TrajMap.py

Molecular Dynamics simulation trajectory visualization

Release: 1.0.0; July 16, 2022

Author: Matej Kožić

Email: mkozic@chem.pmf.hr

Repository: github.com/matkozic/TrajMap

0. Contents

1. Introduction

TrajMap is a Python based module for visualizing molecular dynamics (MD) simulation trajectories in the form of a heatmap. Goal of the module is to provide a tool that conclusively show the whole course of a simulation and the differences between simulations, in an elegant and easy to read format. *When you see a trajectory map, you know the full course of your simulation*.

Main concept behind trajectory maps are movements of residues from their starting position, referred to as shifts. Shifts, defined as a Euclid distances of centers of masses of residues backbone in time t from reference time t_0, are plotted in a matrix for every residue and every frame of the simulation. Using center of mass results in a better resolution of the map because in-residue rotations and vibrations are diminished in magnitude. Expression used for calculating shifts is shown with equation 1.

$$s(r,t,t_{ref}) = \sqrt{\left(x_{r,t} - x_{r,t_{ref}}\right)^2 + \left(y_{r,t} - y_{r,t_{ref}}\right)^2 + \left(z_{r,t} - z_{r,t_{ref}}\right)^2} \qquad (1)$$

A shift $\boldsymbol{s}$ is calculated for every residue $\boldsymbol{r}$ in time $\boldsymbol{t}$ against reference time $\boldsymbol{t_{ref}}$ which is taken as the first frame of the simulation. Coordinates $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are of the center of mass of a residues backbone atoms: CA, C, O, and N. Shifts are then plotted in a heatmap.

TrajMap shows the location, time, and magnitude of every shift of proteins backbone during the simulation, as well as the full course of a simulation in the form of a 2D heatmap. It provides a map of every conformational change in the protein, and offers a way to distinguish between conformational changes that lead to a stable different conformation of a region and changes that result in an increase of fluctuations; which is a distinction otherwise not always clear from other common analyses such as RMSF.

Distinguishing start times of individual conformational changes in forms of bands on a heatmap can aid in interpreting other analyses such as RMSD, R_{gyr}, and others. By comparing an RMSD graph with a TrajMap it is possible to assign RMSD peaks with conformational changes that caused them.

A useful feature of TrajMap is its ability to showcase the order in which conformational changes occur. By examining the starting times of bands responding to shifts of regions involved in a mechanism it's easy to deduce the order in which regions change; which can aid in solving a mechanism of that conformational change. Another powerful feature is a function to plot the average shift of a defined region throughout time, in the form of a 2D graph. That can be used to show the position of e.g., a helix throughout the simulation, as a proof of a conformational change / proposed mechanism.

Heatmaps are color-coded matrices, so it's possible to perform calculations with them. Several heatmaps can be averaged out to show the average course of e.g., a triplicate simulation. Furthermore, by subtracting two matrices (e.g., *apo – holo*, *mut – wild…*) all the differences of their courses and conformational changes are concisely and conclusively shown. The resulting difference TrajMap is made with a divergent colormap to highlight the differences. For an example of *apo – holo* if the shifts of a region are stronger in *holo* type that region of a difference map will be negative, while regions with shifts stronger in *apo* type will be positive. Regions with similar shifts will cancel out and be close to zero. That is an elegant way to conclusively demonstrate all the differences of conformational changes between simulations.

Averaged out matrices can be subtracted to account for different variables between them. For example, a system of four variants of an enzyme *HRP* with two variables: glycosylated (*g*), deglycosylated (*d*), wild (*w*), and recombinant (*r*) type. Two variables result in four simulations: *gwHRP*, *grHRP*, *dwHRP*, and *drHRP*. To showcase the effects of deglycosylation in conjunction with mutations the deglycosylated wild and recombinant types would be subtracted of glycosylated wild and recombinant types: *grHRP + gwHRP – drHRP – dwHRP*. This way the effects of deglycosylation are shown with the effects of mutations on them accounted for.

Another example would be wild and recombinant types at five increasing temperatures. By averaging out the recombinant type across its five temperatures and subtracting it from the temperature-average of the wild type, it's possible to show the differences of wild against recombinant type throughout increasing temperatures; which can be interpreted as the effect of mutations in the context of heating. Alongside shown examples various other calculations are possible.

