Introduction

bcl::Cluster is a general purpose clustering utility. It uses a hierarchical agglomerative clustering algorithm. It takes an input file with pairwise distances between objects that are to be clustered. It outputs information about the clustering that can be used for analysis. Distances can be similarity or dissimilarity measurements. bcl::Cluster can be used to output a python script for visualizing the dendrogram that results from clustering in the Pymol Molecular Graphics System.

For command line options and their descriptions use the help flag :

```
cluster.exe Cluster -help
```

Below are examples demonstrating some command line flag options and giving additional explanation.

Flag Examples

input_format

TableLowerTriangle

```
bcl::storage::Table<double>  1000_0000  1000_0001  1000_0002  1000_0003  1000_0004
1000_0000                            0          0          0          0          0
1000_0001                      13.5371          0          0          0          0
1000_0002                      11.9716    14.9337          0          0          0
1000_0003                      11.7247     8.8339      14.01          0          0
1000_0004                      6.89769    3.24795    4.82608    13.9589          0
```

TableUpperTriangle

```
bcl::storage::Table<double>  1000_0000  1000_0001  1000_0002  1000_0003  1000_0004
1000_0000                            0    13.5371    11.9716    11.7247    6.89769
1000_0001                            0          0    14.9337     8.8339    3.24795
1000_0002                            0          0          0      14.01    4.82608
1000_0003                            0          0          0          0    13.9589
1000_0004                            0          0          0          0          0
```

PairwiseList - objects and distances can be in any column as long as it is consistent throughout the file. The first three lines indicate the columns of object_a, object_b, and the distance. In this example object_a is in column 0, object_b is in column 1, and the distance is in column 2.

```
0
1
2
1000_0000 1000_0001 13.5371
1000_0000 1000_0002 11.9716
1000_0000 1000_0003 11.7247
1000_0000 1000_0004 6.89769
1000_0001 1000_0002 14.9337
1000_0001 1000_0003 8.8339
1000_0001 1000_0004 3.24795
1000_0002 1000_0003 14.01
1000_0002 1000_0004 4.82608
1000_0003 1000_0004 13.9589
```

linkage

Complete – the most different inter-cluster distance between any two members in two clusters (does not take into account intra-cluster distances in the two clusters).

Single – the most similar inter-cluster distance between any two members in two clusters (does not take into account intra-cluster distances in the two clusters).

Average – For two nodes *a* and *b* with members *i* and *j*, it is calculated as $\frac{\sum_{i,j} distance^{i,j}}{size_a \cdot size_b}$

Total - For two nodes *a* and *b* with members *i* and *j*, it is calculated as $\frac{\sum_{i,j} distance^{i,j} + \sum_{i,i} distance^{i,i} + \sum_{j,j} distance^{j,j}}{size_a \cdot size_b + \frac{size_a * (size_a - 1)}{2} + \frac{size_b * (size_b - 1)}{2}}$

This is similar to average linkage but it also takes into account the intra-node members.

output_file

using the TableLowerTriangle input file (in a file named *distances.txt*) from above with the following command line will result in the output files shown below named *cluster_output.Rows.txt* and *cluster_output.Centers.txt*:

```
cluster.exe Cluster –distance_input_file distances.txt -input_format TableLowerTriangle –output_format Rows Centers –linkage Average –output_file cluster_output
```

Rows – outputs information about the dendrogram with one member of a cluster per line. The numberical identifier for the cluster is given in the second column. Every member will have the identifier of the cluster to which it belongs. The name of the member is given in the 6th column, and the size of the cluster to which the member belongs is given in the 10th cluster. If the member is part of a cluster that is at the base of the dendrogram, the 14th column will have a “1” to indicate the member is part of a cluster that is a leaf. The linkage of the cluster in which the member resides is given in the 18th column. The linkage of a cluster that has only one member is undefined and indicated as “nan”. In this example, node 9 has four members and a linkage of 10.8021.

```
NODE 6 : Member : 1000_0000 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0001 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0002 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0003 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0004 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 4 : Member : 1000_0003 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 9 : Member : 1000_0001 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0004 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0002 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0000 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 8 : Member : 1000_0001 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0004 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0002 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 7 : Member : 1000_0001 : Size : 2 : Leaf : 0 : Linkage : 3.24795
NODE 7 : Member : 1000_0004 : Size : 2 : Leaf : 0 : Linkage : 3.24795
NODE 2 : Member : 1000_0001 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 5 : Member : 1000_0004 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 3 : Member : 1000_0002 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 1 : Member : 1000_0000 : Size : 1 : Leaf : 1 : Linkage : nan
```

