are calculating alpha-carbon correlations, it is important to provide a list of residues that do not contain alpha carbons (*non-ca-resis*). In this case, 160 is our cofactor and 161-175 are our counter-ions. We could have excluded these during the *write_ptraj_input_files.py* command, but chose not to.

All of our files follow a standard naming convention, so telling each successive command the start, stop, spacing and size of the windows is enough to make sure that the correct files are read in.

```
nohup do_correlated_md_analysis.py --input-dir=. --structure-name=1rx1 --start=500 --stop=5500
   --window-size=1000 --non-ca-resis=160,161-175&
```

8.2.4 Make the plots

Now we have calculated everything and it's time to make the plots. *make_correlated_dynamics_plots.py* makes the individual plots and has *many* different options (again, use *help* to list them all). Here is a standard run.

- we specify the structure and the details about the windows as before.
- *plot-types* is a comma-separated list of plot types. The standard ones that we use are *ca,avg,max,min,abs,straight,mainheavy* and the command-line help will detail other options for you. Since we don't specify this on the command-line below, it will default to the standard options.

```
nohup make_correlated_dynamics_plots.py --input-dir=. --structure-name=1rx1 --start=500
   --stop=5500 --window-size=1000&
```

That produces plots of each of the individual windows. It's worth examining these on their own. However, it's often a lot more interesting to look at movies of these all pasted together. The *convert* program is used to do this. If it's not installed, it's easy to install on OS X, Linux and Windows. *make_movies.py* calls *convert* appropriately:

```
make_movies.py --structure-name=1rx1 --plot-types=ca,avg,max,min,abs,straight,mainheavy
```

Finally, we may wish to move the movies to another machine for viewing. The movies are in the *images* subdirectory. *make_movies.py* also generates an html file that shows all of the movies with thumbnails. Here's how to collect the movies and html file:

```
cd ptraj_files
tar cvf 1rx1_100ps_movies.tar 1rx1BigAnimatedMovies.html images/animated_*
mv 1rx1_100ps_movies.tar ~
cd ..
```

8.2.5 Other notes

Specific options

make_correlated_dynamics_plots.py has several useful options.

- Residues of interest can be marked with *mark-resis*. This is useful in several cases, including:
  - marking loops, helices or other regions of interest
  - marking every 10th residue to show a grid (especially useful for orientation in the main-chain heavy plots)
  - you can select different color maps with the *cmap* option.

Please read through the command-line help for a detailed, up-to-date explanation.

File formats and how to save space

The bzip2'd files are very small. However, especially when dealing with main-chain heavy atoms, they can be quite slow. *numpy* has an internal format that is much faster to read. You can use *convert_to_numpy_format.py* to covert your output to this format. It looses a small amount of precision, but that seems to be completely negligible. All of the correlated-dynamics scripts will deal transparently with any combination of .dat, .numpy and .bz2 files. You specify things as above and it'll figure out how to read .dat or .numpy or .numpy.bz2 without any trouble.

Smaller movies

The movie files are created as animated GIFs. This has the advantage that they can be played anywhere. However, they can be quite large. Sophisticated tools such as `mencoder` (http://www.mplayerhq.hu) can use two-pass encoding to convert them into much smaller AVI files. We find that the XVid codec produces good quality movies. Codecs and encoders are frequently updated, and it is suggested that users consult websites like the one mentioned above for the most current information.

8.3 Documentation from the scripts

8.3.1 write_ptraj_input_files.py

```
Usage: write_ptraj_input_files.py [options]

Please make sure that you have created the following directories:

  output-dir
  output-dir/structure-name
  output-dir/images

This script will emit the PDB file that you will use as a
reference structure later on.  It will live in <outputdir>/<structure>_ref.pdb.1


Options:
  -h, --help            show this help message and exit
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ. [default:
                        1rx1]
  --output-dir=OUTPUTDIR
                        Directory where we will put our results.  This should
                        be the same as the directory where we put our ptraj
                        files before, and it should contain the .dat files
                        that ptraj outputs.  [default: ptraj_files/]  The
                        images will go to <outputdir>/images and the html file
                        will be in <outputdir>
  --start=START         Time, in ps, to start the windows.  [default: 500]
  --stop=STOP           Time, in ps, to stop the windows.  [default: 10500]
  --window-size=WINDOWSIZE
                        Length, in ps, of window size.  [default: 1000]
  --window-spacing=WINDOWSPACING
                        Spacing between windows, in ps.  [default: 100]
  --input-dir=INPUTDIR  Directory that contains our input files.  It should
                        contain the prmtop file and the mdcrd file.  [default:
                        ./]
  --mdcrd=MDCRD         Comma-separated list of mdcrd files
  --ps=PS               Number of ps per frame.  [default: 5]
  --align=ALIGN         How to align. 'all' means 'rms first *'.  'none' means
                        no alignment.  Any other string will be treated as the
                        alignment string.  For example, if you say ':1-428@CA'
                        the ptraj file will say 'rms first :1-428@CA'.
                        [default: all]
  --strip-hydros        Strip the hydrogens out during the ptraj runs.
  --strip-waters        Strip the waters out during the ptraj runs.  This
                        assumes they're named WAT.
```

```
  --write-covar         Write out the covariance matrix [default: False]
  --other-ptraj-strips=OTHER_PTRAJ_STRIPS
                        Comma-separated list of other things that ptraj should
                        strip.  For example, could be :WAT,:BOB and we would
                        add two lines, one saying strip :WAT and one saying
                        strip :BOB.
```

