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 submodel ID are stored as submodel 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.

\*matlab quirk: ID names must each begin with a letter. Any other character (including a number) breaks the object field, and will be messily prepended to prevent this.

Multimodel file handling - flags

Flags are used to determine the handling during model generation for objects with special considerations. Default behavior randomly selects a group (effectively a file), a model from that group, and places it into a random location in the sample. This works for isolated individual proteins, so they don’t need flags. Multiple flags can always be used together, but most will don’t ‘combine’, only one relevant flag will be chosen at random. Bundle and membrane for instance can’t happen simultaneously, each event will have a 50% chance for each handling method. Cytosol and vesicle work with everything (except membrane which will be obvious)

Valid flags:

Bundle, cluster, complex, assembly, membrane, vesicle, cytosol

Cluster

The cluster flag places a number of particles randomly drawn from the same group into the same region, causing clumpy patches for that group. Example use case: clusters of ribosomes

Bundle

As cluster, but the cluster is constrained with the same particle orientation along one axis. Can create bundles of protein filaments. Inputs must be well-aligned along the same axis between submodels for bundling behavior, though they don’t need to overlap. Example use case: bundles of actin filaments

Complex

A complex is a macromolecular complex defined in multiple submodels. Generally, this is a protein complex separated into individual protein submodels or one large protein broken into domain submodels. When this flag is active, all submodels are placed into the model with the same arrangement, not just a random submodel. They are stored by class ID, so submodels of different class will be labeled differently in the output atlas for segmenting protein complexes/domains. Example use case: segmenting DNA from histones in chromatin

Assembly

Variant of complex that always places the first submodel, and randomly places 0-all of the remaining submodels. Useful for scaffolds that are associated with a variable number of other subunits. Example use case: variable numbers of MIPs housed inside a microtubule segment

Membrane

The protein is membrane-associated. Includes transmembrane or membrane surface attachment.