

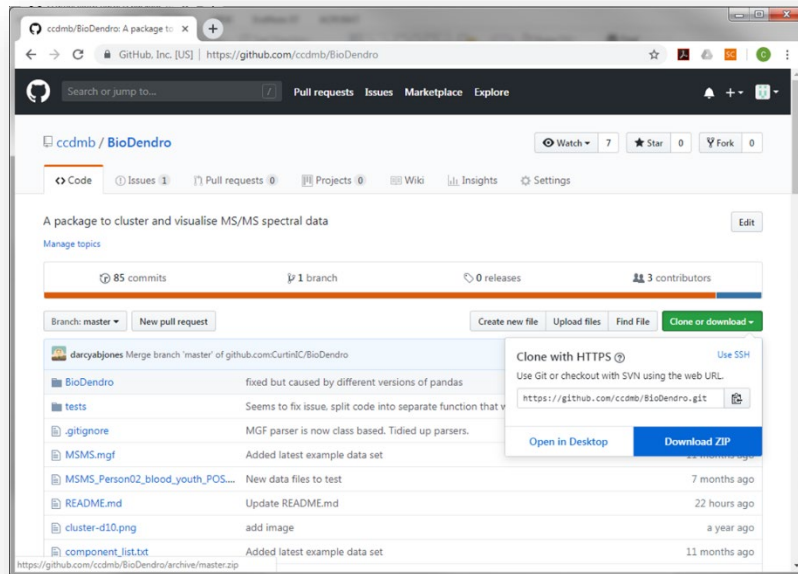
## **Download, install and use of BioDendro for Windows 10**

BioDendro is a Python package conveniently run through Jupyter Notebook and freely downloadable from Github.

Jupyter Notebook is an open source web platform that allows users to create and execute applications. Installation of Python and Jupyter Notebook is implemented through Anaconda, an open source distribution manager of Python and R programming languages.

### **1. Downloading BioDendro from Github**

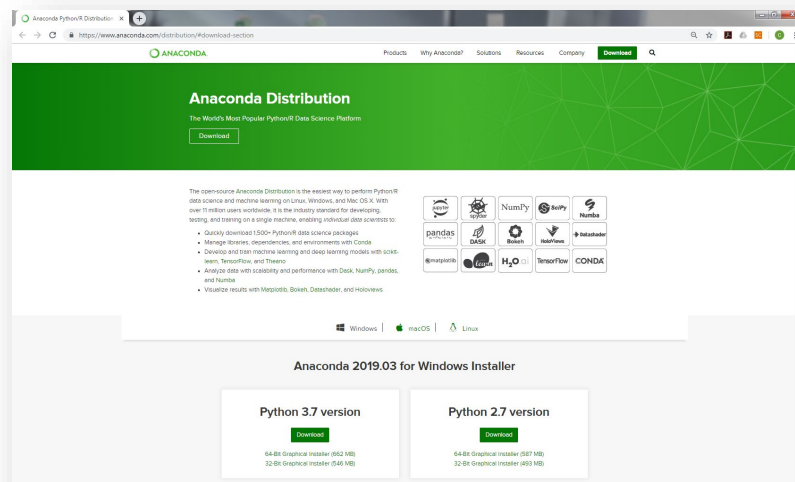
- 1) The .zip file can be downloaded from <https://github.com/ccdmdb/BioDendro> and unpacked in to user directory of choice:



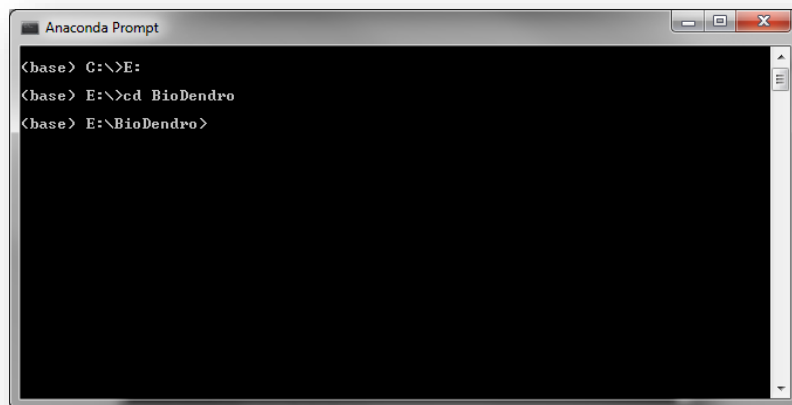
### **2. Installing BioDendro using Anaconda.**

Anaconda is a web based application used to download and install Python and Jupyter Notebook.

