WIP readme

structure file handling

CTS can use .pdb and .cif formats (and the .mat files they are converted into) as atomic structures. Cif formats are preferred for offering additional features that CTS can make use of. These files can contain a single model or multiple submodels – CTS handles both. After loading one of the other file types, CTS will also save a copy as a .mat file that contains only necessary information that loads much faster than .pdb or .cif files.

Structure files are handled according to information in the filename, and .cif submodel names. There’s two important pieces of information: the classification ID for each submodel, and the flags for the model(s). A .cif can instead be saved with submodel names of their class ID, to reduce potentially enormous filename extension. Class ID and flags are delimited by either ‘.’ (a period as a file extension) or ‘\_\_’ (double underscore). If only a single class ID is present, it will apply to all submodels – useful for including multiple conformations/variations of one type of particle in a single file.

Example pdb: kinase\_\_hub\_\_CaMK2\_5u6y.complex.vesicle.pdb (two submodels, ID kinase and hub)

Example cif: GABAar.membrane.complex (five submodels, ID are their names)

Class ID must appear first, and apply to submodels in the same order (ID1 to the first submodel, etc). Flags can appear in any order, and are also useful for storing identification information in the filename, like source PDB codes and the conformation or full name of the protein. Flags used this way don’t impact model generation, they are only functional when they match the specific flag strings.

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.