# Retsulc - Get clusters!

A clustering/clustering analysis program. The program recieves as input several types of '.csv' files containing, amongst other things, coordinates (2d or 3d). The coordinates are used to find clutsters and output an entire analysis based on that clustering.

# Main Window


Figure 1: Retsulc - Main Window

# Manual

## Procuring Samples

TBD

## User Input

The program recieves from the user the following paramters: 'Epsilon', 'Minimum Neighbors', 'Mini Epsilon', Mini Minimum Neighbors', 'Data Type' and (upon selecting 'GO!') a folder.

### Data Types

The program can deal with several types of information:

- 2d
- 3d
- raw_2d
- raw_3d
- old (hidden)

**2d** This data type is used on Vutara generated files. The program expects to find "pairs" of documents. A pair consists of a "green" file and a "red" file, carrying the same name (the same sample). Each of the files in the pair contains the coordinates of a specific protein.

The '2d' causes all points in the program to have (x,y,0) coordinates.

**3d** This data type is used on Vutara generated files. The program expects to find "pairs" of documents. A pair consists of a "green" file and a "red" file, carrying the same name (the same sample). Each of the files in the pair contains the coordinates of a specific protein.

The '3d' causes all points in the program to have (x,y,z) coordinates, according to the '.csv' file.

**raw_2d** This data type is used on "old system" generated files. The program expects to find a single document that contains the coordinates of two proteins, distinguished by a specific column in the file TBD.

The '2d' causes all points in the program to have (x,y,0) coordinates.

**raw_3d** This data type is used on "old system" generated files. The program expects to find a single document that contains the coordinates of two proteins, distinguished by a specific column in the file TBD.

The '3d' causes all points in the program to have (x,y,z) coordinates, according to the '.csv' file.

**old (hidden)** This data type is used on "very old" files. The files contain information regarding a single protein. The test is partial and ment for validation of results comared to "very old" results. It can be used by specifically writing 'old' in the 'Data Type' filed.

### Assumptions

These are assumptions made by the program about the input:

- Files carrying the same name (except 'color') are a pair.
- The sample dimensions are $20\mu m \times 20\mu m$.
- If '3d'- A points with Z values higher than '500' and less than '-500' isn't a valid point and is ignored

## File locating

The program looks inside the given folder (and subfolders) for 'pairs' of 'green' and 'red' files (explained above). For each pair the program performes the following.

## Get to work

An instance of a Sample object is created that will contain some Point objects and Cluster objects that in turn also contain Point obects. Each file in the pair is processed; Point object is created for each valid point in the data set and assigned the appropriate color and position. After this stage the Sample object contains lists of points, red poits, green points.

The points are arranged and fitted for the DBSCAN.

### DBSCAN

The program uses DBSCAN as the clustering algorithm. The algorihtm requires two parameters: 'epsilon' and 'minimum neighbors' which were given by the user[1]. A DBSCAN is performed on each color separatly and on the entire data set (no distinction between green/red). The DBSCAN returns a list of integers in the length of the relevant points' set, each integer correspond to a point in the same index in the points' list and constitute the number of the cluster that the points belongs to or '-1' if no cluster is assigned.

### Build clusters

The program "builds" green clusters, red clusters (and the non distinctive clusters) according to the explanation above. After this stage the Sample object contains three (3) clusters lists (red, green, non-distinctive), where as each cluster contains it's relevant points and color and each point is linked back to it's cluster.

### PCA (1st)

For each cluster the program calculates center of mass, PCA (principal components analysis)

[1] Very small epsilons may cause problems

### 'Pre' results

In this stage the program saves all the information about the single-colored clusters and some general points information (% of clustered points, etc.), all files in the output folder containing the suffix '_pre' are based upon this stage.

### Colocalization

Next the program defines a circular area around each cluster with the large diameter (PCA analysis) as the radius and appends opposite-colored points (green clulsters will get red points...) to the cluster. The program than cleans outliers according to 2.5 $\sigma$ from the center of mass. The program verifies 'true appendage' by re-running the DBSCAN wth the 'mini' parameters on the 'opposite-color- points in each cluster, if the added points form a legit cluster they remain. Each cluster is inspected for change of identity (if now most of the points in a green cluster are red, than the cluster becomes red). Another outliers analysis to clean 'bad' points.

### PCA (2nd)

For each cluster the program calculates center of mass, PCA (principal components analysis). After the appendage the physical attributes of each cluster may change s.a. center of mass, size, etc.

### 'Final' results

The information is saved in this stage and all output files containing the suffix '_final' are based upon this point.

## Output

The program outputs a test folder (name&date) containing the following:

- Analyses folder (name&date) for each "pair" located, containing:
  - "clusters_pre.csv"

    A file containing all the clusters found during the 'pre' results phase (see: 'Pre' results). Each row in the file contains the data of a cluster; it's dominant color, numnber of points, number of green points, sphere score (0-1), angle x (the angle of the large diameter to the base vector [1,0,0]), angle y (the angle of the large doameter to the base vector [0,1,0]), size (in nm, see:size ), density (see:Density) and colocalized (see:Colocalized).