The last function of TrajMap module calculates the distance of each individual residue to the others and shows the result as a square

matrix. By comparing multiple protein-heatmaps it’s possible to compare respective protein structures regardless of their coordinates, without the need for a reference point.

2. Installation & dependencies

~~There are two main ways to use TrajMap module, the first being installing it directly with pip as any other Python module:~~

```
pip install TrajMap
```

The second way is by having the modules .py file in a directory and importing it locally. Local import is recommended for full control as it enables additional options such as custom axes names (e.g., “ t / 15 ns” instead of default “Frames” for *x* axis), full control over titles, and additional advanced functionality such as modifying colormaps, render sizes, additional terminus support, and more.

Dependencies: Pandas, Numpy, Matplotlib

3. Manual

3.1. Quick guide

```
pdb2csv(file, savename, residues)

csv2matrix(file, residues)

# params = [x_major, x_minor, y_major, y_minor,
#                    vmin, vmax, residues, cmap, aspect]
# cmap: 0 - linear, 1 - divergent
# aspect: 0 - auto, 1 - square
matrix2map(matrix, savefig, title, params)

# params = [res1, res2, time1, time2]
matrix2avg(matrix, params)

# y_params = [y_min, y_max, y_step, y_tick]
# x_params = [x_min, x_max, x_step, x_tick]
# avg_params = [rolling_average, color]
avg2graph(avg, savefig, title, label, y_params, x_params, avg_params)


# params = [xy_major, xy_minor, vmin, vmax, residues]
# save = ["y"/"n", savename]
# load = ["y"/"n", loadname]
protmap(file, savefig, title, save, load, params)
```

3.2. Functions

There are a total of six functions. General workflow involves converting files into a matrix from which a map is made. The first step in creating a map is converting a preprocessed .pdb file to a .csv file. The second step is converting a .csv file to a matrix that is loaded as a variable, and then inputted in the third step creating a heatmap. Preprocessing consists of creating a trajectory file in the form of a .pdb file of aligned protein backbone. Preprocessing can be easily done in VMD and is explained thoroughly in chapter 4.1.

The first step is the most time consuming, taking approximately 20 minutes for a 300 residue protein with 500 frames. More frames mean better resolution, but time increases exponentially. Ideal number of

frames is 500 to 1000, so exporting stride should be adjusted accordingly. Other functions take significantly less time.

3.2.1. pdb2csv

```
pdb2csv(file, savename, residues)
```

This function converts a .pdb file to a .csv file and saves it under "*savename*". Input file is a preprocessed .pdb trajectory of an aligned single-peptide protein (see chapter 4.1.). Variable "*residues*" is an integer with the number of residues in a protein. As this is the most time consuming step saving the .csv file saves a lot of time as it can be used in all other functions, and this function has to be run only once for a protein.

3.2.2. csv2matrix

```
csv2matrix(file, residues)
```

This function converts a .csv file made with *pdb2csv* function to a matrix, and outputs it to the variable it's assigned to. Residues are an integer of residues again, and in all future cases.

3.2.3. matrix2map

```
# params = [x_major, x_minor, y_major, y_minor,
#                    vmin, vmax, residues, cmap, aspect]
# cmap: 0 - linear, 1 - divergent
# aspect: 0 - auto, 1 - square
matrix2map(matrix, savefig, title, params)
```

From the inputted matrix made with previous function a heatmap is made, saved to "*savefig*", with a title "*title*", and parameters responding to a list that is defined as shown. *x/y_major/minor* respond to locations of major and minor ticks on *x* and *y* axis respectively, and are integers or floats. Minimal and maximal values of the colorbar are defined with *vmin* and *vmax*. Variable "*cmap*" is either 0 for a linear colormap when showing a single simulation, or 1 for a divergent colormap when showing a difference of simulations. "*aspect*" defines the shape of a pixel, with 0 being "*auto*" resulting in pixels fitted automatically and 1 resulting in square pixels. *Aspect* = 0 is recommended for most uses, but certain situations benefit from square

pixels. It goes without saying how the order of variables in a list is crucial. Once a matrix is loaded it is recommended to play around with the *"params"* list variables to fine-tune the final figure.