- 1) We recommend downloading Python 3.7 version for Windows from <https://www.anaconda.com/distribution/#download-section> (page defaults to macOS installer).



Once installed, open Anaconda Prompt and change directory to reflect where BioDendro was unpacked to:

A screenshot of the Anaconda Prompt window. The title bar reads "Anaconda Prompt". The command prompt shows the following sequence of commands: first, the user is at the root of the C: drive (C:\>E:); then, they navigate to the BioDendro directory (E:\>cd BioDendro); and finally, they are in the BioDendro directory (E:\BioDendro>).

```
<base> C:\>E:
<base> E:\>cd BioDendro
<base> E:\BioDendro>
```

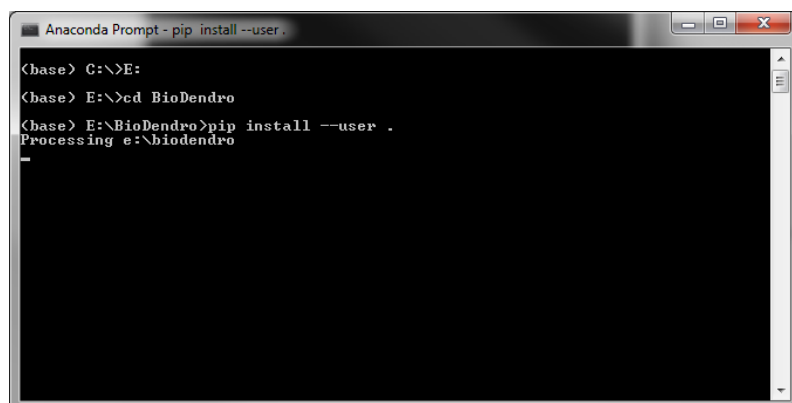
To install the BioDendro pipeline use:

`"pip install --user ."`

or

`"pip3 install -user ." (if multiple Python versions installed)`

and press enter. The "." signifies the installer will look to the current loaded directory for a package to install:

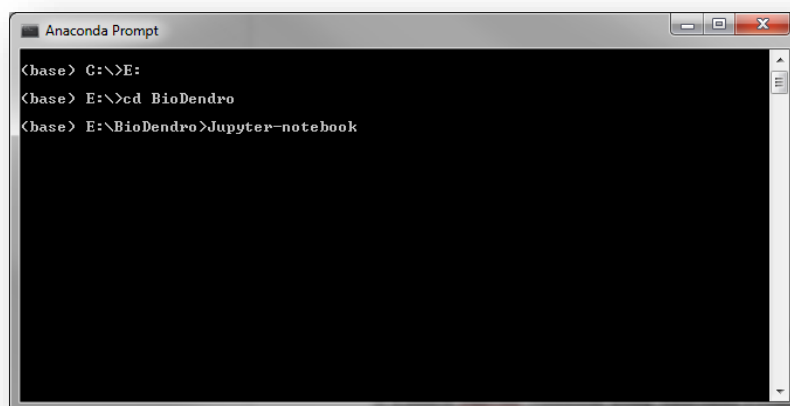
A screenshot of the Anaconda Prompt window with the title "Anaconda Prompt - pip install --user .". The command prompt shows the user navigating to the BioDendro directory and then typing the installation command. The prompt "Processing e:\biodendro" is visible, indicating the installation process has begun.

```
<base> C:\>E:
<base> E:\>cd BioDendro
<base> E:\BioDendro>pip install --user .
Processing e:\biodendro
```

Upon completion of processing, BioDendro is now installed and ready for use through Jupyter notebook.