  - "clusters_final.csv"

    A file containing all the clusters at the final phase (after all the modifications, appendages, cleanings, etc.). Each row in the file contains

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | color | #points | #red points | #green points | sphere score | angle_x | angle_y | size | density | colocalized |
| 2 | red | 1247 | 1247 | 0 | 0.2790772407 | 173.5647428536 | 83.5643965136 | 331.6998943144 | 3.7594223615 | 0 |
| 3 | red | 27 | 27 | 0 | 0.5496044158 | 87.7425270644 | 2.2578192756 | 102.2690856015 | 0.2640094007 | 0 |
| 4 | red | 39 | 39 | 0 | 0.5541693218 | 4.1671828957 | 94.1675292357 | 110.3420420617 | 0.3534464223 | 0 |
| 5 | red | 14 | 14 | 0 | 0.8632315616 | 174.3005909242 | 84.3002445842 | 61.1834140732 | 0.2288201829 | 0 |
| 6 | red | 77 | 77 | 0 | 0.6361423728 | 71.2698504124 | 18.7304959276 | 184.7959689807 | 0.4166757556 | 0 |
| 7 | red | 862 | 862 | 0 | 0.9205396084 | 41.0273400375 | 48.9730063026 | 255.532997174 | 3.3733412496 | 0 |
| 8 | red | 59 | 59 | 0 | 0.5965161384 | 64.023258969 | 154.023605309 | 203.8553209327 | 0.2894209468 | 0 |
| 9 | red | 247 | 247 | 0 | 0.4668293818 | 97.3646685494 | 172.6363704707 | 254.0215851903 | 0.9723583128 | 0 |

Figure 2: cluster_pre.csv example

the data of a cluster; it's dominant color, numnber of points, number of green points, sphere score (0-1), angle x (the angle of the large diameter to the base vector [1,0,0]), angle y (the angle of the large doameter to the base vector [0,1,0]), size (in nm, see:size ), density (see:Density) and colocalized (see:Colocalized).

- "clusters_all.csv"

  A file containing all the clusters found by DBSCAN without distinction between the protein color. Each row in the file contains the data of a cluster; it's dominant color, numnber of points, number of green points, sphere score (0-1), angle x (the angle of the large diameter to the base vector [1,0,0]), angle y (the angle of the large doameter to the base vector [0,1,0]), size (in nm, see:size ), density (see:Density) and colocalized (see:Colocalized).

- "kdist_COLOR.png"

  For each COLOR ('green'/'red'/ and for the sake of this paragraph 'all') the program outputs a graph containing the k-distance for the 16th neighbor (no special reason). The 'all' kdist is for the entire set of legal points.

- "hist_COLOR_PHASE.png"

  For each COLOR ('green'/'red') and for each PHASE ('pre'/'final'/'ALL') the program outputs an histogram of the clusters in color COLOR and in phase PHASE. The height of each bar is the relative number of clusters with the same 'basket size'. The 'basket sizes' are selected automatically.

- "rainbow_PHASE.png"

  The program outputs three 'rainbow' pics comprising the entire collection of discovered clusters at phase PHASE ('pre'/'final'/'preALL'='all') of the program. Each cluster is colored randomly. The triangles ($\triangle$) are the 'red' protein and the circles ($\circ$) are the 'green' protein.

- "diameter_specific_appending.png"

  A picture showing all the appended points ('red') and the 'pre' clusters ('blue'). Notice that here it is not possible to see which point is which protein.

