# Scripting system

Zarek S. Siegel

March 13, 2016

# 1 Summary

## 1.1 Description

High platforms team leverage enthusiastically high infrastructures utilize convergence functionalized action capital. Than growth multimedia viral alternative emerging infrastructures e-enable sucking fashion capital niches standards. Extensible "organic" team re-engineer reinvent quality leading-edge paradigms cultivate infrastructures energistically dynamic. Art holisticly covalent leverage initiatives enterprise interoperable empowerment actualize collaborative invested chains utilize expedite corporate. Improvements potentialities results energistically streamline completely positioning brand adaptive team visualize of holistic infrastructures.

## 1.2 File Structure

- NOTES:
  - ***bolded-italicized*** files/directories need to be created by the user
  - **bolded** files/directories/names need to be modified by the user
  - *italicized* files and directories are created by scripts
  - `plain` files and directories do not (and should not) by modified
  - `$ command` indicates a command line command in the terminal
  - directories and files (and parts of file names) in lower case must included exactly as indicated
  - directories and files in all caps should be named appropriately to the proteins and dockings in question

- ***base_dir*** – The root of the system is the base directory, containing all other files (probably best not to put anything else in this folder but what's indicated below)
  - `Readme.md` –This file
  - ***Dockings.csv*** – A spreadsheet containing master docking parameters
  - ***Gridboxes.csv*** – A spreadsheet specifying all the grid box parameters
  - `ligsets/` –A directory containing all the sets of ligands
    - ***ligsets/LIGSET/*** – for each set, the there should be a directory within the `ligsets/` directory, whose name is the name of the ligand set. Substitute `LIGSET` for some appropriate name (e.g. `ligsets/my_awesome_ligset/`)
      - ***ligsets/LIGSET/pdbqts/*** –a directory containing a PDBQT file for each ligand in the set (whose name is `LIG.pdbqt`, with `LIG` being exactly the same as in `ligsets/LIGSET/LIGSET_list.txt`) [may be scripted later]
      - ***ligsets/LIGSET/LIGSET_list.txt*** –a text file containing a list of all the ligands (one on each line) and nothing else
        - you can create this easily on the command line using: `$ cd base_dir/ligsets/; for l in $(ls LIGSET/pdbqts | sed 's/.pdbqt//'); do echo $l >> LIGSET/LIGSET_list.txt ; done` (appropriately substituting `base_dir` and `LIGSET`, and assuming the PDBQTs are already made)
      - [optional/preliminary] `ligsets/LIGSET/LIGSET.cdxml/` or `ligsets/LIGSET/LIGSET.mol/`– one file to show all the ligands in one page for presentations, PDFmaking and such.
      - [optional/preliminary] `/mols/` and `ligsets/LIGSET/pdbs/` – directories for preparing the initial PDBQT files. Will either be optional, or more scripts will be written.
  - `parameters_csvs/` –A directory containing small CSV files specifying the docking parameters for each individual docking (needed for scripts, at least as of now). Generated by `scripts/write_params_csv.R` from information in `Gridboxes.csv`
    - They will be named **`parameters_csvs/DOCKING_parameters.csv`**, where `DOCKING` is the docking ID
  - `vina_submit_shs/` – A directory containing the submission files for

Vina jobs on the (Wesleyan) cluster. Generated by `write_vina_submit` function of `docking_data_assembly.py`

- **`vina_submit_shs/vina_submit_DOCKING.sh`** –for a docking of 20 models or less, a single submission script is written (and submitted using `$ bsub < vina_submit_DOCKING.sh`, see submission instructions below)
- **`vina_submits_DOCKING/`** –dockings of more than 20 models need to be submitted with multiple scripts (because Vina will not generate more than 20 poses). In this case, the `write_vina_submit` function will create a directory called `vina_submits_DOCKING/` containing $n$ scripts **`vina_submit_DOCKING.1.sh`** , **`vina_submit_DOCKING.2.sh`** through **`vina_submit_DOCKING.n.sh`** . This script is set up to write each $n$ submission scripts, where each of which script has a model number of 20 and $n$ is the number of models divided by 20. Therefore, if greater than 20, the number of models should always be a multiple of twenty, or things will get messed up. (These are submitted used `$ for s in $(ls vina_submits_DOCKING); do bsub < $s; done$`, see instructions below)