### **3. Launch Jupyter Notebook using Anaconda Prompt**

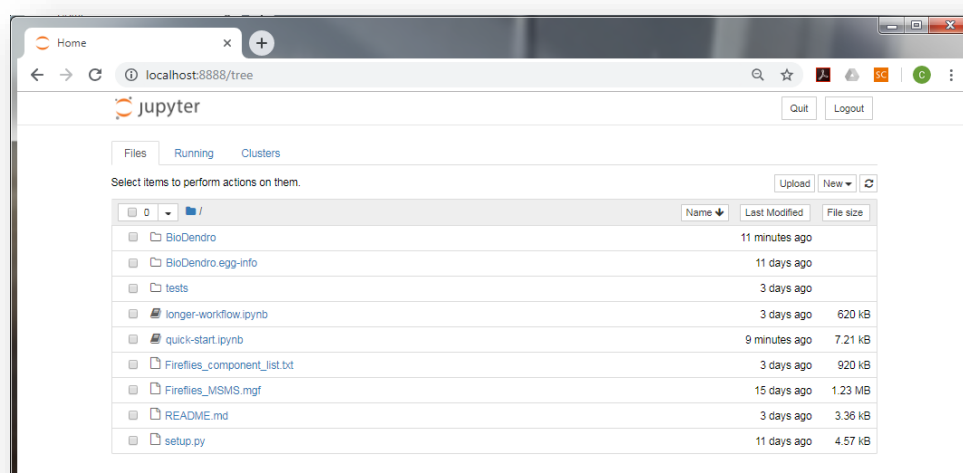
1) The Jupyter notebook for BioDendro can now be launched from Anaconda Prompt, each time you wish to use it. Launch BioDendro from the location from which it is installed:



```
<base> C:\>E:
<base> E:\>cd BioDendro
<base> E:\BioDendro>Jupyter-notebook
```

We recommend you place the required .mgf and .txt file (see section 4 for required files and formats) to this directory on your computer for ease of use.

2) Upon launching the following window will open:



#### **4. Minimum requirements for BioDendro**

BioDendro aligns MS/MS spectra to a list of features derived from metabolomics profiling data. BioDendro requires 2 data inputs: i) Feature list in .txt format and ii) a .mgf file of raw MS/MS spectra.

i) Feature list as a text file

The feature list is representative of the analytes observed in the sample set. This list is derived prior to analysis with BioDendro from software of the users own preferences. Free online tools such as XCMS<sup>11</sup>, MetAlign<sup>12</sup>, MZmine2<sup>13</sup> or instrument vendors proprietary software can generate this list. The list will represent the features that BioDendro will attempt to align MS/MS spectra too and should be built to users own requirements (e.g minimum S/N or filtering out adducts and products of in-source fragmentation).

The example, Fireflies\_feature\_list.txt, is produced as per the original publication using MZmine2. Detailed instructions for generation of sample analytes are provided in Fallon et al.<sup>8</sup> in section 4.6 of the supplementary information of the related article. BioDendro accepts the feature list in the following format:

SampleID\_userinfo\_userinfo\_m/z\_RT (in minutes)  
e.g. Sample1\_pos\_C18\_123.1234\_5.60

ii) Raw MS/MS data in MGF

MS/MS data can be extracted from raw data files or various converted formats through ProteoWizard MSConvert GUI, available free from <http://proteowizard.sourceforge.net/>. We supply Fireflies\_MSMS.mgf file extracted from Ppyr\_hemolymph\_extract.mzML, in GitHub as an example.

## 5. Running BioDendro

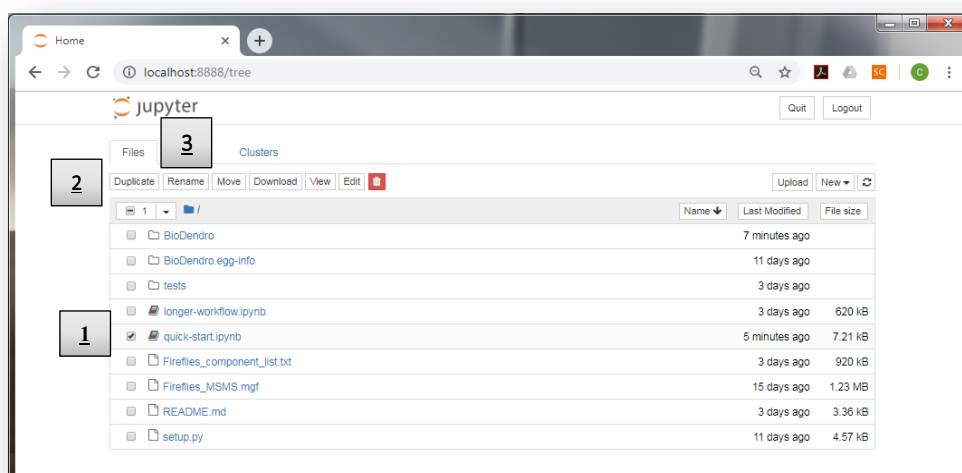
There are two options to run BioDendro.

quick-start.ipynb	This notebook allows users to enter their analysis parameters at a single step and execute the entire pipeline.
longer-workflow.ipynb	This notebook executes each step in the pipeline in a step-wise manner. Recommended for advanced users who wish to extend or manipulate the current pipeline.
quick-start-example.ipynb	This notebook contains the example data set with user parameters and analysis.