Centers – analogous to the output from “Rows” above, except that only the center member of each cluster is output. The cluster center member is calculated as the member that is most similar to all other members in the cluster. For each member the distance to all other members is summed, and the member with the sum indicating it is most similar to other members is the center.

```
NODE 6 : Member : 1000_0004 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 4 : Member : 1000_0003 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 9 : Member : 1000_0004 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 8 : Member : 1000_0004 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 7 : Member : 1000_0001 : Size : 2 : Leaf : 0 : Linkage : 3.24795
NODE 2 : Member : 1000_0001 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 5 : Member : 1000_0004 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 3 : Member : 1000_0002 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 1 : Member : 1000_0000 : Size : 1 : Leaf : 1 : Linkage : nan
```

remove_nodes_below_size – using the command line below will remove any nodes that have a size below 2.

```
cluster.exe Cluster –distance_input_file distances.txt -input_format TableLowerTriangle –output_format
Rows Centers –linkage Average –output_file cluster_output -remove_nodes_below_size 2
```

The cluster Rows output is shown below. Notice that all the nodes with 1 member have been removed. Also, node 7 is now a leaf since the two clusters that merged to form it only had one member each, so node 7 is now at the base of the hierarchy. Node 4 with cluster member 1000_0003 has been removed.

```
NODE 6 : Member : 1000_0000 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0001 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0002 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0003 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0004 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 9 : Member : 1000_0001 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0004 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0002 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0000 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 8 : Member : 1000_0001 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0004 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0002 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 7 : Member : 1000_0001 : Size : 2 : Leaf : 1 : Linkage : 3.24795
NODE 7 : Member : 1000_0004 : Size : 2 : Leaf : 1 : Linkage : 3.24795
```

remove_internally_similar_nodes – after clustering is finished, clusters within clusters that have a linkage less than the given value will be removed from the hierarchy. This ensures that all members are represented.

```
cluster.exe Cluster –distance_input_file distances.txt -input_format TableLowerTriangle –output_format Rows Centers
–linkage Average –output_file cluster_output -remove_ internally_similar_nodes 10
```

```
NODE 6 : Member : 1000_0000 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0001 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0002 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0003 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 6 : Member : 1000_0004 : Size : 5 : Leaf : 0 : Linkage : 12.1319
NODE 4 : Member : 1000_0003 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 9 : Member : 1000_0001 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0004 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0002 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 9 : Member : 1000_0000 : Size : 4 : Leaf : 0 : Linkage : 10.8021
NODE 8 : Member : 1000_0001 : Size : 3 : Leaf : 1 : Linkage : 9.87989
NODE 8 : Member : 1000_0004 : Size : 3 : Leaf : 1 : Linkage : 9.87989
NODE 8 : Member : 1000_0002 : Size : 3 : Leaf : 1 : Linkage : 9.87989
NODE 1 : Member : 1000_0000 : Size : 1 : Leaf : 1 : Linkage : nan
```

height_cutoff – the clustering will be stopped when a cluster is formed that has a linkage greater than the supplied cutoff.