3.2.4. matrix2avg

```
# params = [res1, res2, time1, time2]
matrix2avg(matrix, params)
```

This function calculates the average of residues *res1* to *res2* in a time *time1* to *time2* of an inputted matrix, and outputs it to a variable it's assigned to.

3.2.5. avg2graph

```
# y_params = [y_min, y_max, y_step, y_tick]
# x_params = [x_min, x_max, x_step, x_tick]
# avg_params = [rolling_average, color]
avg2graph(avg, savefig, title, label, y_params, x_params, avg_params)
```

From a matrix of averages, made with previous function, a graph is made with parameters as defined. *"label"* is the label of the curve on a graph, *"rolling_average"* is an integer of number of points used to calculate a rolling average. Color is a string. All available colors can be found in Matplotlib documentation

3.2.6. protmap

```
# params = [xy_major, xy_minor, vmin, vmax, residues]
# save = ["y"/"n", savename]
# load = ["y"/"n", loadname]
protmap(file, savefig, title, save, load, params)
```

Protein map is made from a single-frame .pdb file that doesn't have to be preprocessed in any way. "save" and "load" are lists as shown, with first entry being "y" for saving/loading and anything else for not saving/loading.

4. Usage & examples

4.1. VMD preprocessing

Input file for the module is an **aligned** trajectory of a single peptide chain backbone in .pdb format. Such can easily be created in VMD:

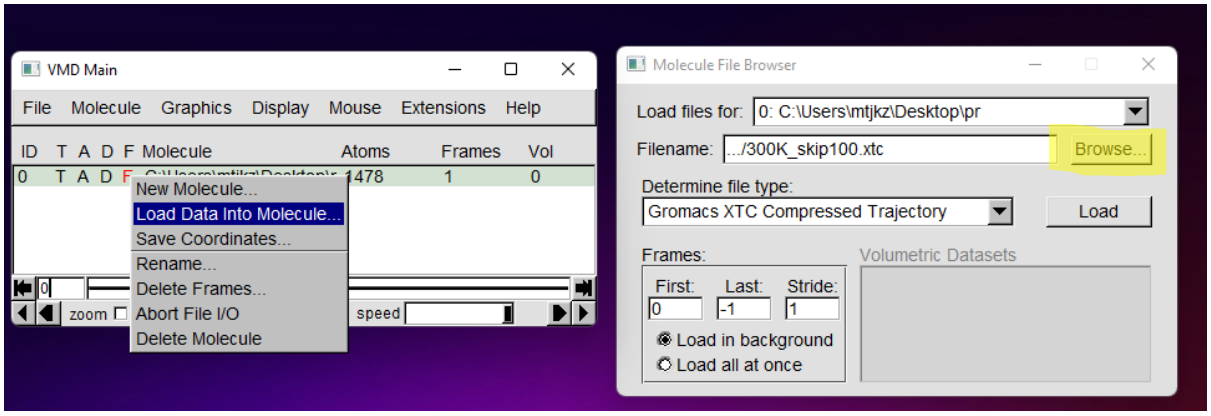


Fig 1: A molecule is opened in VMD. The trajectory file is loaded by left-clicking the molecule, therefore selecting it, then right clicking and following the prompt.