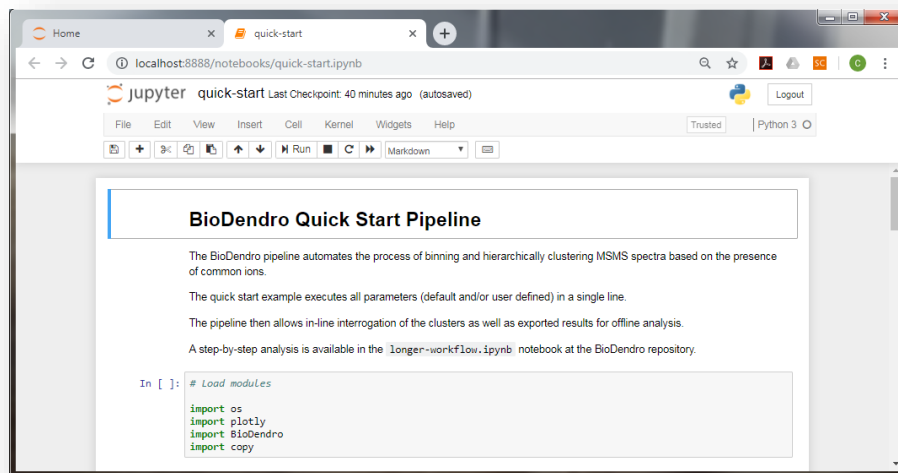
The following instructions will be demonstrated using “quick-start-example.ipynb” with the Firefly data from Metabolights repository (MTBLS698) and as discussed in the BioDendro article.

We do recommend creating a notebook for each analysis you do. This can be done by:

- 1) ticking the box to the left of intended notebook
- 2) Duplicating selected notebook and
- 3) Renaming.



2) Click your notebook to open the application:



## **6. Running BioDendro**

BioDendro runs from “cells” containing code. Each cell needs to be run to execute the command within that cell.

Each cell in the BioDendro pipeline can be run by selecting that cell and clicking “Run”, pressing “▶|” (Windows 10 only) button next to that cell or “shift+enter”.

The `In [ ] :` text next to each cell lets the user know the order in which the cell was run. After executing a cell, the `[ ]` will be populated with a number indicative of the order it ran in. During execution of the cell it will be populated as `[*]`, indicating a function is still running.

a) To use BioDendro, first load the modules operating it:

```
In [1]: # Load modules

import os
import plotly
import BioDendro
import copy
```

- b) Unless otherwise stated, BioDendro will run on default settings. To view the available parameters, use the help function. The default values are displayed. This step is optional:

```
To get a list of possible parameters and defaults you can run help(BioDendro.pipeline).

In [2]: help(BioDendro.pipeline)

Help on function pipeline in module BioDendro:

pipeline(mgf_path, components_path, neutral=False, cutoff=0.6, bin_threshold=0.0008, clustering_method='jaccard', processed='processed.xlsx', results_dir=None, out_html='simple_dendrogram.html', width=900, height=1200, quiet=False, scaling=False, filtering=False, eps=0.6, mz_tol=0.002, retention_tol=5, *kwargs)
    Runs the BioDendro pipeline.

    BioDendro takes an MGF and a list of components, finds MSMS spectra matching your components, and writes out useful summaries of clusters based on shared ion spectra

    Minimal example:
    >>> tree = BioDendro.pipeline("MSMS.mgf", "component_list.txt")
```

- c) The next cell runs the entire BioDendro pipeline. All user defined parameters are set here:

```
In [*]: # Run the complete BioDendro pipeline

tree = BioDendro.pipeline("Fireflies_MSMS.mgf", "Fireflies_component_list.txt", scaling=True, filtering=True)

Running BioDendro v0.0.1

- input mgf file = Fireflies_MSMS.mgf
- input components file = Fireflies_component_list.txt
- neutral = False
- cutoff = 0.6
- bin_threshold = 0.0008
- clustering_method = jaccard
- output processed file = results_20190429123228\processed.xlsx
- output results directory = results_20190429123228
- output html dendrogram = results_20190429123228\simple_dendrogram.html
- dendrogram figure width = 900
- dendrogram figure height = 1200
- scaling = True
- filtering = True
- eps = 0.005
- mz_tolerance = 0.002
- retention_tolerance = 5

Processing inputs
Binning and clustering
This may take some time...
Writing per-cluster summaries
```