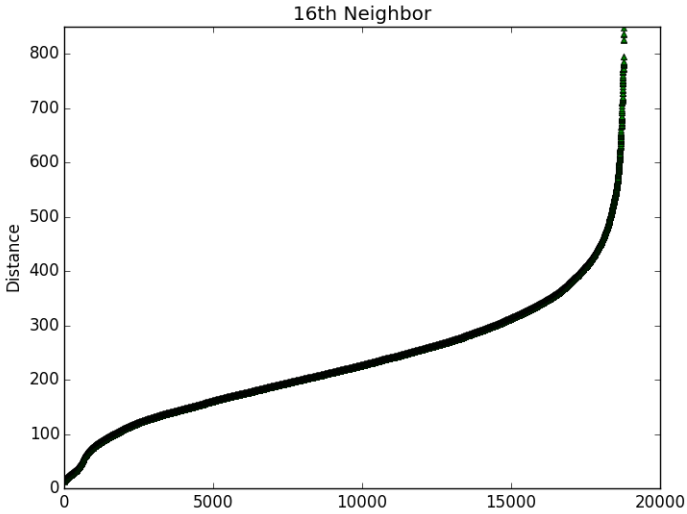



Figure 3: kdist example

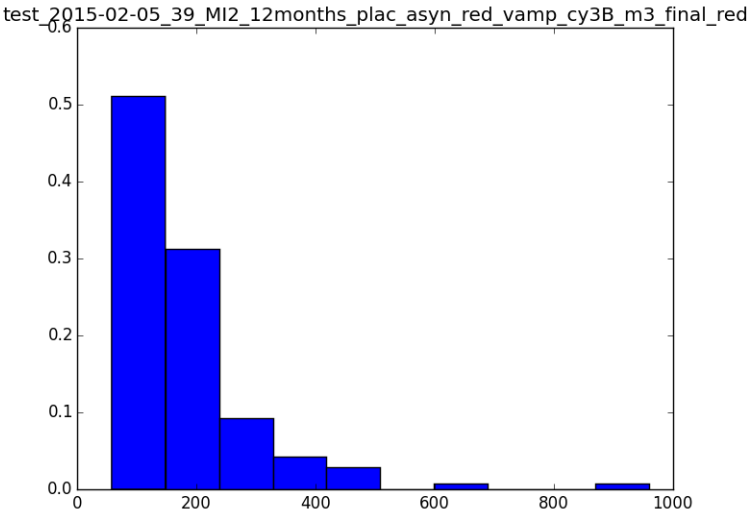



Figure 4: histogram example

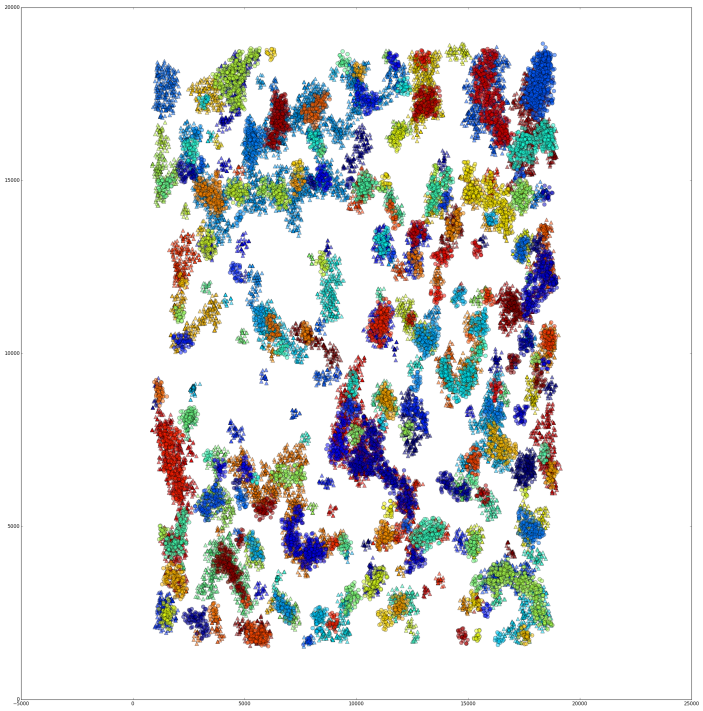



Figure 5: rainbow (pre) example

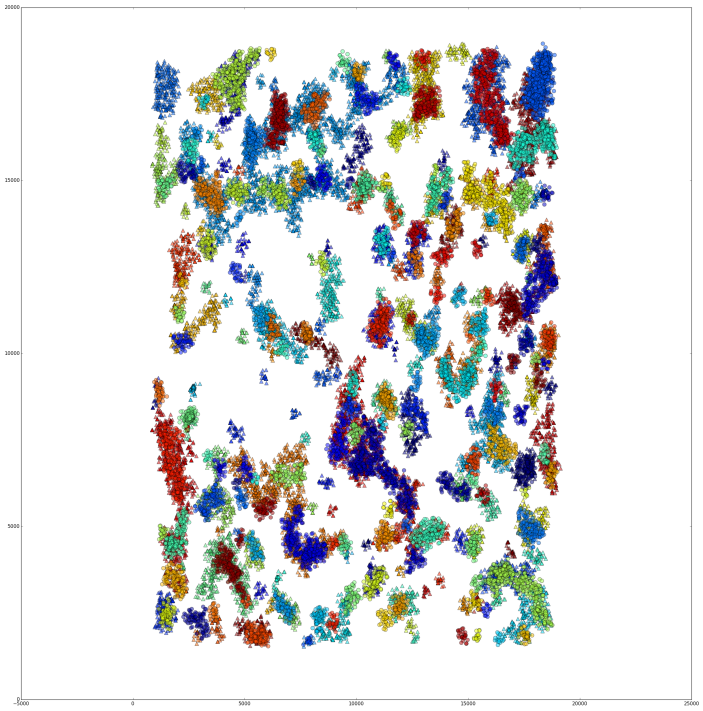



Figure 6: rainbow (final) example

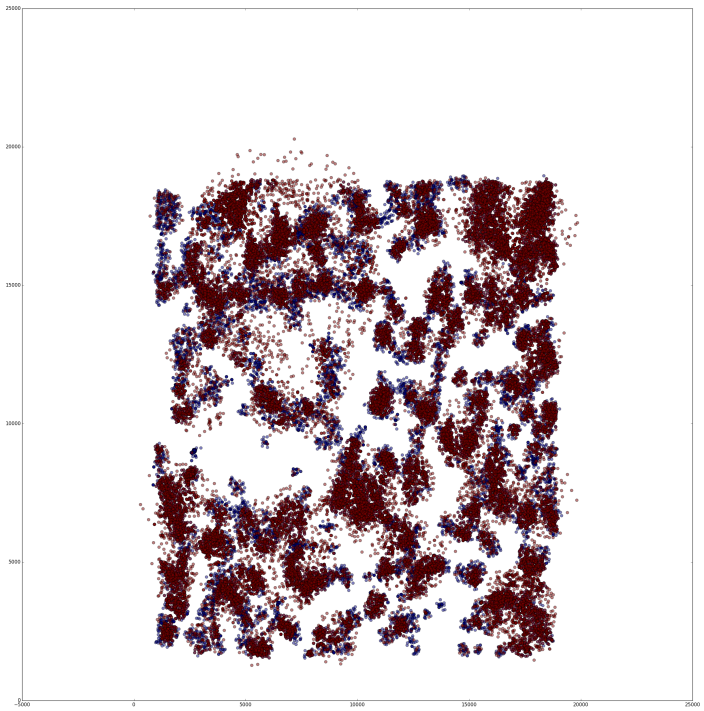



Figure 7: appendage example

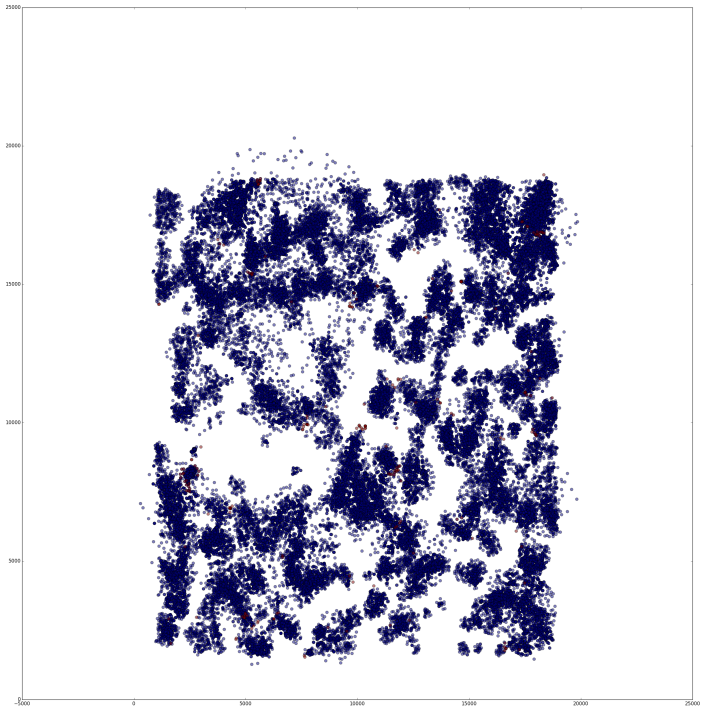


Figure 8: outliers example

  - "outliers.png"

    A picture showing all the points ('red') removed by the outlier analysis (see:Outliers Analysis) and the 'pre' clusters + appemded points ('blue'). Notice that here it is not possible to see which point is which protein.

  - "summary.txt"

    A text file containing some basic information about the sample. It is here due to historical reasons but is smetimes comfortable to verify correctness or just to look-up something.

- "pre" summary .csv file