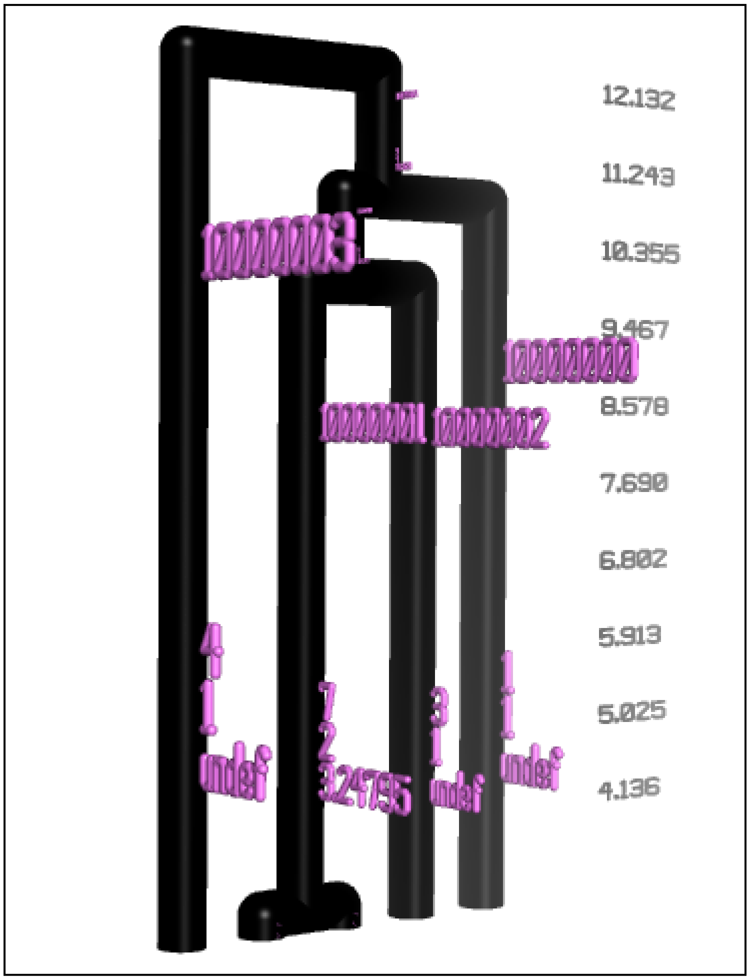
```
cluster.exe Cluster –distance_input_file distances.txt -input_format TableLowerTriangle –output_format Rows Centers –linkage Average –output_file cluster_output -height_cutoff 9
```

```
NODE 6 : Member : 1000_0000 : Size : 5 : Leaf : 0 : Linkage : 9.87989
NODE 6 : Member : 1000_0001 : Size : 5 : Leaf : 0 : Linkage : 9.87989
NODE 6 : Member : 1000_0002 : Size : 5 : Leaf : 0 : Linkage : 9.87989
NODE 6 : Member : 1000_0003 : Size : 5 : Leaf : 0 : Linkage : 9.87989
NODE 6 : Member : 1000_0004 : Size : 5 : Leaf : 0 : Linkage : 9.87989
NODE 1 : Member : 1000_0000 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 4 : Member : 1000_0003 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 8 : Member : 1000_0001 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0004 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 8 : Member : 1000_0002 : Size : 3 : Leaf : 0 : Linkage : 9.87989
NODE 7 : Member : 1000_0001 : Size : 2 : Leaf : 0 : Linkage : 3.24795
NODE 7 : Member : 1000_0004 : Size : 2 : Leaf : 0 : Linkage : 3.24795
NODE 2 : Member : 1000_0001 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 5 : Member : 1000_0004 : Size : 1 : Leaf : 1 : Linkage : nan
NODE 3 : Member : 1000_0002 : Size : 1 : Leaf : 1 : Linkage : nan
```

Here cluster 8 is formed with a linkage of 9.87989, which is above the height cutoff. So, clustering is stopped and cluster 6 just contains all clustered objects and is given the last linkage determined.