The quick-start-example.ipynb has several of the default values changed and this cell is set as:

```
tree = BioDendro.pipeline("Fireflies_MSMS.mgf", "Fireflies_component_list.txt",
clustering_method="braycurtis", scaling=True, filtering=True, eps=0.001,
bin_threshold=0.004)
```

This means for `clustering_method`, `scaling/filtering/eps` and `bin_threshold`, we are not using default values.

The quick-start.ipynb is set as:

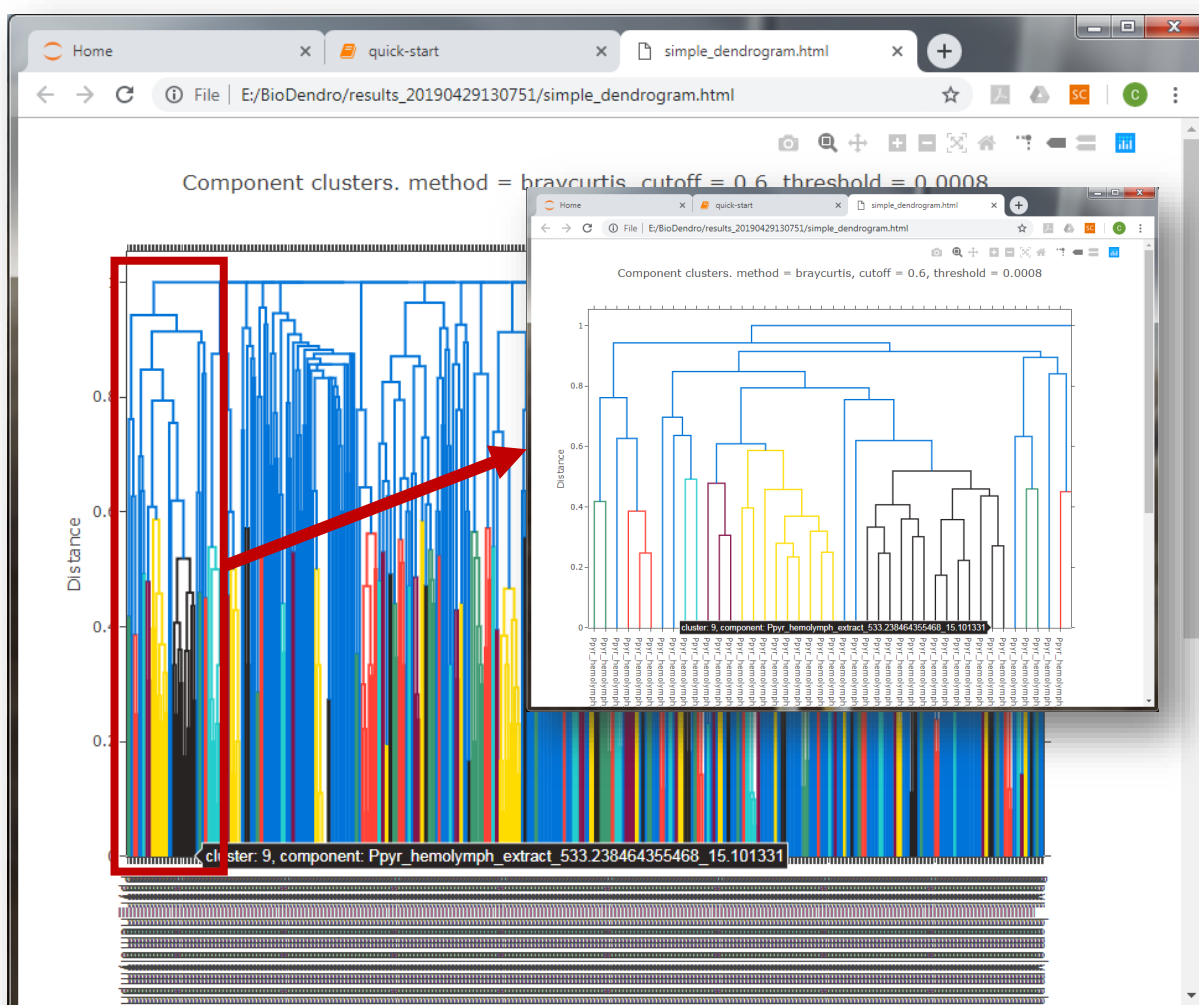
```
tree = BioDendro.pipeline("MGF_filename_here.mgf", "Component_filename_here.txt")
```

The MGF and component list file names are the minimum requirement to run BioDendro.

