**Manual for *ClassifySAPDATA***

By Kosuke Hamaguchi

Mooney Lab, Duke University

2010/09/13

**0. Purpose**

Songbirds sing hundreds of bouts per day which usually includes more than 10,000 syllables per day. It is therefore needed to have a semi-automatic method to detect syllables and classify them, especially when one needs to detect subtle changes in the song. The purpose of this program (*ClassifySAPDATA*)is to classify detected syllables in semi-automatic fashion and upload syllable features into mysql database. The flow of the analysis is like below;

1. Record birdsongs (like using SAP)
2. Pre-process sound data to detect syllables (*KinkaSongAnalysis*)
3. **Classify syllables with this program (*ClassifySAPDATA*).**

Here I explain **3) Classify syllables** part.

In this manual, we assume that the table name in mysql database is ”WAVEDIR”, and also the wave file directory is “WAVEDIR” where \*.wav files (song data) are stored. The feature data is stored in “WAVEDIR\_feature”. C*lassifySAPDATA*  will try to find feature and sound directories based on the above rules, and if it cannot find them, it will ask you to locate these directories.

**1. Installation　(Windows)**

You will need

a) MATLAB

b) MATLAB database toolbox　(If you have bioinformatics toolbox, it contains a code, *svmtrain.m* and it conflicts with my *svmtrain.m* included in my code. Please remove the path to that toolbox.)

c) MySQL database ver.4 (usually comes with SAP (Sound Analysis Pro)).

SAP can be downloaded from Ofer Tchernichovski’s home page.

http://forum.sci.ccny.cuny.edu/Members/ofer/sound-analysis-pro/sound-analysis-pro-2

d) MySQL ODBC driver 3.51 (not ver.4); <http://dev.mysql.com/downloads/connector/odbc/3.51.html>

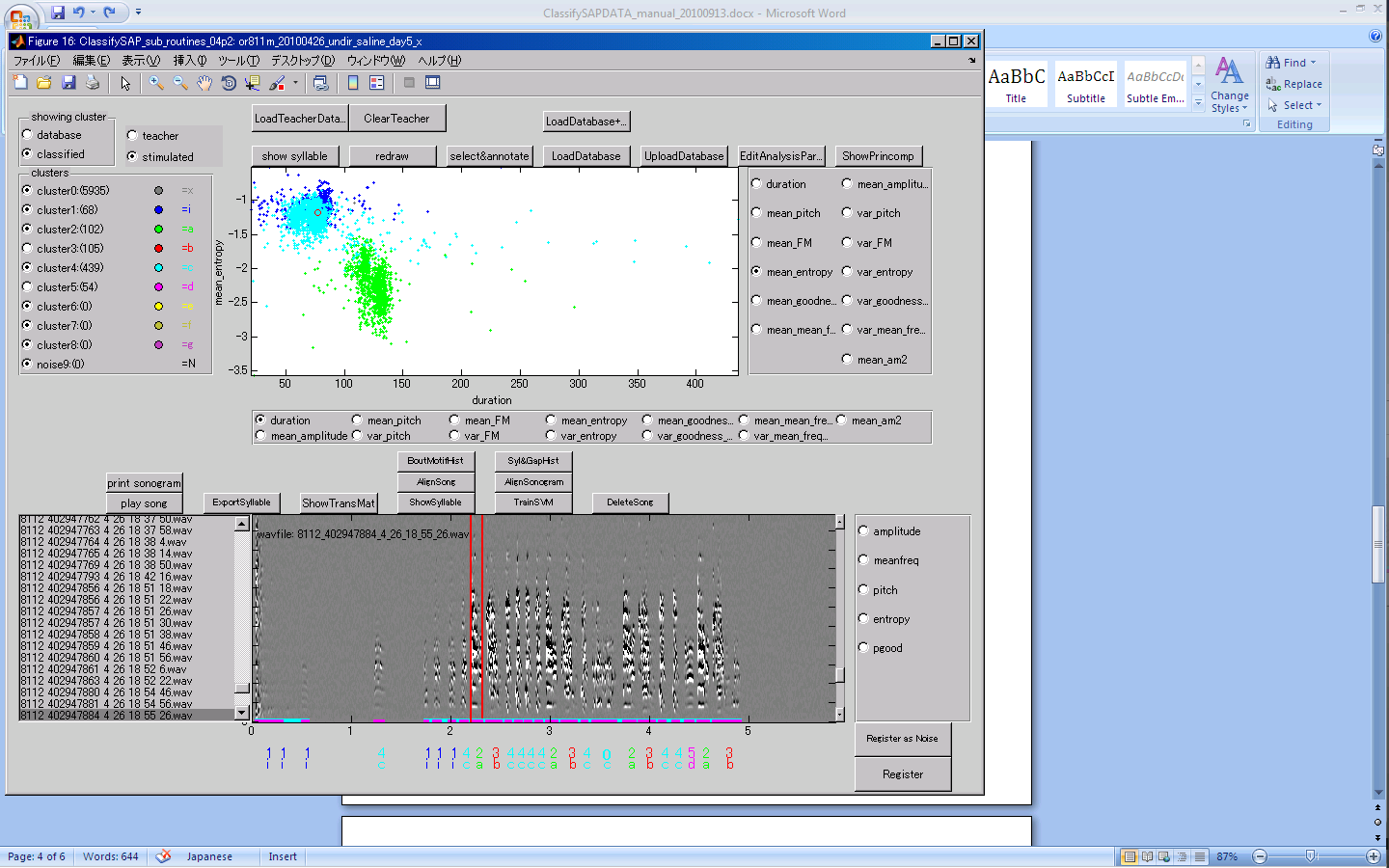
You need (b, MATLAB database toolbox) to upload/download syllable data to (c) mysql database, through (d) mysql ODBC connector.