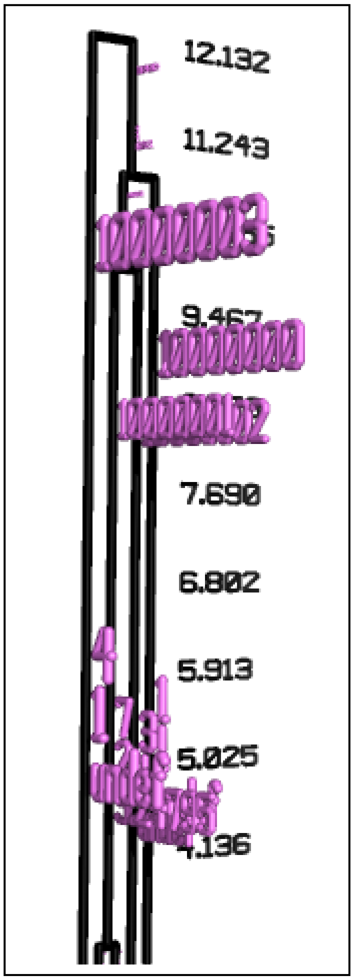
output_pymol – enables creation of a python script that can be run in Pymol to create a dendrogram. In Pymol go to “File” then “Run” then select the appropriate file. Each cluster has text indicating from top to bottom : a) the name of the cluster center member b) the cluster identification c) the size of the cluster d) the linkage of the cluster. In the example below the python script is being output to a file named dendrogram.py.

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 100 25 50 25 10 dendrogram.py
```



The unit length, width, and spacing of the cylinders can be adjusted to change the dimensions of the dendrogram.

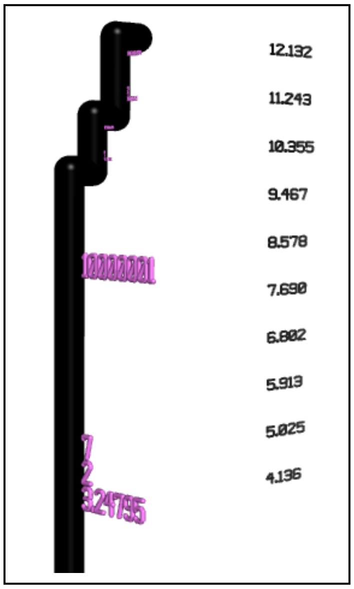
```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 100 5 10 25 10 dendrogram.py
```



The dendrogram will be affected by other flags such as remove_nodes below size :

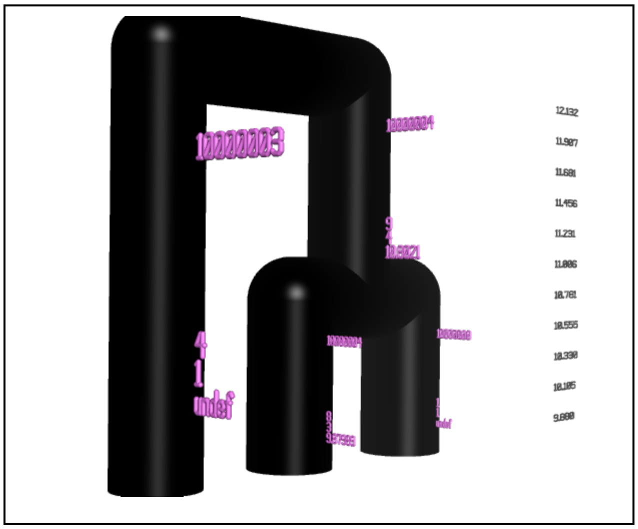
```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 100 25 50 25 10 dendrogram.py -remove_nodes_below_size 2
```

In this example, with such a small number of objects, all the branching is lost since clustering results in members just being added one at a time to cluster number 7.



The dendrogram will also be affected by the remove_internally_similar_nodes flag :

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 100 25 50 25 10 dendrogram.py -remove_internally_similar_nodes 10
```

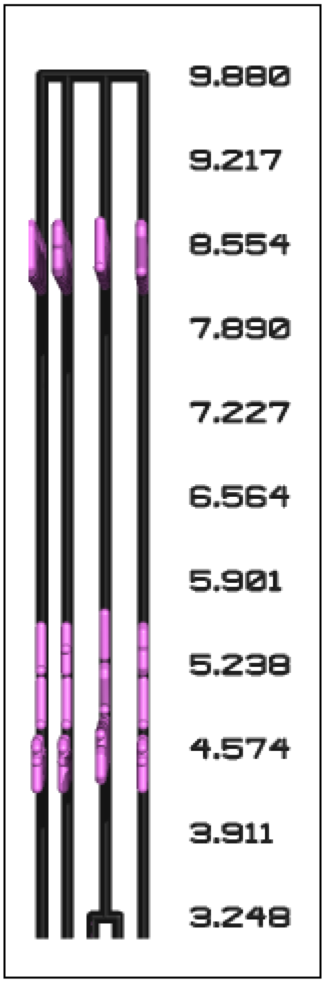


There are three clusters that exist when the dendrogram is cut at the bottom at 10.