Note: if you do not place your MGF and component list files in the BioDendro directory, the file path to these files needs to be reflected in the filename. As an example:

```
tree = BioDendro.pipeline("E:/Data/Firefly_data/Fireflies_MSMS.mgf",
"E:/Data/Firefly_data/Fireflies_component_list.txt")
```

- d) At the completion of analysis an interactive dendrogram will pop up in a separate window, users can zoom, pan and search (using Windows search function, Ctrl+F) for analytes of interest. The hover key shows the analyte name and which cluster the branch belongs too.



The following files are also generated during analysis:

File name	Description
cluster_xx_yy.xlsx	A table for each cluster is generated and the ions present in each component is represented as True or False. Filename format: cluster_cluster#_# of components.xlsx
cluster_xx_yy.png	A histogram is produced for each cluster showing the frequency of each ion within that cluster. File format: cluster_cluster#_# of components.png

clusters.xlsx	A summary of all clustered analytes and their respective cluster number
params.txt	The parameters used for analysis
processed.xlsx	The component list and its aligned MS/MS spectra from the MGF file. MGF MS/MS identifier (concatenated from): "TITLE" "PEPMASS" "RTINSECONDS"
simple_dendrogram.html	The dendrogram from analysis.

Note: these files will be saved to the BioDendro root directory within the created `results<datetime>` folder.

## 7. Interrogating results

The computing stage is now complete and results can be interrogated offline. Alternatively, results can be viewed in-line, analysis can be saved/annotated and viewed in the future. All further cells are for in-line visualisation of results and are optional for the user, all in-line tables and images are output to `results<datetime>` as .xlsx and .png files.

- 1) Use the following cells to find and present the data for individual analytes.

- a) Locating the cluster an analyte of interest belongs too.

Required: Exact name of analyte as it appears in component list. Can paste in to cell.

```
tree.cluster_map["sample_id_here"]
```

Output: Cluster number analyte belongs too

```
In [9]: # return the cluster number for which the queried analyte belongs
tree.cluster_map["Ppyr_hemolymph_extract_533.238464355468_15.101331"]
```

```
Out[9]: 9
```

- b) Visualising, in-line, the table of components and ions for a chosen cluster.

Required: The cluster number you wish to interrogate

```
tree.cluster_table(cluster=cluster_number_here)
```

Output: table representing all components and ions



```
In [10]: # for visualising the ion table of your cluster of interest
tree.cluster_table(cluster=9)
```

```
Out[10]:
```

	bins	105.0701_105.0698_105.0707	107.0492_107.0489_107.0498	119.0859_119.085
	component			
Ppyr_hemolymph_extract_1082.49499511718_15.123002		True	True	
Ppyr_hemolymph_extract_1110.52667236328_17.431269		True	True	
Ppyr_hemolymph_extract_491.227569580078_12.962671		True	True	
Ppyr_hemolymph_extract_491.227676391601_10.204906		True	True	
Ppyr_hemolymph_extract_533.238464355468_15.101331		True	True	
Ppyr_hemolymph_extract_547.254150390625_17.431269		True	True	
Ppyr_hemolymph_extract_550.264587402343_15.123002		True	True	
Ppyr_hemolymph_extract_561.269592285156_19.535629		True	True	
Ppyr_hemolymph_extract_561.269592285156_19.784291		True	False	
Ppyr_hemolymph_extract_564.280609130859_17.431269		True	False	
Ppyr_hemolymph_extract_574.264526367187_15.101331		True	True	
Ppyr_hemolymph_extract_578.296142578125_19.784291		True	True	
Ppyr_hemolymph_extract_998.47412109375_12.962671		True	True	

13 rows × 136 columns

- c) Display the histogram for the ions within a specified cluster.