8.3.2 run_ptraj.py

```
Usage: run_ptraj.py [options]

Please make sure that you have created the following directories:

  output-dir
  output-dir/structure-name
  output-dir/images


Options:
  -h, --help            show this help message and exit
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ. [default:
                        1rx1]
  --output-dir=OUTPUTDIR
                        Directory where we will put our results.  This should
                        be the same as the directory where we put our ptraj
                        files before, and it should contain the .dat files
                        that ptraj outputs.  [default: ptraj_files/]  The
                        images will go to <outputdir>/images and the html file
                        will be in <outputdir>
  -i INPUTDIR, --input-dir=INPUTDIR
                        Directory that contains our input files.  It should
                        contain the prmtop file and the mdcrd file.  [default:
                        ./]
  -p PRMTOP, --prmtop=PRMTOP
                        Name of prmtop file
```

8.3.3 do_correlated_md_analysis.py

```
Usage: do_correlated_md_analysis.py [options]

Please make sure that you have created the following directories:

  output-dir
  output-dir/structure-name
  output-dir/images


Options:
  -h, --help            show this help message and exit
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ. [default:
                        1rx1]
```

```
  --output-dir=OUTPUTDIR
                        Directory where we will put our results.  This should
                        be the same as the directory where we put our ptraj
                        files before, and it should contain the .dat files
                        that ptraj outputs.  [default: ptraj_files/]  The
                        images will go to <outputdir>/images and the html file
                        will be in <outputdir>
  --start=START         Time, in ps, to start the windows.  [default: 500]
  --stop=STOP           Time, in ps, to stop the windows.  [default: 10500]
  --window-size=WINDOWSIZE
                        Length, in ps, of window size.  [default: 1000]
  --window-spacing=WINDOWSPACING
                        Spacing between windows, in ps.  [default: 100]
  --non-ca-resis=NON_CA_RESIS
                        Comma separated list of residues that don't contain
                        alpha carbons.  We need this to make some of our
                        output images.  [default: []], but you could say
                        160,161 for example
```