The dendrogram will also be affected by the height_cutoff flag :

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 100 5 10 25 10 dendrogram.py -height_cutoff 9
```

At a height cutoff of 9, only one merging has occurred, as seen below.

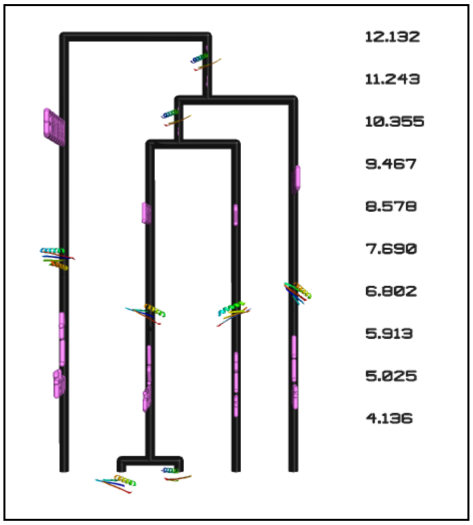


pymol_label_output_string – this is the default method for labeling the dendrogram as seen above

pymol_label_output_protein_model_from_string – the dendrogram will be labeled with the actual protein models. The protein model PDB file names are created from the member names and the prefix and postfix parameters passed to this flag. In this example, the four objects clustered are named `1000_0000`, `1000_0001`, `1000_0002`, `1000_0003`, `1000_0004`. They correspond to Protein Data Base formatted files (PDBs) that are located in "/home/user/pdbs/" and the files start with "model". The PDB files end in "_final.pdb". So the PDB for `1000_0001` is "/home/user/pdbs/model_1000_0001_final.pdb".

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 50 5 50 25 10 dendrogram_a.py -pymol_label_output_protein_model_from_string /home/user/pdbs/model _final.pdb
```

The unit values for the length, radius, and separation of the cylinders in the dendrogram should be adjusted according to the size of the protein.

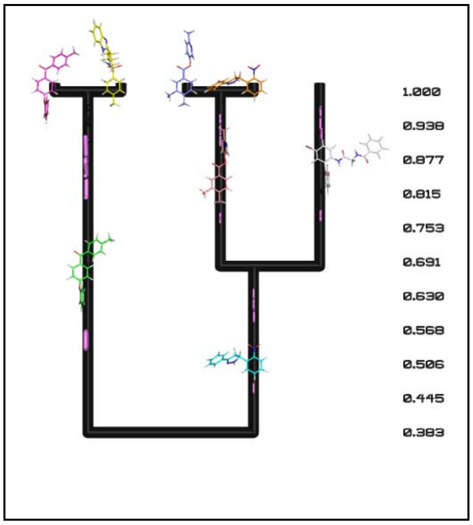


pymol_label_output_small_molecule – the dendrogram will be labeled with molecules taken from an SDF formatted file. In this case, clustering is being done on a similarity measure, so clusters have linkages indicated high similarity are at the top of the dendrogram. When using this flag, the distance input file is assumed to have the objects numbered from 0 to N and the number of the object corresponds to its position within the SDF file. Below is the example of the distance input file.

```
bcl::storage::Table<double> 0 1 2 3 4
0 1.000000 1.000000 0.393939 0.243243 0.222222
1        0 1.000000 1.000000 0.255814 0.181818
2        0        0 1.000000 1.000000 0.368421
3        0        0        0 1.000000 1.000000
4        0        0        0        0 1.000000
```