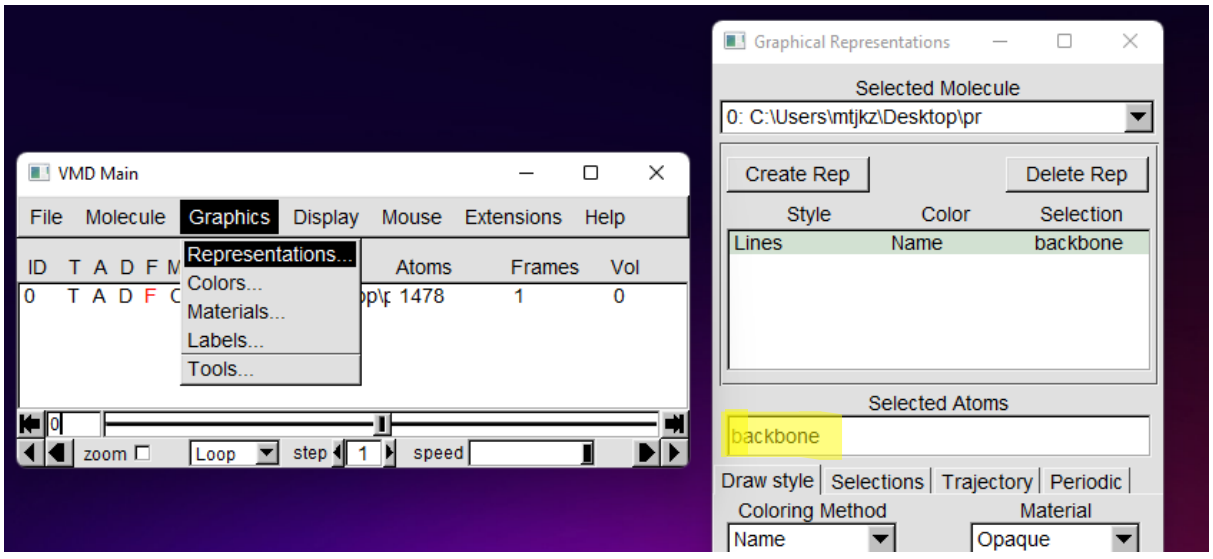


Fig 2: In representations tab “backbone” is selected.

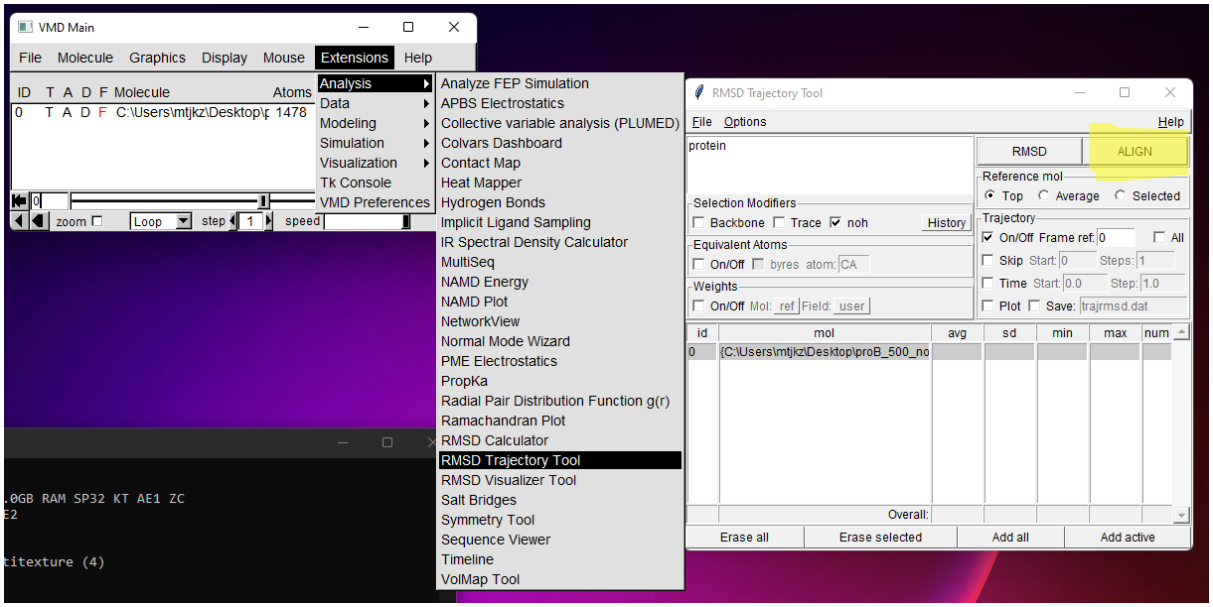


Fig 3: Under Extensions/Analysis/ RMSD Trajectory Tool the molecule is aligned throughout the frames of the trajectories.