◦ ***`PROTEIN/`*** , etc. – A directory for *each* protein, whose name is the name of the protein, the reference name/abbreviation used throughout, it just needs to be consistent (e.g. I primarily dock the proteins HepI and p300 and have the directories `hepi/` and `p300/` in my `base_dir/`)

- ***`PROTEIN/PROTEIN.pdbqt`*** – the PDBQT file to be used for docking (`PROTEIN` must be *exactly* the same for the directory/file names and in the `Dockings.csv` and `Gridboxes.csv` entries for the protein's dockings) [may be scripted later, from `PROTEIN.pdb`]
- [optional/preliminary] `PROTEIN.pdb` – the original PDB file
- **`PROT/DOCKING/`** – for every docking, there should be a directory whose name in the docking ID (the same as in `Dockings.csv`). Note: the user shouldn't make this folder, it is made by `vina_submit_DOCKING.sh`.
  - This will eventually contain
- ***`PROT/binding_sites/`*** – a directory containing binding site PDBs:
  - To do binding site scoring by residues contacted (which is detected by the AutoDockTools script `process_VinaResult.py`),

There must be one or more `PROT/binding_sites/BINDING_SITE.pdb` files. These are subsets of the original `PROTEIN.pdb` (I originally created mine by grabbing the residues within $5\AA$ of the bound ligands that came with the crystal structure, but it could be done many other ways).

- `scripts/` – all the scripts needed to use this set up
  - `write_params_csv.R` – writes `DOCK_parameters.csv` using information in `Dockings.csv` and `Gridboxes.csv`
  - `load_parameters.sh` – loads parameters from `DOCKING_parameters.csv` (used in the next script)
  - `separate_vina_results.sh` – runs Vina result PDBQTs through the AutoDockTools script `process_VinaResult.py`, which separates the poses into separate files and extracts the receptor contacts. The resulting files ended up in `DOCKING/processed_pdbqts/`, named `DOCKING_LIGAND_mMODEL.pdbqt`, where MODEL is the particular pose represented by the file. (A docking with $l$ ligands and with Vina set to produce $m$ models will therefore end up having $l \times m$ files after this script runs.)
  - `cleanup_processed_vina_results.sh` – cleans up processed PDBQTs (`DOCKING/processed_pdbqts/DOCKING_LIGAND_mMODEL.pdbqt`) and converts then to PDBs using the AutoDockTools script `pdbqt_to_pdb.py`
  - `parse_pdb.py` – defines a object class called `Pdb` for parsing PDB and PDBQT files for their 3D coordinates and in the case of processed Vina results, their binding energy, protein contacts, and other data generated by Vina. (Necessary for `docking_data_assembly.py`.)
  - `aiad_icpd.py` – defines functions to calculate the AIAD (averaged inter–atomic distance) and ICPD (inter–centerpoint distance) between two `Pdb` objects, two useful parameters for determining where a pose is binding on a protein and clustering poses together based on proximity. (Necessary for these functions in `docking_data_assembly.py`.)
  - `docking_data_assembly.py` – defines an object class called `Docking` for preparation and analysis of dockings. Contains several important functions:
    - `write_vina_submit` – prepares Vina job submission scripts
    - `assemble_dic` – assembles a data dictionary that contains all

the mined data from the Vina results
    - `score_binding_sites` – scores each pose for the proportion of residues contacted in each reference binding site
    - `assess_all_resis` – uses binding scores to determine a True/-False for each pose binding in at each binding site (based on a threshold score, currently 0.1 or 10%)
    - (`aiad_icpd_binding_sites`) – calculates AIAD and ICPD scores for each pose compared to each binding site
    - `write_alldata_csv` – writes the data dictionary to a CSV file called `DOCKING_alldata.csv`
    - `cluster_poses` [`prepare_clustering_csv.py`] – calculates AIAD scores for every pose compared to every other pose, for cluster analysis later on
- `pre_and_post_control.py` – links to all of the above scripts to coordinate their function, providing the global variables required to run this system on different computers.
- **Other scripts that may or may not be added:**
    - A script to add an entry to `Dockings.csv` or `Gridboxes.csv` with use input