  Each row in this file contains the averages and stds over all the sessions (m'1', 'm2', etc.) of the data relevant to the phase 'pre'.

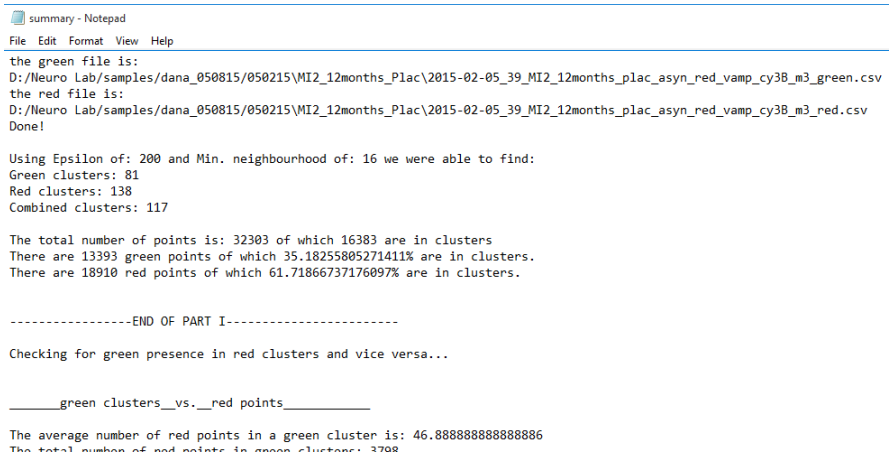



Figure 9: outliers example

| test# | total_number_of_points | total_number_of_red_points | total_number_of_green_points | total_number_of_clustered_points |
|---|---|---|---|---|
| 1 | 24866 | 9734 | 15132 | 290 |
| 2 | 27449 | 16193 | 11256 | 313 |
| 3 | 18298 | 6921 | 11377 | 189 |
| 4 | 21918 | 11232 | 10686 | 1220 |

| total_number_of_unclustered_points | total_number_of_clustered_red_points | total_number_of_clustered_green_points |
|---|---|---|
| 24576 | 221 | 69 |
| 27136 | 258 | 55 |
| 18109 | 62 | 127 |

| relative_clustered_points | relative_unclustered_points | relative_red_clustered_points | relative_green_clustered_points |
|---|---|---|---|
| 0.0116625111 | 0.9883374889 | 0.0227039244 | 0.0045598731 |
| 0.0114029655 | 0.9885970345 | 0.0159328105 | 0.0048862829 |
| 0.0103289977 | 0.9896710023 | 0.008958243 | 0.0111628725 |

