**Supplementary file 2**

**Supplementary file 2. Protocol Capture**

**S 1. Generating conformer library with OpenEye Omega and RosettaLigand params – Bash executable**

|  |  |
| --- | --- |
| 1  2  ∙  3  4  5  6  7  8  ∙  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  ∙  24  ∙  25  ∙  26  27  28  29  30  31  32  33  ∙  34  35  36  37  38  39  40  41  42 | #!/bin/bash  # New to OpenEye? Install the openeye toolkit with Conda.  (Read the OpenEye toolkit README)  # Using Python2.7? Look for pip install packages at: https://anaconda.org/OpenEye  rosetta\_src={YOUR\_FOLDER\_PATH}/rosetta\_bin\_linux\_2021.07.61567\_bundle/main/source  openeye\_bin={YOUR\_FOLDER\_PATH}/openeye/bin  **if** [ $# -lt 1 ]; **then**  echo "USAGE: generate-ligand-conformers.sh  <ligand name only (input in .mol2 format)> "  exit  **fi**  drug=$1  dir=**$(**echo **${**PWD**})**  set -v  # Set up directory structure  mkdir -p ligand  # Make ligands  pushd ligand  mkdir -p {fa,cen}/{conf1,confs,kins,withxtal}  omega="${openeye\_bin}/omega2 -includeInput -commentEnergy"  $omega -in $dir/$drug.mol2 -out $drug.omega.mol2 -prefix \_$drug  python **${**rosetta\_src**}**/src/apps/public/ligand\_docking/assign\_charges.py  < $drug.omega.mol2 > $drug.am1bcc.mol2  python **${**rosetta\_src**}**/scripts/python/public/molfile\_to\_params.py -c -nX00  -p$drug -k$drug.kin $drug.am1bcc.mol2  cat **${**drug**}**\_????.fa.pdb | gzip -c > fa/withxtal/**${**drug**}**\_confs.fa.pdb.gz &&  ( [ -f **${**drug**}**\_0002.fa.pdb ] || cp **${**drug**}**\_0001.fa.pdb **${**drug**}**\_0002.fa.pdb )  mv **${**drug**}**\_0001.fa.pdb fa/conf1/  cat **${**drug**}**\_????.fa.pdb | gzip -c > fa/**${**drug**}**\_confs.fa.pdb.gz  mv **${**drug**}**\_????.fa.pdb fa/confs/  echo "PDB\_ROTAMERS ${drug}\_confs.fa.pdb" >> $drug.fa.params  cp $drug.fa.params fa/withxtal/  mv $drug.fa.params fa/  mv $drug.fa.kin fa/kins/  cat **${**drug**}**\_????.cen.pdb | gzip -c > cen/withxtal/**${**drug**}**\_confs.cen.pdb.gz &&  ( [ -f **${**drug**}**\_0002.cen.pdb ] || cp **${**drug**}**\_0001.cen.pdb **${**drug**}**\_0002.cen.pdb )  mv **${**drug**}**\_0001.cen.pdb cen/conf1/  cat **${**drug**}**\_????.cen.pdb | gzip -c > cen/**${**drug**}**\_confs.cen.pdb.gz  mv **${**drug**}**\_????.cen.pdb cen/confs/  echo 'PDB\_ROTAMERS $drug\_confs.cen.pdb' >> $drug.cen.params  cp $drug.cen.params cen/withxtal/  mv $drug.cen.params cen/  mv $drug.cen.kin cen/kins/  popd |

**S2. RosettaLigand – Bash executable**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  ∙  17 | #!/bin/bash  ${rosetta\_bin}/rosetta\_scripts.static.linuxgccrelease **\**  -in:path:database $rosetta\_database **\**  -s ./seed\_files/**${**protein\_ligand\_complex.pdb**}** **\**  -parser:protocol ./seed\_files/dock.xml **\**  -extra\_res\_fa ./seed\_files/**${**file\_handle**}**.params **\**  -ex1 **\**  -ex2 **\**  -no\_optH false **\**  -flip\_HNQ true **\**  -ignore\_ligand\_chi true **\**  -nstruct 10 **\**  -overwrite **\**  -out:pdb true **\**  -out:prefix **${**array\_prefix**}**\_ **\**  -out:file:scorefile **${**array\_prefix**}**\_Rosetta\_ligand\_**${**file\_handle**}**.sc |