## 1.3 Complete listing of included files

- **Notes**:
    - ... indicates more of the same kind of file or directory
    - .py files (except for the control script) may have a compiled .pyc file with them
    - (files) are optional (for now)
    - [files] are works in progress and may not be included ultimately
    - [[files]] need to be made

```
/path/to/base_dir/
  Readme.md
  Dockings.csv
  Gridboxes.csv
  ligsets/
    LIGSET1/
      LIGSET1_list.txt
      (LIGSET1.cdxml)
      (mols/...)
      (pdbs/...)
```

```
          pdbqts/
            LIG1_1.pdbqt
            LIG1_2.pdbqt
            LIG1_3.pdbqt
            ...
        LIGSET2/
          LIGSET1_list.txt
          (mols/...)
          (pdbs/...)
          pdbqts/
            LIG2_1.pdbqt
            LIG2_2.pdbqt
            LIG2_3.pdbqt
            ...
        ...
      vina_submit_shs/
        vina_submit_A1.sh
        vina_submits_A2/
          vina_submit_A2.1.sh
          vina_submit_A2.2.sh
          vina_submit_A2.3.sh
          ...
        vina_submit_B1.sh
        ...
      PROT_A/
        (PROT_A.pdb)
        PROT_A.pdbqt
        binding_sites/
          BINDING_SITE_ALPHA1.pdb
          BINDING_SITE_ALPHA2.pdb
          BINDING_SITE_ALPHA3.pdb
          ...
        A1/
        A2/
        A3/
        ...
      PROT_B/
        (PROT_B.pdb)
        PROT_B.pdbqt
        binding_sites/
          BINDING_SITE_BETA1.pdb
          ...
        B1/
        B2/
        ...
```

```
      ...
      scripts/
        [new_grid_or_dock_entry.R]
        [[write_ligset_list_txt.sh]]
        load_parameters.sh
        [[ligand, protein preparation]]
        separate_vina_results.sh
        cleanup_processed_vina_results.sh
        parse_pdb.py
        aiad_icpd.py
        [[prepare_clustering_csv.py]]
        docking_data_assembly.py
        pre_and_post_control.py
        [[post_docking_graphs.R]]
        [[clustering_graphs.R]]
        [[R cript to select poses to view in PyMol]]
        [[Py script to load PyMol sessions from lists]]
```

### 1.3.1 After docking post-processing:

```
      PROTEIN/
        (PROTEIN.pdb)
        PROTEIN.pdbqt
        binding_sites/
          SITE1.pdb
          SITE1.pdb
          SITE1.pdb
          ...
        DOCKING/
          result_pdbqts/
            DOCKING_LIG1_results.pdbqt
            DOCKING_LIG2_results.pdbqt
            DOCKING_LIG3_results.pdbqt
            ...
          processed_pdbqts/
            DOCKING_LIG1_m1.pdbqt
            DOCKING_LIG1_m2.pdbqt
            DOCKING_LIG1_m3.pdbqt
            ...
            DOCKING_LIG2_m1.pdbqt
            DOCKING_LIG2_m2.pdbqt
            DOCKING_LIG2_m3.pdbqt
```

```
            ...
            DOCKING_LIG3_m1.pdbqt
            DOCKING_LIG3_m2.pdbqt
            DOCKING_LIG3_m3.pdbqt
            ...
          processed_pdbs/
            DOCKING_LIG1_m1.pdb
            DOCKING_LIG1_m2.pdb
            DOCKING_LIG1_m3.pdb
            ...
            DOCKING_LIG2_m1.pdb
            DOCKING_LIG2_m2.pdb
            DOCKING_LIG2_m3.pdb
            ...
            DOCKING_LIG3_m1.pdb
            DOCKING_LIG3_m2.pdb
            DOCKING_LIG3_m3.pdb
            ...
          DOCKING_alldata.csv
          DOCKING.p
          DOCKING_clustering.csv [[DOCKING_pose_pose_aiads.csv]]
          [[DOCKING_best_aiad_pairs.csv]]
          [[graphs/]]
            [[...graphs...]]
```