| #clusters | #red clusters | #green clusters | avg green in red clusters | std green in red clusters | avg red in green clusters | std |
|---|---|---|---|---|---|---|
| 11 | 7 | 4 | 0 | 0 | 0 | |
| 9 | 7 | 2 | 0 | 0 | 0 | |
| 7 | 4 | 3 | 0 | 0 | 0 | |

| std red in green clusters | avg red sphericity | std red sphericity | avg green sphericity | std green sphericity |
|---|---|---|---|---|
| 0 | 0.844829347 | 0.1040439737 | 0.8799500289 | 0.1362754803 |
| 0 | 0.7572139635 | 0.1814742589 | 0.8908026883 | 0.075112897 |
| 0 | 0.8655342747 | 0.126365139 | 0.7228803247 | 0.1025960859 |

| avg red XangI | std red XangI | avg red YangI | std red YangI | avg green XangI | std green XangI | avg green YangI |
|---|---|---|---|---|---|---|
| 80.7812632859 | 35.882772901 | 116.4136756518 | 31.460122765 | 68.3099229521 | 35.8791399564 | 71.5299158094 |
| 77.8194730953 | 39.015501075 | 62.326713145 | 31.264253255 | 85.9013088739 | 46.9315179995 | 133.2384791086 |
| 91.3969903441 | 33.421719204 | 85.1554548963 | 52.306677591 | 87.6473257003 | 42.9927497068 | 97.7709566051 |

| avg green YangI | std green YangI | avg red size | std red size | avg green size | std green size | sample density |
|---|---|---|---|---|---|---|
| 71.5299158094 | 37.265042272 | 36.5595083793 | 9.4970964757 | 31.2208927333 | 4.8105103164 | 0.0002990797 |
| 133.2384791086 | 0.3764752062 | 39.4411415651 | 14.8606138405 | 37.7392771398 | 9.5084632438 | 0.0002761197 |
| 97.7709566051 | 45.881404912 | 29.7072145785 | 6.2525578876 | 36.7738383843 | 4.4449438535 | 0.0001799743 |

| avg red density | std red density | avg green density | std green density | red median size | green median size |
|---|---|---|---|---|---|
| 0.7831865037 | 0.4432721792 | 0.5227058878 | 0.3142528933 | 36.2405975722 | 30.9043874942 |
| 0.8090971584 | 0.4171000001 | 0.672125015 | 0.4489658802 | 36.4183769784 | 37.7392771398 |
| 0.5126131346 | 0.2073577892 | 1.166306662 | 0.3414666131 | 29.6569687456 | 36.5337928572 |

- "final" summary .csv file

  Each row in this file contains the averages and stds over all the sessions (m'1', 'm2', etc.) of the data relevant to the phase 'final'.

- "all" summary .csv file

  Each row in this file contains the averages and stds over all the sessions (m'1', 'm2', etc.) of the data relevant to the phase 'all'.

- "done" file

  A file (marker) letting me (and the program) know that the program has visited this folder and already analyzed it (altough those are probably historical reasons, please do not remove).

# Methods

- The program is written in Python 3.4
- Clustering algorithm - DBSCAN from 'scikit-learn'
- GUI - Tkinter
- Data manipulation & Statistics - Numpy
- PCA - matplotlib.mlab

### PCA

> Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables. https://en.wikipedia.org/wiki/Principal_component_analysis

The PCA used by the program is that of matplotlib.mlab library:

**class matplotlib.mlab.PCA(a, standardize=True)**

compute the SVD of a and store data for PCA. Use project to project the data onto a reduced set of dimensions

**Inputs:**

*a*: a *numobservations* × *numdims* array standardize: True if input data are to be standardized. If False, only centering will be carried out.

**Attrs:**

*a* a centered unit sigma version of input a

*numrows, numcols*: the dimensions of a

*mu* : a numdims array of means of a. This is the vector that points to the origin of PCA space.

*sigma* : a numdims array of standard deviation of a

*fracs* : the proportion of variance of each of the principal components

*s* : the actual eigenvalues of the decomposition

*Wt* : the weight vector for projecting a numdims point or array into PCA space

*Y* : a projected into PCA space

The factor loadings are in the Wt factor, i.e., the factor loadings for the 1st principal component are given by Wt[0]. This row is also the 1st eigenvector.

http://matplotlib.org/api/mlab_api.html

The program uses the PCA results in order to obtain:

```
def pca_analysis(self, dim=3, sphere=True): # if sphere is True (default) also updates the sphere score of the cluster.
    if dim > 2:
        self.pca = PCA(np.array([x.point for x in self.points], dtype=np.float))
    else:
        self.pca = PCA(np.array([x.point for x in self.points], dtype=np.float), standardize=False)
    self.center = self.pca.mu
    v1 = self.pca.Wt[0]
    self.angle_x = angle(v1, [1,0,0])
    self.angle_y= angle(v1, [0,1,0])
    self.angle_z = angle(v1, [0,0,1])
    min_sig = float(self.pca.sigma[0])/self.pca.sigma[1] if self.pca.sigma[1] != 0 else np.nan
    if min_sig != np.nan and min_sig > 1:
        min_sig = 1./min_sig
    if sphere:
        self.shape_2d = min_sig
    self.large_diameter = max(self.pca.sigma[0],self.pca.sigma[1])*2
    self.size = math.sqrt(self.pca.sigma[0]*self.pca.sigma[1])
```

**center of mass** Namely *mu*

**self angles** The angles with the axis are calculated by taking the angle between the weights vector (Wt) and the axis's base vector ([1,0,0]/[0,1,0]/[0,0,1]).