8.3.4 make_correlated_dynamics_plots.py

```
Usage: make_correlated_dynamics_plots.py [options]

Please make sure that you have created the following directories:

  output-dir
  output-dir/structure-name
  output-dir/images

This will spit out an html file that will show you your images.  If you
need to look at the images on another machine, tar up the html file and
output-dir/images together and move that to the other machine.


Options:
  -h, --help            show this help message and exit
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ. [default:
                        1rx1]
  --output-dir=OUTPUTDIR
                        Directory where we will put our results.  This should
                        be the same as the directory where we put our ptraj
                        files before, and it should contain the .dat files
                        that ptraj outputs.  [default: ptraj_files/]  The
                        images will go to <outputdir>/images and the html file
                        will be in <outputdir>
  --start=START         Time, in ps, to start the windows.  [default: 500]
  --stop=STOP           Time, in ps, to stop the windows.  [default: 10500]
  --window-size=WINDOWSIZE
                        Length, in ps, of window size.  [default: 1000]
  --window-spacing=WINDOWSPACING
                        Spacing between windows, in ps.  [default: 100]
  --cmap=CMAP           Color map to use when making the plots. Our custom
                        cmaps are Normal and Scaled.  Standard matplotlib
                        cmaps ['Spectral', 'summer', 'RdBu', 'gist_earth',
                        'Set1', 'Set2', 'Set3', 'Dark2', 'hot', 'RdPu',
```

```
                        'YlGnBu', 'RdYlBu', 'gist_stern', 'cool', 'gray',
                        'GnBu', 'gist_ncar', 'gist_rainbow', 'bone', 'RdYlGn',
                        'spring', 'Accent', 'PuBu', 'spectral', 'gist_yarg',
                        'BuGn', 'YlOrRd', 'Greens', 'PRGn', 'gist_heat',
                        'Paired', 'hsv', 'Pastel2', 'Pastel1', 'copper',
                        'OrRd', 'jet', 'BuPu', 'Oranges', 'PiYG', 'YlGn',
                        'gist_gray', 'flag', 'BrBG', 'Reds', 'RdGy', 'PuRd',
                        'Blues', 'Greys', 'autumn', 'pink', 'binary',
                        'winter', 'prism', 'YlOrBr', 'Purples', 'PuOr',
                        'PuBuGn'] are also supported.[default: Normal]
  --plot-types=PLOTTYPES
                        Comma-separated list of plot types.  [default: ['ca',
                        'avg', 'max', 'min', 'abs', 'straight', 'mainheavy',
                        'allheavy', 'sidechainhbond', 'hbond']]
  --mark-resis=MARKRESIS
                        A list of residues to mark on the plots.  [default:
                        []]
  --highlight=HIGHLIGHT
                        How strongly to highlight the marked residues.  Note
                        that --highlight-mode tells us how exactly we will do
                        the highlighting.  0.1 and 0.2 are decent values if
                        you want to use this feature for most plots, although
                        you'll need something stronger for the absolute value
                        plots. [default: 0.2]
  --highlight-mode=HIGHLIGHTMODE
                        When highlight-mode is 'negative' we put a white block
                        down on top of the marked residues, the opacity of
                        which is controled by --highlight.  When it's
                        'positive', we put that white block down on squares of
                        residues that are *not* highlighted instead.  When
                        it's 'supernegative', we do just like 'negative'
                        except that the block will be twice as opaque where
                        the highlighted rows and columns intersect.  Positive
                        and supernegative seem to be more useful than
                        negative.  [default: positive]
  --skip-resis=SKIPRESIS
                        A list of residues that will be skipped in the plots.
                        [default: []]
  --no-ticks            do not include tick marks on the axes
  --dpi=DPI             dpi for figures [default: 200]
  --title=TITLE         If no title is specified, one will be automatically
                        generated. Note that the title is part of the filename
                        that we save. [default: none]
```

8.3.5 make_movies.py

```
Usage: make_movies.py [options]

Please make sure that you have created the following directories:

  output-dir
  output-dir/structure-name
  output-dir/images


Options:
```

```
  -h, --help            show this help message and exit
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ. [default:
                        1rx1]
  --output-dir=OUTPUTDIR
                        Directory where we will put our results.  This should
                        be the same as the directory where we put our ptraj
                        files before, and it should contain the .dat files
                        that ptraj outputs.  [default: ptraj_files/]  The
                        images will go to <outputdir>/images and the html file
                        will be in <outputdir>
  --plot-types=PLOTTYPES
                        Comma-separated list of plot types.  [default: ['ca',
                        'avg', 'max', 'min', 'abs', 'straight', 'mainheavy',
                        'allheavy', 'sidechainhbond', 'hbond']]
  --no-slow-movies      Set this if you do not want to generate the movies
                        that have 0.5s spacing between the frames.
  --movie-link=MOVIELINK
                        'fast' if you want the thumbnails to link to the fast
                        images, anything else for the slow ones. [default:
                        fast]
```

8.3.6 convert_to_numpy_format.py

```
Usage: This will convert the .dat or .dat.bz2 files to numpy versions.
    It will not automatically delete the .dat(.bz2) files.  If your input
    files are bz2, your output files will be too.

    If you already have a corresponding .numpy or .numpy.bz2 file, we won't
    write out a new file.


Options:
  -h, --help            show this help message and exit
  --dir=DIR             Directory in which the files reside. [default: .]
  --structure-name=STRUCTURENAME
                        Name of your structure.  E.g. 1RX1 or 1SGZ
  --compression=COMPRESSION
                        Type of compression currently used on files.  Leave
                        blank for uncompressed, .bz2 if they're .bz2 files.
                        Note that it's '.bz2' not 'bz2'. [default: ]
  --all-dat-files       By default, we will only convert the
                        all_atom_correlmat files.  If you use this option, we
                        will convert all dat files.  Don't forget, though,
                        that you'll still have to call this command twice if
                        you have some files that are bzipped and some that are
                        not.
```