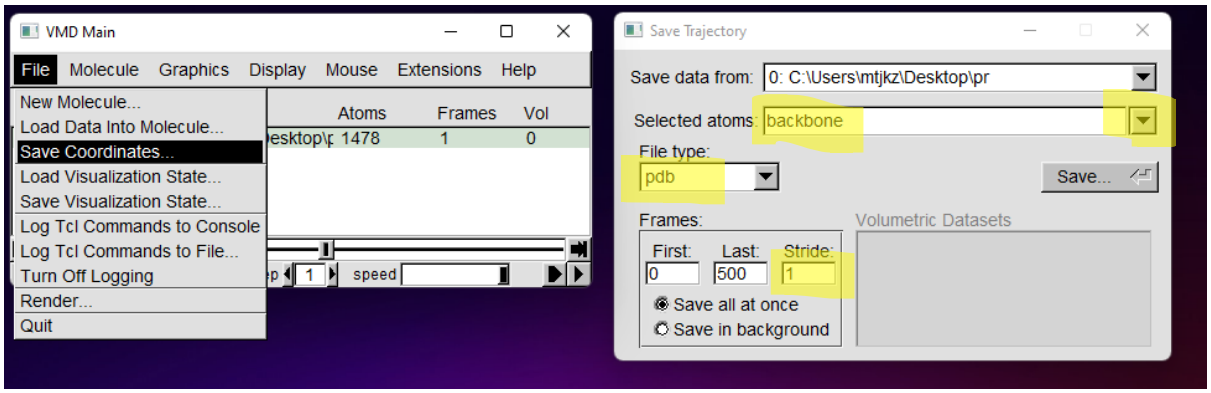
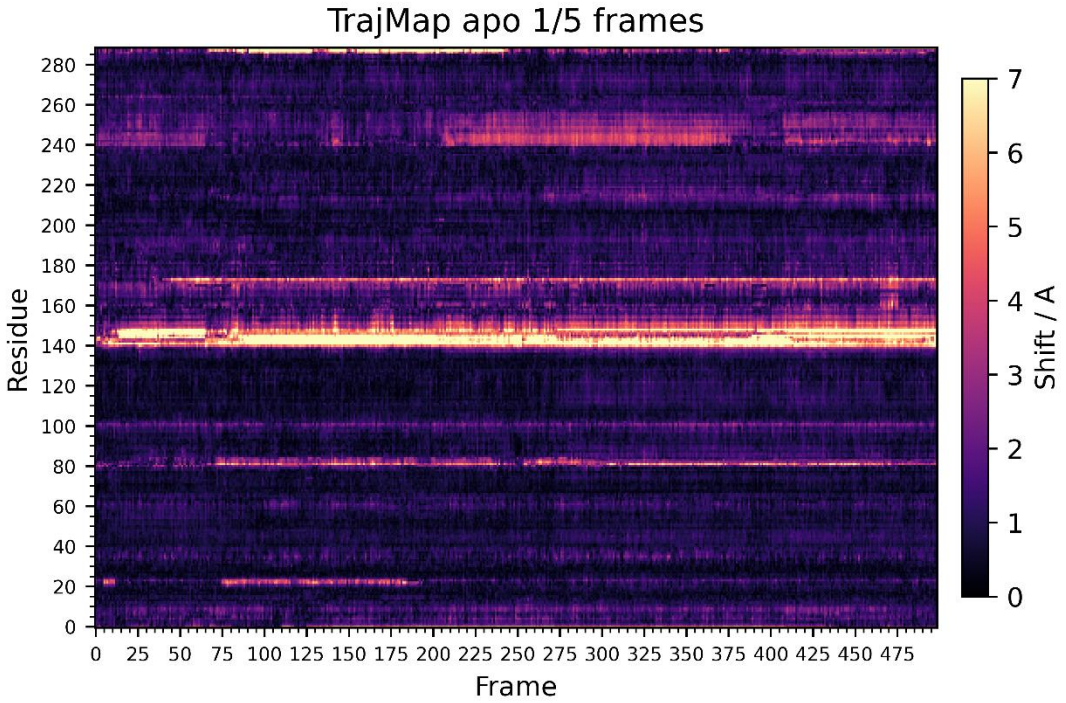


Fig 4: The selected molecule is exported with “Save Coordinates” option. Selected atoms are “backbone”, file type is “pdb”, and with stride option the number of exported frames can be adjusted to each frame, each 10th frame, or else. The recommended number of frames is 500 to 1000.