The angle is calculated by: $\arccos(v1 \bullet v2) \times 57.296$ (back to angles instead of radians).

**sphericity** The sphericity is calculated by TBD

**size** The size is calculated by taking the first two (2) *sigma* values $(\sigma_1, \sigma_2)$ from the PCA which represent the distance (from the center of a cluster) of the directions with the most variance (most distance). The size is the mean (geometric mean) radius of the cluster: $size = \frac{\sqrt{2\sigma_1 \times 2\sigma_2}}{2} = \frac{2\sqrt{\sigma_1 \times \sigma_2}}{2} = \sqrt{\sigma_1 \times \sigma_2}$

**Other**

**Density** The density calculated of a cluster is the number of points of the cluster divided by the cluster's size. As such this estimator may recieve a value of more than 1 and is not a "real" density.

**Colocalized** This attribute of a cluster is calculated after the 'pre' part (see:'Pre' results). The program appends "opposite-colored" (e.g. "red") points to a "colored" cluster (e.g. "green") by searching new points in the area around the cluster's center of mass and with a radius of the cluster's large diameter found in the 1st PCA phase. After the appending the program performes DBSCAN on the "opposite-colored" points of the cluster with the "mini" parameters in order to verify "true" colocalization. If at leat one cluster is found in this scan that contains at least 10 points than the cluster is considered "colocalized" and recieves the calue '1' (otherwise '0').

**Outliers Analysis** This analysis is performed for each cluster by calculating the average distance ($\mu$) of a point in the cluster to the cluster's center of mass and the standard deviation ($\sigma$) of that distance. Every point found more distant than $2.5\sigma + \mu$ from the center of the cluster is removed.

# Retlif - Filter clusters!

Retlif is a complementary tool for Retsulc. It provides filtering abilities for the clusters found in Retsulc.

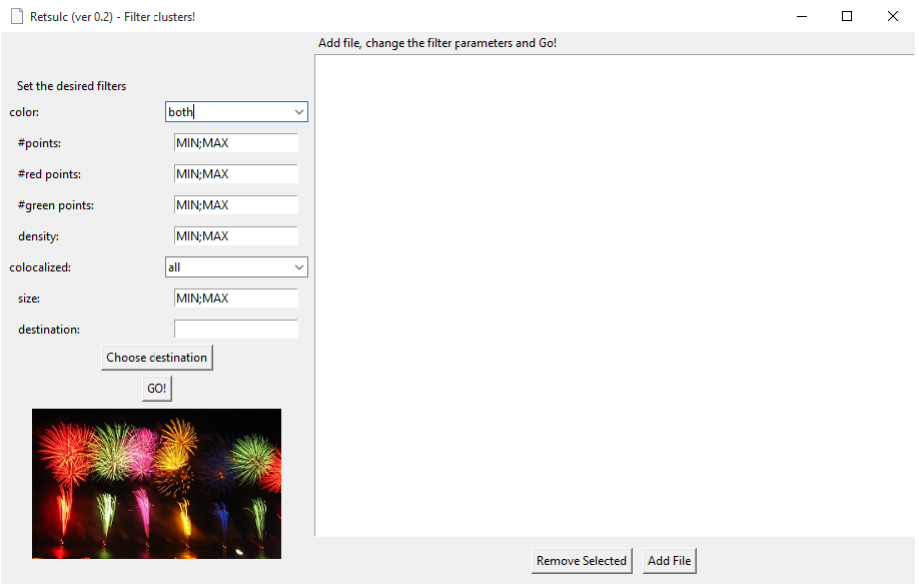



Figure 11: Retlif - Main Window

## Main Window

In the main window the user can define filters. The program currently supports the following filters:

- color (green/red/both)

  filter the clusters according to their color (the dominant protein)

- #points

  filter the cluster according to the number of points in each cluster(size in terms of points)

- #red points

  filter the cluster according to the number of red points in each cluster

- #green points

  filter the cluster according to the number of green points in each cluster

- density

  filter the clusters according to the each cluster's density

- colocalized (yes/no/all)

  choose to filter non/colocalized clusters (reminder: colocalized means that the cluster of X color has a sub-cluster of Y color, s.t. X isn't Y)

- size

  filter by cluster size (reminder: size is defined as the average radius of the cluster)

- destination

  choose where to save the filtered analysis

- GO!

  performes the analysis (same as in Retsulc) on the chosen, filtered (according to the user's wishes), files and writes the results to a file called "filtered_summary*time&date*.csv" in the chosen destination folder.

