Structure Project Instructions

1. Search RNA CoSSMos for secondary structure of interest.
    a. http://cossmos.slu.edu
    b. Experiment parameters:
        i. Only X-ray diffraction
        ii. Select motif of interest
    c. Save search to your CoSSMos account
    d. Download Results
2. Download and Clip .pdb files (this can take several hours to run)
    a. Save clip.py script in same directory as CoSSMos results
    b. Open terminal or command line in the same directory
    c. Run clipping script: > python clip.py (filename of results).txt
    d. This script creates two folders
        i. "pdbs"—folder contains full .pdb files
        ii. "clips"—folder contains clipped and renumbered .pdb files
    e. In the "clips" folder:
        i. Clipped .pdb files are named as follows:
           pdb id_sequence_chain+1st residue of loop.pdb
        ii. h_removed folder and H_atoms_removed.txt
            1. These are files that contained H atoms in original .pdb file that were removed during clipping process
        iii. pdbfile_rejects.txt
            1. Files that were empty
        iv. periods_removed.txt
            1. Files that contained periods in residue id
3. Check .pdb files
    a. Save check_files.py script in same directory
    b. Run check files script: > python check_files.py
    c. This script checks to make sure that clipped .pdb files are:
        i. Not empty
        ii. Are not missing atoms
        iii. Do not contain multiple coordinates for the same atom
    d. This script creates 2 folders
        i. "good_clips"—folder that contains files that pass the quality check
        ii. "bad_clips"—folder that contains files that did NOT pass quality check
    e. In the "bad_clips" folder:
        i. bad_file_data.txt provides details about rejected files
            1. Normal P count = 3
            2. Normal sugar count = 9
            3. Normal total count > 12
            4. Normal mult. atom count = 0
            5. Normal is_empty = 0
4. Determine representative structures
    a. Save Avg-Structure.py script in "good_clips" folder

   b. Navigate to “good_clips” folder in terminal
   c. Run average structure script: > python avg_structure-kr.py
   d. This script creates “best-fits” folder that contains selected representative structures
   e. For each unique sequence:
      i. Creates average structure .pdb file
      ii. Creates RMSD text file
         1. The 1st structure is fixed and all others are superimposed onto it to calculate RMSDs
      iii. Creates results text file
         1. 1st column: 0 = NOT representative structure
            1 = representative structure
         2. 2nd column: # of structures with the same sequence
         3. 3rd column: .pdb filenames
         4. 4th column: RMSD of that structure to the average structure
         5. 5th column: RMSD of that structure to the representative structure
   f. Note---RMSD values are calculated using ALL atoms

5. Calculate RMSDs of representative structures
   a. Save RMSD.py script in “best-fits” folder
   b. Navigate to the “best-fits” folder in the terminal
   c. Run RMSD script: > python RMSD.py
   d. This script generates rmsd.txt file that contains all-against-all RMSD values for the representative structures
   e. When calculating the RMSD values, all atoms in the sugar-phosphate backbone are used along with 3 atoms from each base
      i. A or G (N9, C8, and C4)
      ii. U or C (N1, C2, C6)

6. Cluster and create group folders
   a. Copy “best-fits” folder and rename with desired RMSD cut-off
   b. Save RMSD_Tree_Folders.py script in folder
   c. Navigate to the folder in the terminal
   d. Run script: > python RMSD_Tree_Folders.py
   e. *When asked for the RMSD cutoff, type in numerical value
   f. This script creates seq_analysis.txt
      i. Walks the tree from each node back to the root and analyzes sequence composition at each branch point

```
                          Branch Pt: Consensus Seq.  Degenerate Seq.
                             0 1  2  3  4  5  6  (Positions in tetraloop)
Fraction of               A:
each base at  ------->    C:
each position             G:
                          U:
```

   g. This script creates 2 trees:

            i. UPGMAtree.xml
            ii. UPGMAtree_degenerate.xml (this tree has the inner nodes renamed with the degenerate sequence)

   h. This script moves .pdb files to folders that are named with the degenerate sequence based on the RMSD cutoff
   i. Note—this script requires a modified matrix.py file that includes U in the dictionary
   j. Note—tree .xml files can be viewed with phylogenetic tree software, such as *Archeopteryx*[1]