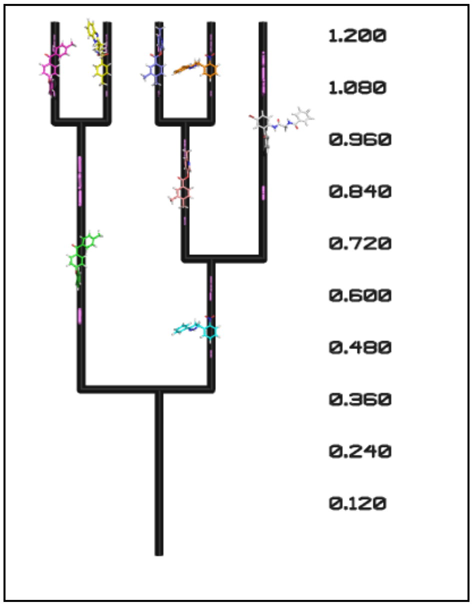
The command line is below. The SDF file is provided as “/home/user/molecules.sdf” :

```
cluster.exe Cluster -output_file cluster_results.txt -distance_input_file distances.txt -input_format TableUpperTriangle -
output_format Centers -linkage Average -distance_definition greater -output_pymol 100 1 10 20 10 dendrogram.py -
pymol_label_output_small_molecule /home/user/molecules.sdf
```



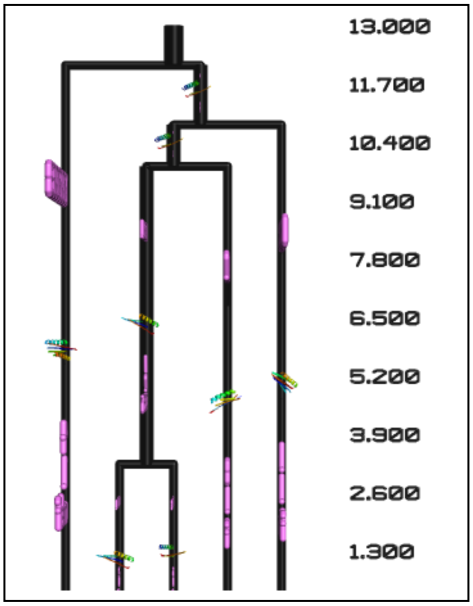
pymol_set_min_max_girth – the minimum and maximum that the dendrogram reaches to can be set with this flag. Compare the dendrogram below to the dendrogram above.

```
cluster.exe Cluster -output_file cluster_output.txt -distance_input_file distances.txt -input_format TableUpperTriangle -output_format Centers -linkage Average -distance_definition greater -output_pymol 100 1 10 20 10 dendrogram.py -pymol_label_output_small_molecule /home/user/molecules.sdf -pymol_set_min_max_girth 0.0 1.2
```



pymol_scale_node_with_size - the radius of the cylinder representing a cluster is increases as the number of members in the clusters increases

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 50 5 50 25 10 dendrogram_a.py -pymol_label_output_protein_model_from_string /home/user/pdbs/model _final.pdb -pymol_set_min_max_girth 0.0 13.0 -pymol_scale_node_with_size
```

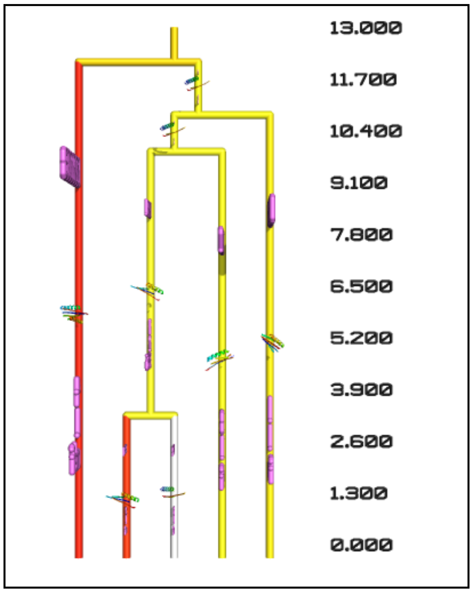


pymol_color_nodes_by_description - Colors clusters in a gradient based on some numerical descriptor. Each cluster is colored according to the average of the numerical descriptors for all its members. The gradient goes from Red (small average numerical descriptor) to White (large average numerical descriptor). The numerical descriptor could be some score, or RMSD to a native structure, etc. An example numerical descriptor file is given below. The member names in the descriptor file must match the member names in the distance input file.