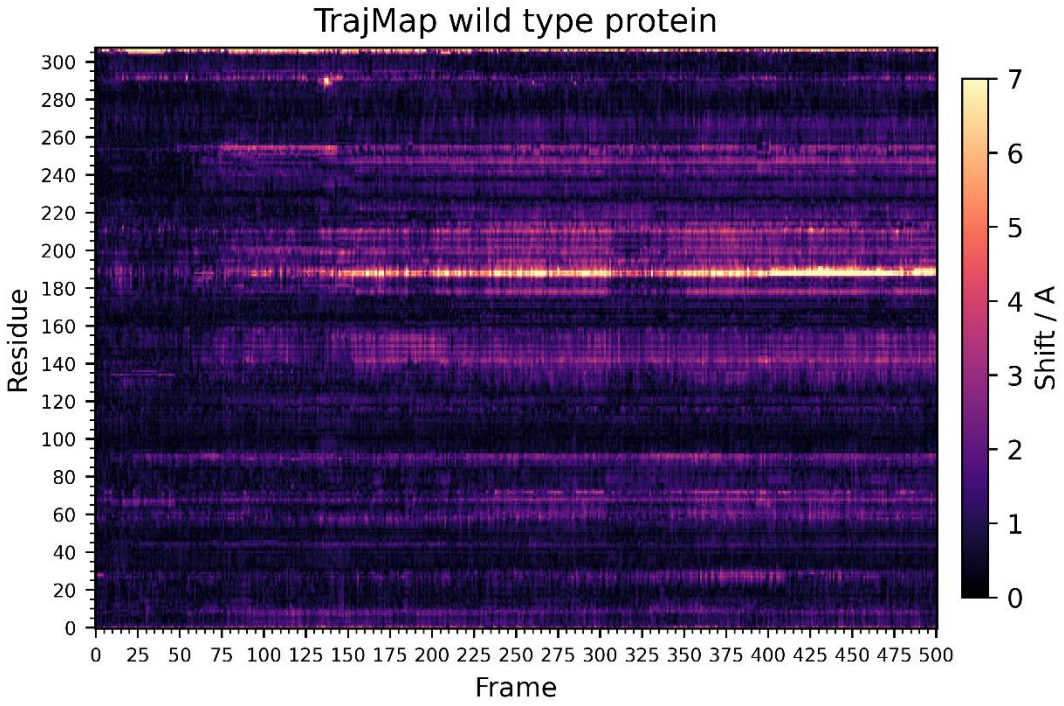
4.2. Single map from a .pdb file

```
import TrajMap_local as tm

file = "data/apo_stride_5.pdb"
save = "data/apo_stride_5.csv"
residues = 289
tm.pdb2csv(file, save, residues)

file = "data/apo_stride_5.csv"
residues = 289
matrix = tm.csv2matrix(file, residues)

savefig = "figs/apo.png"
title = "apo 1/5 frames"
params = [25, 5, 20, 5, 0, 7, 289, 0, 0]
tm.matrix2map(matrix, savefig, title, params)
```