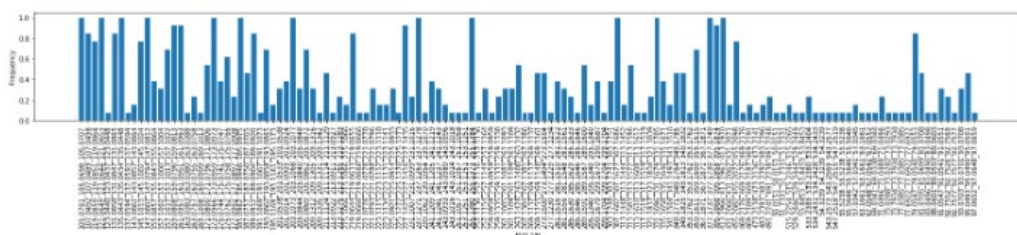
Required: The cluster number you wish to interrogate

```
%matplotlib inline
tree.cluster_hist(cluster=cluster_number_here)
```

Output: table representing all components and ions with selected cluster

```
In [11]: # for plotting the histogram of your cluster of interest
%matplotlib inline
tree.cluster_hist(cluster=9)
```

```
Out[11]: (<Figure size 1540.8x360 with 1 Axes>,
<matplotlib.axes._subplots.AxesSubplot at 0x14752828>)
```



- 2) New clusters can be generated if it is deemed the cutoff was inappropriate for the dataset.  
a) Apply a new cluster to the currently generated tree. New clusters are derived from already existing tree

Required: new dendrogram cut off (between 0 and 1)

```
new_tree.cut_tree(cutoff=new_cutoff_here)
```

Output: the number of clusters the new cutoff would generate

```
In [12]: # Show the number of clusters before adjustment
print("BEFORE: Cutoff:", tree.cutoff, "n clusters:", len(set(tree.clusters)))

# Re-set a new cutoff for clusters
new_tree=copy.deepcopy(tree)
new_tree.cut_tree(cutoff=0.8)

# Show number of clusters after adjustment
print("AFTER: Cutoff:", new_tree.cutoff, "n clusters:", len(set(new_tree.clusters)))

BEFORE: Cutoff: 0.6 n clusters: 273
AFTER: Cutoff: 0.8 n clusters: 166
```

- b) Write the new clusters (note: all clusters derived from the new cutoff are now associated with the object `new_tree`)

Required: New directory name, this is created when run

```
os.makedirs("new_dir_name_here", exist_ok=False)
new_tree.write_summaries(path="new_dir_name_here")
new_tree.plot(filename="new_dir_name_here/simple_dendrogram.html",
width=900, height=1200);
```

Output: The new .xlsx tables, .png histogram images and interactive dendrogram.

```
In [13]: # Generate the out plots and tables of the new clusters.
# rename a new directory.
# To write out the new tree.
os.makedirs("results-cutoff-08", exist_ok=False)
new_tree.write_summaries(path="results-cutoff-08")
new_tree.plot(filename="results-cutoff-08/simple_dendrogram.html", width=900, height=1200);
```

- c) View, in-line, the dendrograms with the old and new cutoffs

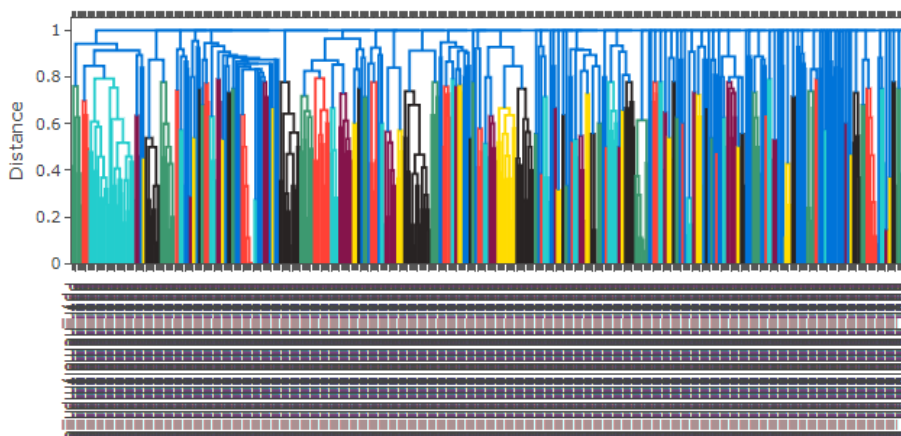
Required: This step is optional, it is purely for convenience of viewing both dendrograms next to each other. `ip1` is the original cut off and `ip2` is the new. Users can set the dendrogram size. The original dendrogram has not been shown in this image.