**S3. RosettaLigand Docking – XML script**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  ∙  11  ∙  12  ∙  13  14  15  16  ∙  17  ∙  18  ∙  19  20  21  22  ∙  23  ∙  ∙  24  25  26  27  28  29  30  31  ∙  32  ∙  33  ∙  34  ∙  35  36  37  38  39  40  41  42  43  44 | <ROSETTASCRIPTS>  <SCOREFXNS>  <ScoreFunction name="ligand\_soft\_rep" weights="ligand\_soft\_rep">  </ScoreFunction>  <ScoreFunction name="hard\_rep" weights="ligand">  </ScoreFunction>  </SCOREFXNS>  <LIGAND\_AREAS>  <LigandArea name="inhibitor\_dock\_sc" chain="X" cutoff="6.0"  add\_nbr\_radius="true" all\_atom\_mode="false"/>  <LigandArea name="inhibitor\_final\_sc" chain="X" cutoff="6.0"  add\_nbr\_radius="true" all\_atom\_mode="false"/>  <LigandArea name="inhibitor\_final\_bb" chain="X" cutoff="7.0"  add\_nbr\_radius="false" all\_atom\_mode="true" Calpha\_restraints="0.3"/>  </LIGAND\_AREAS>  <INTERFACE\_BUILDERS>  <InterfaceBuilder name="side\_chain\_for\_docking"  ligand\_areas="inhibitor\_dock\_sc"/>  <InterfaceBuilder name="side\_chain\_for\_final"  ligand\_areas="inhibitor\_final\_sc"/>  <InterfaceBuilder name="backbone"  ligand\_areas="inhibitor\_final\_bb" extension\_window="3"/>  </INTERFACE\_BUILDERS>  <MOVEMAP\_BUILDERS>  <MoveMapBuilder name="docking"  sc\_interface="side\_chain\_for\_docking" minimize\_water="false"/>  <MoveMapBuilder name="final"  sc\_interface="side\_chain\_for\_final" bb\_interface="backbone"  minimize\_water="false"/>  </MOVEMAP\_BUILDERS>  <SCORINGGRIDS ligand\_chain="X" width="15">  <ClassicGrid grid\_name="classic" weight="1.0"/>  </SCORINGGRIDS>  <MOVERS>  <Transform name="transform" chain="X" box\_size="7.0"  move\_distance="0.2" angle="20" cycles="500" repeats="1" temperature="5"/>  <HighResDocker name="high\_res\_docker" cycles="6"  repack\_every\_Nth="3" scorefxn="ligand\_soft\_rep" movemap\_builder="docking"/>  <FinalMinimizer name="final" scorefxn="hard\_rep"  movemap\_builder="final"/>  <InterfaceScoreCalculator name="add\_scores" chains="X"  scorefxn="hard\_rep"/>  </MOVERS>  <PROTOCOLS>  <Add mover\_name="transform"/>  <Add mover\_name="high\_res\_docker"/>  <Add mover\_name="final"/>  <Add mover\_name="add\_scores"/>  </PROTOCOLS>  </ROSETTASCRIPTS> |

**S4. Example hdbscan clustering – Bash executable**

|  |  |
| --- | --- |
| 1  2  3  4  ∙  5 | #!/bin/bash  Biomol2Clust=/home/user/Software/Biomol2Clust\_v.1.3/main.py  python3 $Biomol2Clust method=hdbscan noh=true input=./cluster\_inputs  min\_cluster\_size=50 output=./hdbscan\_50 |

**S5. Example PLIP analysis – Bash executable**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #!/bin/bash  plip=/home/user/anaconda2/envs/plip/bin/plip  pdb={YOUR PDB INPUT}  plip -yvpt –nohydro -f $pdb |