```
1000_0000 16.4436

1000_0001 10.4263

1000_0002 17.3385

1000_0003 9.7106

1000_0004 24.9397
```

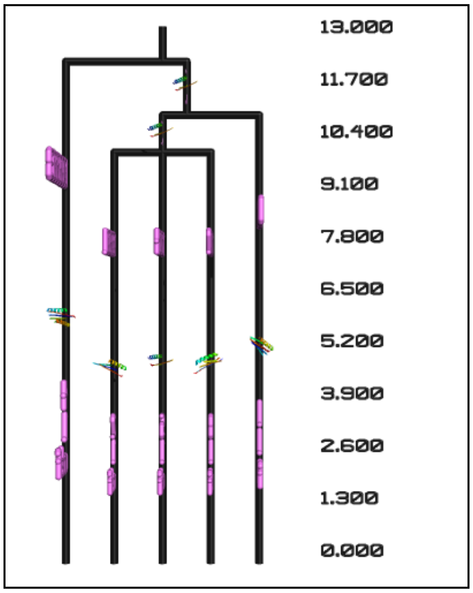
The minimum and maximum descriptor values for red and white, respectively, are set so that extreme values will not affect the gradient.

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows Centers -output_file cluster_output -output_pymol 50 5 50 25 10 dendrogram_a.py -pymol_label_output_protein_model_from_string /home/user/pdbs/model _final.pdb -pymol_set_min_max_girth 0.0 13.0 -pymol_color_nodes_by_description descriptions.ls 9 25
```



precluster – before clustering begins, makes a single pass through all objects and uses single linkage to populate clusters. So, during the single pass through all objects, two objects will be combined into a cluster if they have a distance that meets the provided cutoff. Additional objects will be added to a cluster if the distance from the current object to any of the objects already in the cluster meets the cutoff threshold. This is used to speed up clustering since having prepopulated clusters will reduce the number of iterations needed to develop the whole hierarchy. However, if the threshold is too generous the results in the important parts of the dendrogram could be affected.

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows
Centers -output_file cluster_output -output_pymol 50 5 50 25 10 dendrogram_a.py -pymol_label_output_protein_model_from_string
/home/user/pdbs/model _final.pdb -pymol_set_min_max_girth 0.0 13.0 -precluster 5
```



Here objects are added to a cluster if they have a distance of less than 5. Three objects are combined in the preclustering step leading to the cluster with three objects. The clusters further up the hierarchy are unaffected.

output_node_members – prints a file for every cluster. Each file lists the objects that are contained within that cluster. The files are named "dendrogram_node_*X*.ls", where *X* is a cluster identifier. The clusters are taken from the dendrogram after it has been filtered by remove_internally_similar_nodes and remove_nodes_below_size.

```
cluster.exe Cluster -distance_input_file distances.txt -input_format TableLowerTriangle -linkage Average -output_format Rows
Centers -output_file cluster_results.txt -output_node_members
```

The 9 output files are shown below with the filenames.

dendrogram_node_1.ls 1000_0000	**dendrogram_node_2.ls** 1000_0001	**dendrogram_node_3.ls** 1000_0002
dendrogram_node_4.ls 1000_0003	**dendrogram_node_5.ls** 1000_0004	**dendrogram_node_7.ls** 1000_0001 1000_0004
dendrogram_node_8.ls 1000_0001 1000_0004 1000_0002	**dendrogram_node_8.ls** 1000_0001 1000_0004 1000_0002 1000_0000	

No file is created for the cluster that contains everything (cluster 6), since it just contains all objects.

Not all clusters are necessarily printed to a file. The number of clusters that are output is controlled by the output_pymol max_number_node_labels parameter. So to increase or decrease the number of files that are created (i.e. clusters that are output to a file), change this number. When the output_node_members flag is used, the Pymol python script is created whether or not the output_pymol flag is used, so setting the output_pymol flag just to change the max_number_of_node_labels does not matter.