- Add File

  opens a file browser to add another file to the analysis. The chosen file will be added to the list-box (the white big frame)

- Remove Selected

  removes the highlighted file from the analysis. The file will be removed from the list-box (the white big frame)

# mega - save time!

filter & analyze the data with a click of a button! (well, almost ☻)

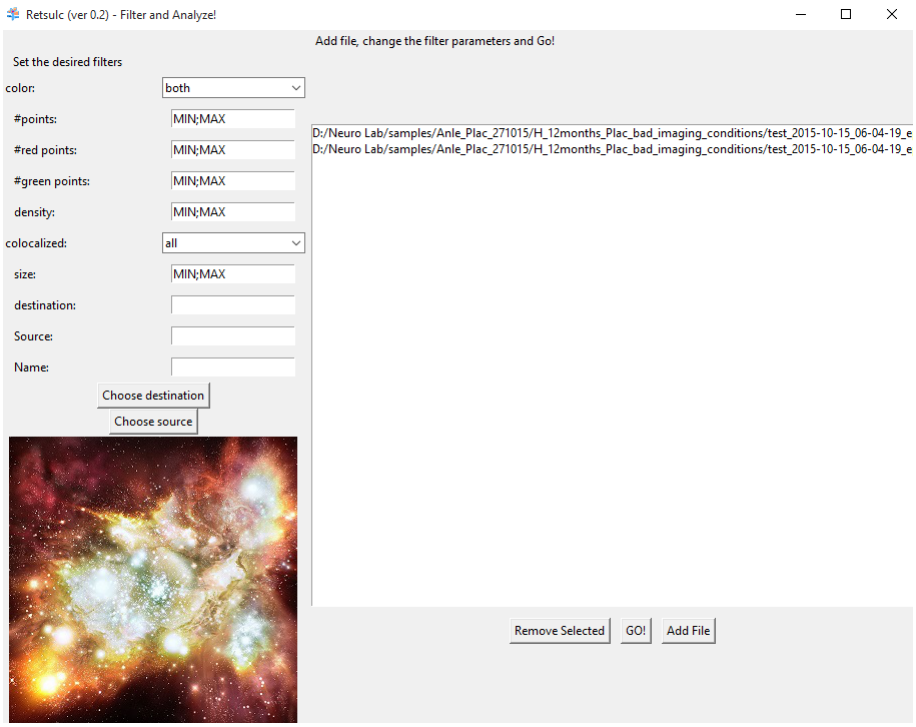
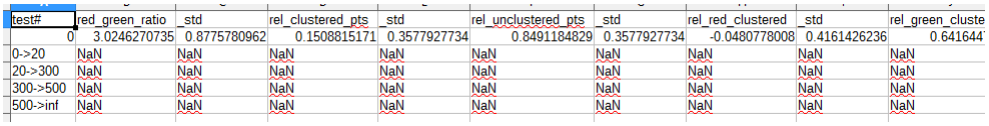



Figure 12: mega - Main Window

The mega tool is a shortcut. It enables us to filter & analyze real quick a lot of data. We noticed that once we have a good-enough idea of what to do wnd what to look for in all of the data we would do a lot of drudgery (e.g. sort all the clusters into four fixed sized beans, count all the XL clusters points and add them to count C..., etc.). the mega tool is very similar to the filter tool. It only needs another argument; 'source', which is a pre calculated sample's summary in a certain phase (pre/final). Given the 'source' file it is then possible to calculate the entire sample analysis. It wasn't possible in the filter tool since it didn't get basic data about the sample s.a. number of points. The mega tool outputs a special summary, giving out specific information and in general a specific point of view we wanted about the data. The output is two files; summary and a clusters files. The clusters file is identical to the one outputed by the filter tool (...containes the clusters that passed the filtration). A summary file example is presented:



| test# | red_green_ratio | _std | rel_clustered_pts | _std | rel_unclustered_pts | _std | rel_red_clustered | _std | rel_green_cluste |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 3.0246270735 | 0.8775780962 | 0.1508815171 | 0.3577927734 | 0.8491184829 | 0.3577927734 | -0.0480778008 | 0.4161426236 | 0.641644 |
| 0->20 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 20->300 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 300->500 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 500->inf | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

Figure 13: mega - output example

# Appendix A - Keep up with decisions

> It is important to keep up with the decisions we have made along the way (mainly so we won't forget why we chose this & that).

This appendix will list the specific parameters we have chosen and the rationalization beyond them (if any).

## DBSCAN parameters

The DBSCAN algorithm uses two key parameters: 'Epsilon' and 'Minimum neighbors'.

### Epsilon

The Epsilon parameter is used by the DBSCAN algorithm to define a distance or rather a radius for each point in the data set that constitute it's search radius. Within distance epsilon from each points DBSCAN looks for neighbors and counts them. We chose to use epsilon value of 200 nanometer. We chose this value for various reasons:

- k-nearest neighbor disribution

  By graphing the kdist graphs we noticed that athough the proteins differ in their breaking point (the 'knee' -like shape in the graph) it's around 200-300 nanometer in a typical sample.

- biologicaly

  The 200 value still counts as a reasonable size of oligomres.

- experience

  We have tested a lot of different values and the results looks definitly closer to what is possible to see from the microscope's pictures.

- better safe than sorry

  It occured to us that it is much easier to deal with false possitive clusters (i.e. filtration) than dealing with no clusters.

### Minimum Neighbors (Mn)

The Mn parameter is a threshold during the nighbor-looking and counting phase in the clustering algorithm. This parameter defines a central point in a cluster and may be viewed as a resriction on the size of the found clusters (no less than Mn). We chose to use the Mn value of 16 neighbors. Why? becuse:

- Makes sense biologically

  We expect that ....TBD

- ..TBD

- ..TBD

## Filtration parameters

Our analyzes are based on a filtered data set.

### Density

We have chosen to 'count' as clusters in our analysis only clusters that have density greater than 0.0098.