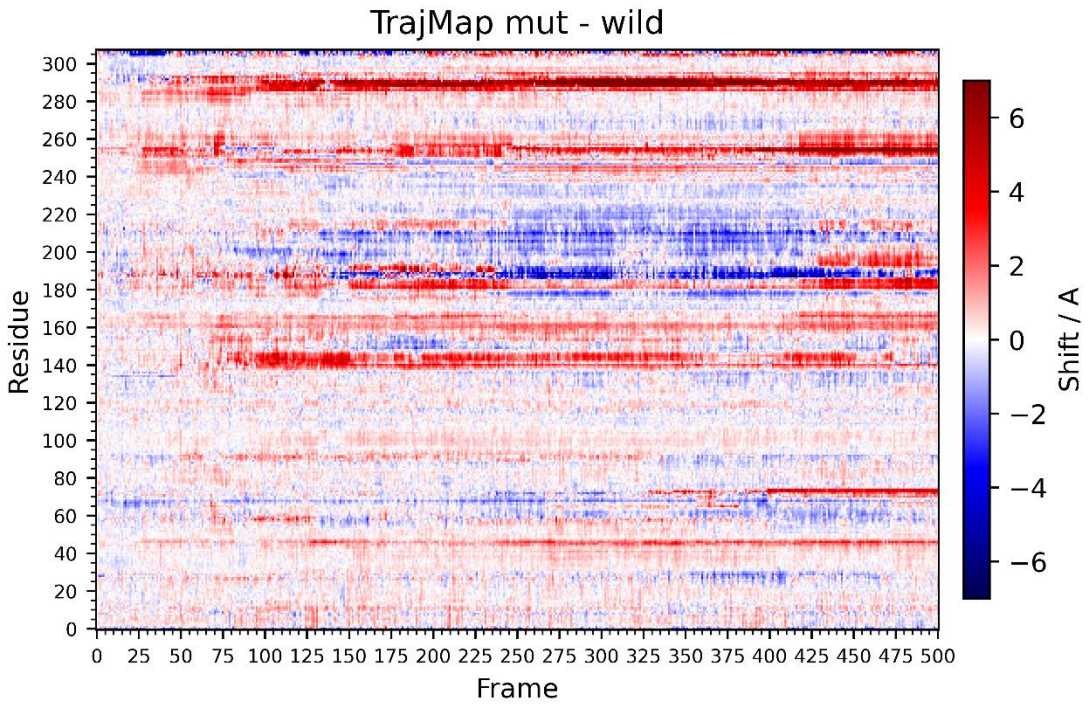
4.3. Single map from an existing .csv

```
import TrajMap_local as tm

file = "data/wild.csv"
residues = 308
matrix = tm.csv2matrix(file, residues)

savefig = "figs/wild.png"
title = "wild type protein"
params = [25, 5, 20, 5, 0, 7, 308, 0, 0]
tm.matrix2map(matrix, savefig, title, params)
```



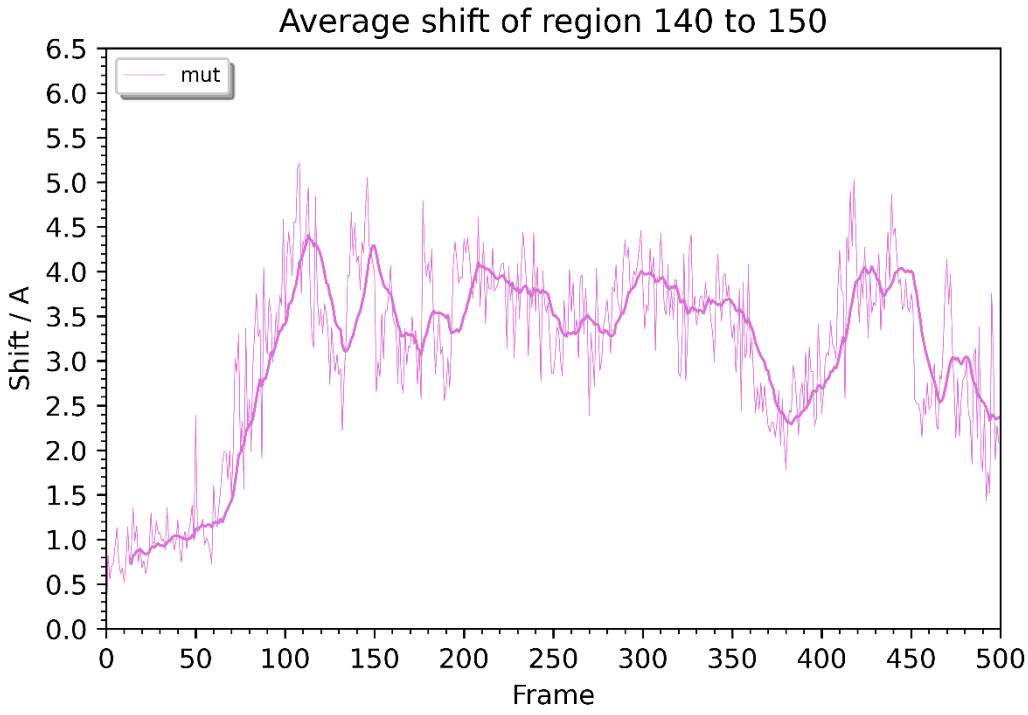
4.4. Difference map from .csv files

```
import TrajMap_local as tm

file = "data/mut.csv"
mut = tm.csv2matrix(file, residues)

file = "data/wild.csv"
wild = tm.csv2matrix(file, residues)

diff_matrix = mut - wild

savefig = "figs/mut_wild"
title = "mut - wild"
params = [25, 5, 20, 5, -7, 7, 308, 1, 0]
tm.matrix2map(diff_matrix, savefig, title, params)
```