## 1.4 Example Dockings.csv file

```
Docking ID,Date,Protein,Ligset,Grid box,Exhaustiveness,Number of Models,Number
     of CPUs,Notes
A1,20160301,PROTA,LIGS1,AAS,20,10,2,looking at active size of protein A
A2,20160308,PROTA,LIGS2,AWP,50,400,4,high volume docking of whole protein A
B1,20160308,PROTB,LIGS3,BWP,8,20,1,initial docking of whole protein B
```

## 1.5 Example Gridboxes.csv file

```
Gridbox Name,Protein,Size in x-dimension,Size in y-dimension,Size in z-
    dimension,Center in x-dimension,Center in y-dimension,Center in z-
    dimension,Notes
```

| Docking ID | Date | Protein | Ligset | Grid box | Exhaustiveness | Number of Models | Number of CPUs | Notes |
|---|---|---|---|---|---|---|---|---|
| A1 | 20160301 | PROTA | LIGS1 | AAS | 20 | 10 | 2 | looking at active size of protein A |
| A2 | 20160308 | PROTA | LIGS2 | AWP | 50 | 400 | 4 | high volume docking of whole protein A |
| B1 | 20160308 | PROTB | LIGS3 | BWP | 8 | 20 | 1 | initial docking of whole protein B |

```
AAS,PROTA,60,72,88,41.89,2.69,-1.85,active site of protein A
AWP,PROTA,126,126,126,41.89,2.69,-1.85,all of protein A
BWP,PROTB,126,126,126,4.89,-5.27,12.0,all of protein B
```

| Gridbox Name | Protein | Box size (x) | Box size (y) | Box size (z) | Box center (x) | Box center (y) | Box center (z) | Notes |
|---|---|---|---|---|---|---|---|---|
| AAS | PROTA | 60 | 72 | 88 | 41.89 | 2.69 | -1.85 | active site of protein A |
| AWP | PROTA | 126 | 126 | 126 | 41.89 | 2.69 | -1.85 | all of protein A |
| BWP | PROTB | 126 | 126 | 126 | 4.89 | -5.27 | 12.0 | all of protein B |

# 2 Scripts

## 2.1 `constants.py`

### 2.1.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
```

```
    self.list_coords()
    self.get_centerpoint()

def threeD_distance(triple1, triple2):
  x1 = triple1[0]
  y1 = triple1[1]
  z1 = triple1[2]
  x2 = triple2[0]
  y2 = triple2[1]
  z2 = triple2[2]
  distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
  return distance

def caclulate_aiad(pdb1, pdb2):
  molc1 = Molecule(pdb1)
  molc2 = Molecule(pdb2)
  dist_list = []
  for triple1 in molc1.coord_triples:
    dists_from_t1 = []
    for triple2 in molc2.coord_triples:
      dist = threeD_distance(triple1, triple2)
      dists_from_t1.append(dist)
    dist_list.append(min(dists_from_t1))
  return mean(dist_list)

def calculate_icpd(pdb1, pdb2):
  molc1 = Molecule(pdb1)
  molc2 = Molecule(pdb2)
  return threeD_distance(molc1.centerpoint, molc2.centerpoint)

def main():
  # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
    .pdbqt")
#   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
    p27_s2_m142.pdbqt")
#   m1 = Molecule(m1_p)
#   m2 = Molecule(m2_p)
#   caclulate_aiad(m1_p, m2_p)
  pass

if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.2 `new_grid_or_dock_entry.py`

