WIP readme

structure file handling

CTS can load and parse .pdb and .cif formats as initial macromolecular inputs. Typically, each file will contain a single model, see below for usage for how CTS can make use of multi-model inputs. After loading a file and parsing its data, CTS will attempt to save a .mat file with the same base name as the input file that records all details necessary for CTS. This .mat file will load much faster than the original .pdb/.cif, but will contain identical information.

Classification ID

Each model is assigned a class ID that determines how it is stored. During model generation, parallel to the model array itself each class has its array that stores the location of all of its members present in the whole model. Class IDs are extracted from the filename. Filenames are split into segments by . and \_\_ (double underscore), and for single-model files the first segment is used as the class ID. Otherwise, if there are 2 more segments than models the leading segments will become the class ID for the corresponding model. Anything else will use the first filename segment for all model IDs, appended by a number.

Examples:

actin.pdb -> class ID is actin if a single model, or actin\_1 actin\_2 … actin\_n for multiple models.

actin\_\_cofilactin.bundle.mmcif -> the first model is ID actin, the second cofilactin.

\*matlab quirk: due to internal handling, class IDs need to start with a standard letter. If they do not, fix\_ will be prepended to them to try to avoid malfunctions.

Multimodel file handling

CTS can generate a handful of organizations of macromolecules, dictated by the last part of the filename before the extension. The modeling process iteratively selects a files’ group for placement, and it can attempt to do so in a few ways. By default, it randomly selects one model from the set present from the original file and attempts placement. This default is the ‘single’ or ‘group’ method. Other methods require the method to be indicated in the filename, they are ‘bundle’ ‘cluster’ ‘complex’ and ‘assembly’.

Cluster attempts to place more random members of the file nearby for a clumpy grouping.

Example use case: clusters of ribosomes

Bundle is similar to cluster, but the randomization restricts models to the same long-axis orientation as the initial model placement, producing filamentous bundles. This requires models to be oriented along the same axis in their original input file, though they do not need to overlap.

Example use case: bundles of actin/cofilactin, pure and mixed

Complex places every model from a file group with no relative movements, as if they were a single model entity but still records them separately. This is useful for protein complexes where subunit information is still of interest.

Example use case: separating barrel and cap domains of groEL

Assembly is similar to complex, but only the first model is placed. Other members are randomly included, and it is possible for all or none to be placed in the model.

Example use case: inconsistent protein complex segmentation

Using these methods requires using method 2 for file naming, with class IDs leading two additional segments, the last of which is the method. The second-to-last segment is useful as a description.

Examples:

cofilin\_\_cofilin\_\_actin\_\_actin\_\_x3-x4\_long.bundle.pdb - 2 cofilin models and 2 actin models that will be placed via the ‘bundle’ method. The ‘x3-x4\_long’ segment is a description for the mixed lengths of the models the file contains.

ribo\_\_ribo\_\_4ug0\_4v6x.group – 2 different ribosomes as a single group to increase variability of a model without increasing ribosome abundance. The descriptor lists the source PDB files used.