7. Calculate average and representative structures for each group folder
   a. For each group folder:
      i. Save Avg_Structure-cluster.py file
      ii. Open the terminal and make sure you are in the group folder's directory
      iii. Run script: > python Avg_Structure-cluster.py
   b. This script creates:
      i. A "representative" folder that contains selected representative structure
      ii. Creates average structure .pdb file
      iii. Creates RMSD text file
         1. The 1st structure is fixed and all others are superimposed onto it to calculate RMSDs
      iv. Creates results text file
         1. 1st column: 0 = NOT representative structure
            1 = representative structure
         2. 2nd column: # of structures within group folder
         3. 3rd column: .pdb filenames
         4. 4th column: RMSD of that structure to the average structure
         5. 5th column: RMSD of that structure to the representative structure

   Note---RMSD values are calculated using all atoms in the sugar-phosphate backbone are used along with 3 atoms from each base
      - A or G (N9, C8, and C4)
      - U or C (N1, C2, C6)

8. **Run DSSR on all structures in each group folder (On LINUX Systems)**
   a. Make sure you have DSSR (executable file) and DSSR.sh in each folder
   b. Open the terminal and make sure you are in the group folder's directory
   c. Type "./dssr.sh" into your terminal and hit enter.
   d. This will run DSSR on all files in the current directory, and create a "merged.json" file as output.
9. Parse JSON file
   a. Before we count number of occurrences, we need to parse the json file to get the information we need.

b. Staying in the directory from the previous step, make sure you have python scripts “json_stacks_10_31_17.py”, “json_conf_pucker.py”, “json_hbond.py”, “json_bp.py”, “json_detailedbp5_26.py” and “create.py” in your directory.
c. Open the terminal in the current directory and type “python json_stacks_10_31_17.py” and hit enter. This parses all the stacking information from the json file and exports it to a text file.
d. Go back to your terminal and type “python json_conf_pucker.py” This parses all the base-sugar conformation and sugar pucker information from the json file and exports it to a text file.
e. Go back to your terminal and type “python json_hbond.py” This parses all the non-pairing hydrogen bonding information from the json file and exports it to a text file.
f. Go back to your terminal and type “python json_bp.py” This parses all the pairing hydrogen bonding information from the json file and exports it to a text file.
g. Go back to your terminal and type “python json_detailedbp5_26.py” This parses all the information regarding base pair type from the json file and exports it to a text file.
h. Finally, go back to your terminal and type “python create.py” This will convert all of our text files into CSV files, which makes the counting step easier.

10. Counting number of occurrences
    a. Open the terminal in the same directory you have been working in and check to see if “count_updated_10_31_17.py” is in your folder.
    b. Type “python count_updated_10_31_17.py”. This will tally the number of occurrences of interactions present in each group and calculates a percentage telling us what interactions are common to the majority of structures in each group.
11. **Run DSSR (Windows System)**
    a. Make sure you have the following scripts in your directory:
        i. Dssr-all.py
        ii. Json_stacks_10_31_17.py
        iii. Json_conf_pucker.py
        iv. Json_hbond.py
        v. Json_bp.py
        vi. Json_detailedbp_5_26.py
        vii. Create.py
        viii. Count_Updated_10_31_17.py
    b. Open the iPython terminal
    c. Navigate to the directory with your scripts
    d. Run dssr and the json scripts by typing “%run dssr-all.py”
        i. You should see a window pop up that prompts you to find the location of DSSR on your computer. Select dssr.
        ii. A second window will pop up asking where your PDB files are. Select the folder where your files are located.
        iii. If a security box pops up, click allow.

            iv. You will know the script is finished when you see a json file corresponding to all PDB files in the directory.

12. Convert json files to CSV files
    a. Type “%run create.py”
13. Run counting Script
    a. Type “%run count_updated_10_31_17.py.py”
        i. Tables should appear in terminal
        ii. If you notice an error, just re run the create.py script and the count_updated_10_31_17.py script again.

References

[1] Han, M. V., and Zmasek, C. M. (2009) phyloXML: XML for evolutionary biology and comparative genomics, *BMC Bioinf. 10*, 356.