### 2.2.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32 def threeD_distance(triple1, triple2):
33   x1 = triple1[0]
34   y1 = triple1[1]
35   z1 = triple1[2]
36   x2 = triple2[0]
37   y2 = triple2[1]
38   z2 = triple2[2]
39   distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40   return distance
41
42 def caclulate_aiad(pdb1, pdb2):
43   molc1 = Molecule(pdb1)
44   molc2 = Molecule(pdb2)
45   dist_list = []
46   for triple1 in molc1.coord_triples:
47     dists_from_t1 = []
48     for triple2 in molc2.coord_triples:
49       dist = threeD_distance(triple1, triple2)
50       dists_from_t1.append(dist)
51     dist_list.append(min(dists_from_t1))
52   return mean(dist_list)
53
54 def calculate_icpd(pdb1, pdb2):
55   molc1 = Molecule(pdb1)
56   molc2 = Molecule(pdb2)
57   return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59 def main():
60   # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
       .pdbqt")
61 #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
       p27_s2_m142.pdbqt")
62 #   m1 = Molecule(m1_p)
63 #   m2 = Molecule(m2_p)
64 #   caclulate_aiad(m1_p, m2_p)
65   pass
66
67 if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.3 `load_parameters.sh`

### 2.3.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32 def threeD_distance(triple1, triple2):
33   x1 = triple1[0]
34   y1 = triple1[1]
35   z1 = triple1[2]
36   x2 = triple2[0]
37   y2 = triple2[1]
38   z2 = triple2[2]
39   distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40   return distance
41
42 def caclulate_aiad(pdb1, pdb2):
43   molc1 = Molecule(pdb1)
44   molc2 = Molecule(pdb2)
45   dist_list = []
46   for triple1 in molc1.coord_triples:
47     dists_from_t1 = []
48     for triple2 in molc2.coord_triples:
49       dist = threeD_distance(triple1, triple2)
50       dists_from_t1.append(dist)
51     dist_list.append(min(dists_from_t1))
52   return mean(dist_list)
53
54 def calculate_icpd(pdb1, pdb2):
55   molc1 = Molecule(pdb1)
56   molc2 = Molecule(pdb2)
57   return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59 def main():
60   # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
       .pdbqt")
61 #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
       p27_s2_m142.pdbqt")
62 #   m1 = Molecule(m1_p)
63 #   m2 = Molecule(m2_p)
64 #   caclulate_aiad(m1_p, m2_p)
65   pass
66
67 if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.4 `separate_vina_results.sh`

### 2.4.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32 def threeD_distance(triple1, triple2):
33   x1 = triple1[0]
34   y1 = triple1[1]
35   z1 = triple1[2]
36   x2 = triple2[0]
37   y2 = triple2[1]
38   z2 = triple2[2]
39   distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40   return distance
41
42 def caclulate_aiad(pdb1, pdb2):
43   molc1 = Molecule(pdb1)
44   molc2 = Molecule(pdb2)
45   dist_list = []
46   for triple1 in molc1.coord_triples:
47     dists_from_t1 = []
48     for triple2 in molc2.coord_triples:
49       dist = threeD_distance(triple1, triple2)
50       dists_from_t1.append(dist)
51     dist_list.append(min(dists_from_t1))
52   return mean(dist_list)
53
54 def calculate_icpd(pdb1, pdb2):
55   molc1 = Molecule(pdb1)
56   molc2 = Molecule(pdb2)
57   return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59 def main():
60   # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
       .pdbqt")
61 #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
       p27_s2_m142.pdbqt")
62 #   m1 = Molecule(m1_p)
63 #   m2 = Molecule(m2_p)
64 #   caclulate_aiad(m1_p, m2_p)
65   pass
66
67 if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.5 `cleanup_processed_vina_results.sh`