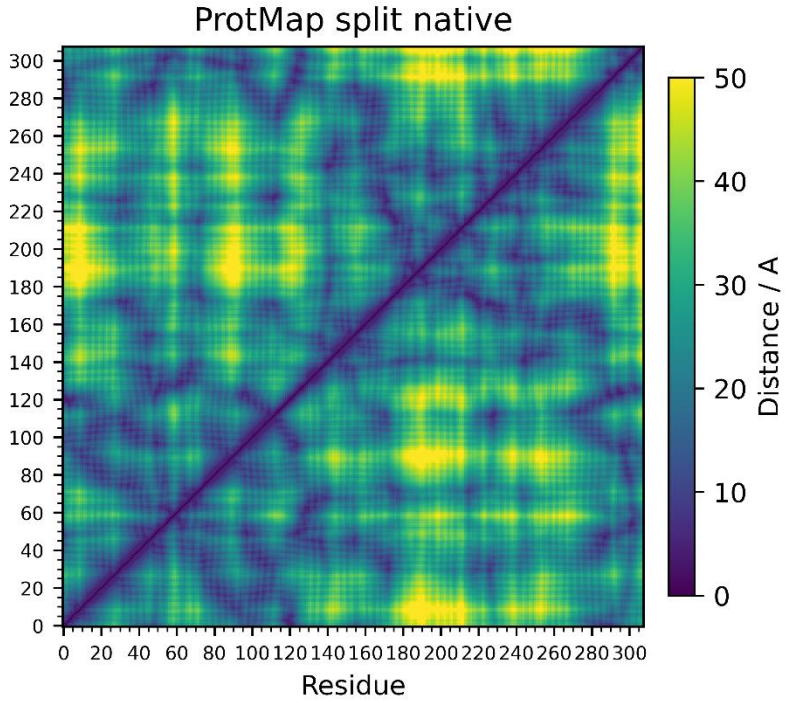
4.5. Shift graph from a loaded matrix

```
import TrajMap_local as tm

params = [140, 150, 0, 500]
avg = tm.matrix2avg(mut, params)

savefig = "figs/shift.png"
title = "Average shift of region 140 to 150"
label = "mut"
y_params = [0, 7, 0.5, 0.1]
x_params = [0, 500, 50, 10]
avg_params = [15, "orchid"]
tm.avg2graph(avg, savefig, title, label,
             y_params, x_params, avg_params)
```



4.6. Protein heatmap

```
import TrajMap_local as tm

file = "data/split.pdb"
savefig = "figs/protmap_split.png"
title = "split native"
save = ["n", "-"]
load = ["n", "-"]
params = [20, 5, 0, 50, 308]
tm.protmap(file, savefig, title, save, load, params)
```



5. Project status & roadmap

The project is in its earliest stages of development and is worked on solely by the author Matej Kožić. **The module has not been thoroughly tested, peer-reviewed, or officially published in a journal.**

Future releases will include additional functionality such as overlaying graphs (such as RMSD and R_{gyr}) on the map for easier assignation of correlation with conformational changes, support for peptide chains with more terminuses, support for maps with reference frame being the previous frame (t_i against t_{i-1}), overall improvements, and more.

All feedback is encouraged and welcomed, including but not limited to ideas for features, suggestions, criticisms, bug reports, and anything else.

The author is open for collaborations. The author kindly asks the project to be appropriately credited if used in any meaningful way.

Repository: github.com/matkozic/TrajMap

Contact: mkozic@chem.pmf.hr