We noticed that sometimes large and homogenically spread protein forms a giant cluster which strongly seems as an artifact of the test and looks unlikley in a biological pov (point of view). We started filtering out data by size of the clusters but we felt uncomfortable discriminating only upon size. So, we decided to filer by density. The estimator for the filtration density value was chosen as to be $1.5\sigma$ less than the average density of the clusters from the population of 20-300 (size).

# Appendix B - some results

Let's take for example a pair of samples taken in 05/02/15.

The samples consists of:

1. MI2_12month_Anle

   Mice at the age of 12 months that have a truncated alpha-synuclein (aggresive symptoms) that were treated three months (6 months - 9 months of age) with ana anti-aggregate drug. The smaple consists of 10 pairs of csv files (red&green or: alpha-synuclein/vamp).

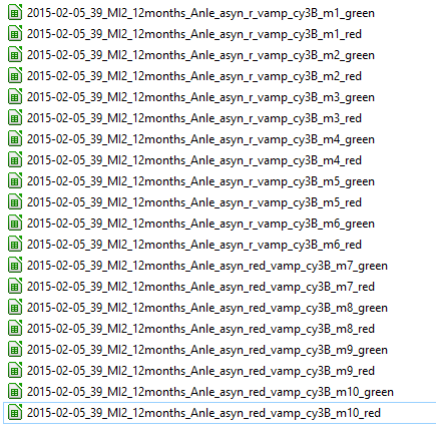



Figure 14: csvs input - 050215 anle sample

2. MI2_12months_Plac

   Same mice only that these received a placebo (and not the anti-aggregate). This sample also consists of 10 pairs of csv files (although it doesn't have to be the same number as the other condition).

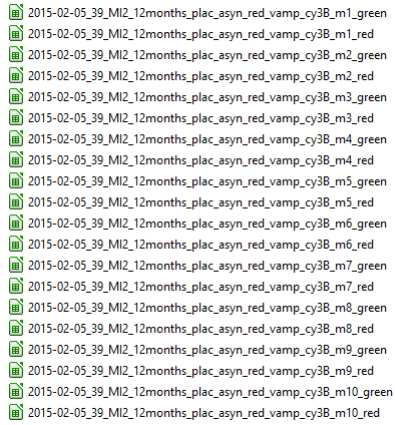



Figure 15: csvs input - 050215 plac sample

Foreach of these samples the program outputs a folder containing the results:

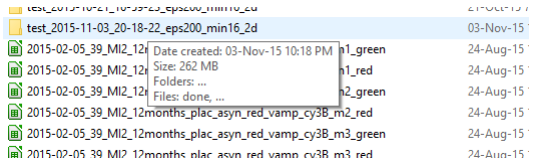



Figure 16: output - main output folder (marked)

Inside the folder there is an analysis for each pair of input CSV files.

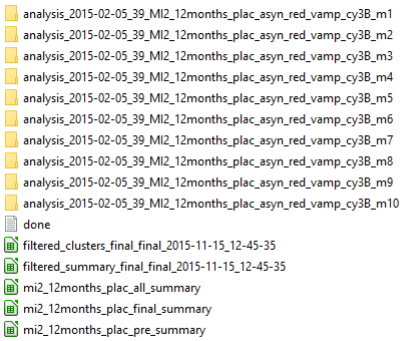



Figure 17: output - inside the: main output folder (marked)

Each 'analysis' folder looks like:

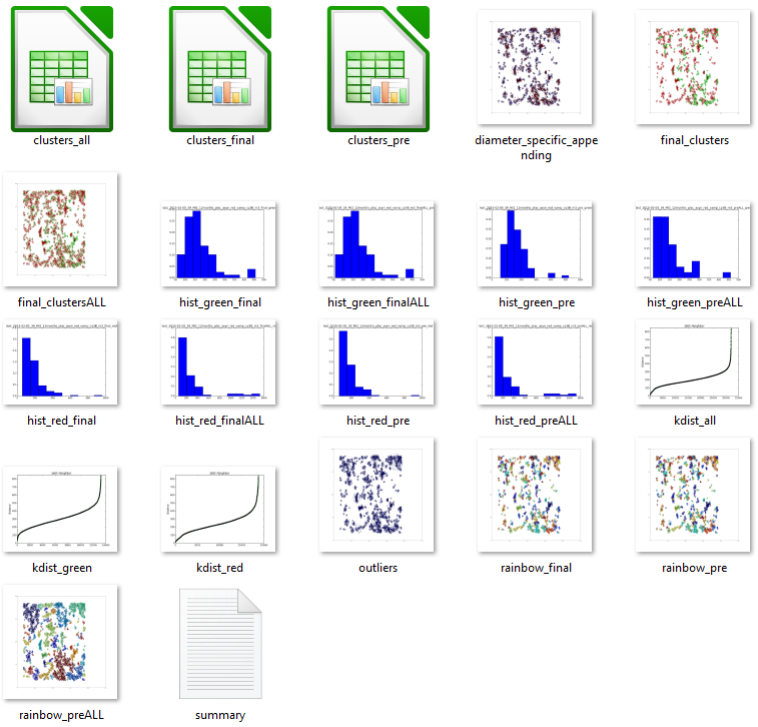



Figure 18: output - inside the: main output folder (marked)