### 2.5.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32  def threeD_distance(triple1, triple2):
33    x1 = triple1[0]
34    y1 = triple1[1]
35    z1 = triple1[2]
36    x2 = triple2[0]
37    y2 = triple2[1]
38    z2 = triple2[2]
39    distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40    return distance
41
42  def caclulate_aiad(pdb1, pdb2):
43    molc1 = Molecule(pdb1)
44    molc2 = Molecule(pdb2)
45    dist_list = []
46    for triple1 in molc1.coord_triples:
47      dists_from_t1 = []
48      for triple2 in molc2.coord_triples:
49        dist = threeD_distance(triple1, triple2)
50        dists_from_t1.append(dist)
51      dist_list.append(min(dists_from_t1))
52    return mean(dist_list)
53
54  def calculate_icpd(pdb1, pdb2):
55    molc1 = Molecule(pdb1)
56    molc2 = Molecule(pdb2)
57    return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59  def main():
60    # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
        .pdbqt")
61  #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
        p27_s2_m142.pdbqt")
62  #   m1 = Molecule(m1_p)
63  #   m2 = Molecule(m2_p)
64  #   caclulate_aiad(m1_p, m2_p)
65    pass
66
67  if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.6 `parse_pdb.py`

### 2.6.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32  def threeD_distance(triple1, triple2):
33    x1 = triple1[0]
34    y1 = triple1[1]
35    z1 = triple1[2]
36    x2 = triple2[0]
37    y2 = triple2[1]
38    z2 = triple2[2]
39    distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40    return distance
41
42  def caclulate_aiad(pdb1, pdb2):
43    molc1 = Molecule(pdb1)
44    molc2 = Molecule(pdb2)
45    dist_list = []
46    for triple1 in molc1.coord_triples:
47      dists_from_t1 = []
48      for triple2 in molc2.coord_triples:
49        dist = threeD_distance(triple1, triple2)
50        dists_from_t1.append(dist)
51      dist_list.append(min(dists_from_t1))
52    return mean(dist_list)
53
54  def calculate_icpd(pdb1, pdb2):
55    molc1 = Molecule(pdb1)
56    molc2 = Molecule(pdb2)
57    return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59  def main():
60    # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
        .pdbqt")
61  #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
        p27_s2_m142.pdbqt")
62  #   m1 = Molecule(m1_p)
63  #   m2 = Molecule(m2_p)
64  #   caclulate_aiad(m1_p, m2_p)
65    pass
66
67  if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.7 `aiad_icpd.py`

### 2.7.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32 def threeD_distance(triple1, triple2):
33   x1 = triple1[0]
34   y1 = triple1[1]
35   z1 = triple1[2]
36   x2 = triple2[0]
37   y2 = triple2[1]
38   z2 = triple2[2]
39   distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40   return distance
41
42 def caclulate_aiad(pdb1, pdb2):
43   molc1 = Molecule(pdb1)
44   molc2 = Molecule(pdb2)
45   dist_list = []
46   for triple1 in molc1.coord_triples:
47     dists_from_t1 = []
48     for triple2 in molc2.coord_triples:
49       dist = threeD_distance(triple1, triple2)
50       dists_from_t1.append(dist)
51     dist_list.append(min(dists_from_t1))
52   return mean(dist_list)
53
54 def calculate_icpd(pdb1, pdb2):
55   molc1 = Molecule(pdb1)
56   molc2 = Molecule(pdb2)
57   return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59 def main():
60   # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
       .pdbqt")
61 #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
       p27_s2_m142.pdbqt")
62 #   m1 = Molecule(m1_p)
63 #   m2 = Molecule(m2_p)
64 #   caclulate_aiad(m1_p, m2_p)
65   pass
66
67 if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.8 `docking_data_assembly.py`