Output: In-line dendrograms of old and new cutoffs.

```
In [14]: # View the new dendrogram cutoff inline
# for visualising plot inline
plotly.offline.init_notebook_mode(connected=True)
iplot1 = tree.plot(width=800, height=900)
plotly.offline.iplot(iplot1)
```

```
In [20]: iplot2 = new_tree.plot(width=800, height=900)
plotly.offline.iplot(iplot2)
```

Component clusters. method = braycurtis, cutoff = 0.8, threshold = 0.0008



3) The clusters can be queried for location and tables with the new cutoff.

Required: As per step 1 a-c) in section 6, all commands are the same but results are derived from `new_tree`

Output:

```
In [21]: # return the cluster number for which the queried analyte belongs
new_tree.cluster_map["Ppyr_hemolymph_extract_533.238464355468_15.101331"]
```

```
Out[21]: 3
```

```
In [ ]: # for visualising the ion table of your cluster of interest
new_tree.cluster_table(cluster=3)
```

```
In [ ]: # for plotting the histogram of your cluster of interest
%matplotlib inline
new_tree.cluster_hist(cluster=3)
```

Notebook tips:

Keyboard shortcut	Description
Shift+Enter	Keyboard shortcut for running selected cell.
Tab	List of possible operations in cell. Useful in BioDendro pipeline cell for all available parameters.

The current pipeline allows you to search and visualise one analyte at a time. If you wish to keep a record of multiple analytes, then the appropriate cells can be copied and pasted in-line. Select the below cells:

```
Find analytes and their clusters below

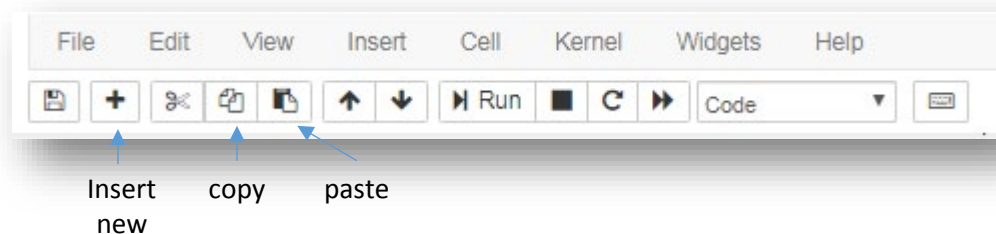
The results folder contains .csv and .png files of all clusters. This information can also be viewed in-line.

In [ ]: # return the cluster number for which the queried analyte belongs
tree.cluster_map["Ppyr_hemolymph_extract_533.238464355468_15.101331"]

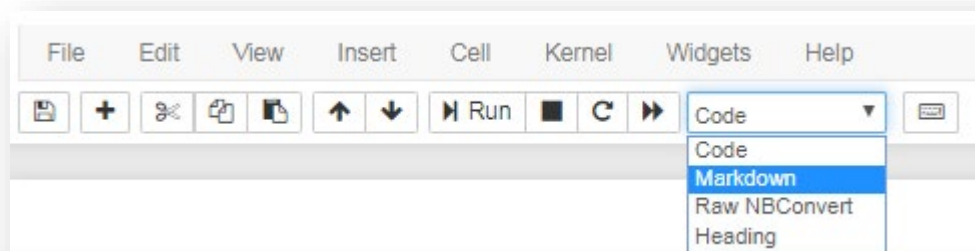
In [ ]: # for visualising the ion table of your cluster of interest
tree.cluster_table(cluster=9)

In [ ]: # for plotting the histogram of your cluster of interest
%matplotlib inline
tree.cluster_hist(cluster=9)
```

Copy and paste these cells where desired by using the functions at the top of the page:



Additional commentary can be added to the pipeline by inserting a markdown cell. The default cell type is code, insert a new cell and change it to markdown. Double click cell to start typing:



```
Find analytes and their clusters below

The results folder contains .csv and .png files of all clusters. This information can also be viewed in-line.

The following analyte is ....|

In [ ]: # return the cluster number for which the queried analyte belongs
tree.cluster_map["Ppyr_hemolymph_extract_533.238464355468_15.101331"]

In [ ]: # for visualising the ion table of your cluster of interest
tree.cluster_table(cluster=9)

In [ ]: # for plotting the histogram of your cluster of interest
%matplotlib inline
tree.cluster_hist(cluster=9)
```

Visit <https://www.markdownguide.org/> to find additional uses of markdown cells.