### 2.8.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32  def threeD_distance(triple1, triple2):
33    x1 = triple1[0]
34    y1 = triple1[1]
35    z1 = triple1[2]
36    x2 = triple2[0]
37    y2 = triple2[1]
38    z2 = triple2[2]
39    distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40    return distance
41
42  def caclulate_aiad(pdb1, pdb2):
43    molc1 = Molecule(pdb1)
44    molc2 = Molecule(pdb2)
45    dist_list = []
46    for triple1 in molc1.coord_triples:
47      dists_from_t1 = []
48      for triple2 in molc2.coord_triples:
49        dist = threeD_distance(triple1, triple2)
50        dists_from_t1.append(dist)
51      dist_list.append(min(dists_from_t1))
52    return mean(dist_list)
53
54  def calculate_icpd(pdb1, pdb2):
55    molc1 = Molecule(pdb1)
56    molc2 = Molecule(pdb2)
57    return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59  def main():
60    # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
        .pdbqt")
61  #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
        p27_s2_m142.pdbqt")
62  #   m1 = Molecule(m1_p)
63  #   m2 = Molecule(m2_p)
64  #   caclulate_aiad(m1_p, m2_p)
65    pass
66
67  if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py

## 2.9 `pre_and_post_control.py`

### 2.9.1 Function

Transcompiler picojava pseudolanguage java ee regular expression syntactic sugar eclipse rust bug. Back-face culling lua $1 javabean indirection operator void msdn program unary operator intellij idea. Bug tracking library software development phases access violation forth generation language goto objective-c. Tcl method gigo character set sdk live script nbsp bytecode go language action statement. Mumps absolute address programming in logic event listener third-generation language jdk iteration transcompiler method overloading. Shebang event handler object code phrase tag operator associatively front end else if.

```
#!/usr/bin/env python

### Calculating average inter-atomic distance
# (c) Zarek Siegel
# v1 3/6/16

from parse_pdb import *
from math import sqrt
from numpy import mean

class Molecule():
  def list_coords(self):
    self.coord_triples = []
    for atom in self.pdb.coords:
      self.coord_triples.append(atom['xyz'])

  def get_centerpoint(self):
    x_coords = []
    y_coords = []
    z_coords = []
    for triple in self.coord_triples:
      x_coords.append(triple[0])
      y_coords.append(triple[1])
      z_coords.append(triple[2])
    self.centerpoint = (mean(x_coords), mean(y_coords), mean(z_coords))

  def __init__(self, pdb):
    self.pdb = pdb
    self.list_coords()
    self.get_centerpoint()
```

```
31
32 def threeD_distance(triple1, triple2):
33   x1 = triple1[0]
34   y1 = triple1[1]
35   z1 = triple1[2]
36   x2 = triple2[0]
37   y2 = triple2[1]
38   z2 = triple2[2]
39   distance = sqrt( ((x2 - x1)**2) + ((y2 - y1)**2) + ((z2 - z1)**2) )
40   return distance
41
42 def caclulate_aiad(pdb1, pdb2):
43   molc1 = Molecule(pdb1)
44   molc2 = Molecule(pdb2)
45   dist_list = []
46   for triple1 in molc1.coord_triples:
47     dists_from_t1 = []
48     for triple2 in molc2.coord_triples:
49       dist = threeD_distance(triple1, triple2)
50       dists_from_t1.append(dist)
51     dist_list.append(min(dists_from_t1))
52   return mean(dist_list)
53
54 def calculate_icpd(pdb1, pdb2):
55   molc1 = Molecule(pdb1)
56   molc2 = Molecule(pdb2)
57   return threeD_distance(molc1.centerpoint, molc2.centerpoint)
58
59 def main():
60   # m1_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/p27_s3_m4
       .pdbqt")
61 #   m2_p = Pdb("/Users/zarek/lab/Docking/p300/p27/res_pdbqts_cleaned/
       p27_s2_m142.pdbqt")
62 #   m1 = Molecule(m1_p)
63 #   m2 = Molecule(m2_p)
64 #   caclulate_aiad(m1_p, m2_p)
65   pass
66
67 if __name__ == "__main__": main()
```

/Users/zarek/GitHub/TaylorLab/zvina/scripts/aiad_icpd.py