After installing them, start MATLAB and add mysql databse in the database source (only for the first time).

>> querybuilder

* Query🡪 Define ODBC Data Source 🡪 Add, and add mysql.

**2.　Syllable Classification**

****

**2.0 Connect to mysql from MATLAB**

I guess this is a bug in MATLAB, but every time when you start MATLAB, you need to connect to mysql database by using

>> querybuilder

Query🡪 Define ODBC Data Source 🡪Double-click “mysql”,

Hit “Test”. You should see “Success!”

Then, start the sound analysis program.

>> ClassifySAPDATA\_\*\*(choose latest version)

**Basic flow of syllable classification**

2.1) Load database

2.2) Manually annotate syllables

2.3) Train SVM (support vector machine) and predict the remaining syllable classes

2.4) Check errors. Repeat 2.2&2.3 until no obvious errors.

2.5) Upload database

2.6) Do more analysis based on identified syllables

**Details**

* 1. “LoadDatabase” to load syllable database (which you generated in KinkaSongAnalysis or SAP batch analysis) from mysql.

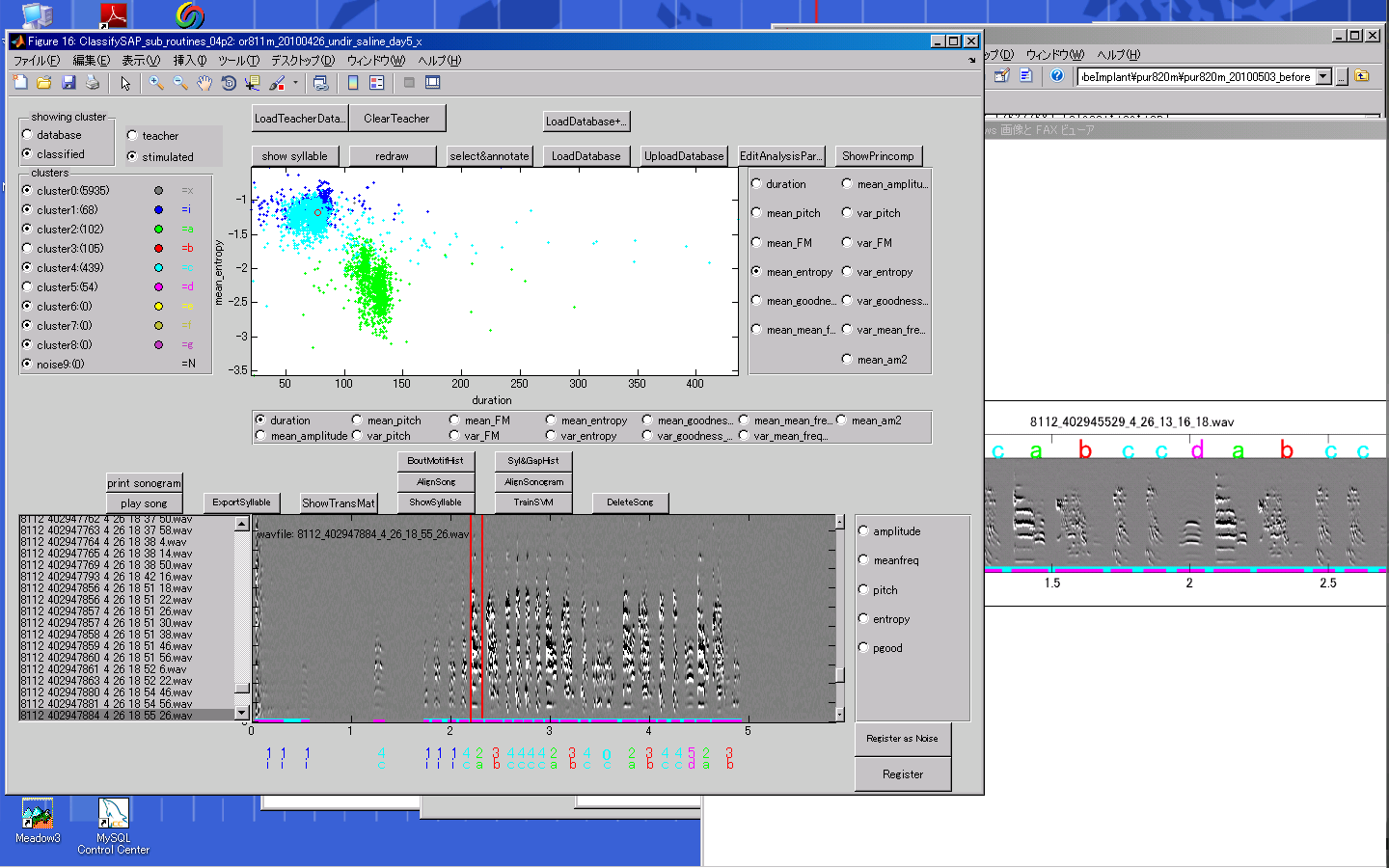
When you load a mysql database named “WAVEDIR”, you will be asked the upload-table name where your new classification data will be uploaded. The default table name is, “WAVEDIR \_x”. I use different name to protect the original data from overwritten. When you re-analyze your data, you can work on your previously analyzed database “WAVEDIR \_x”, or you can start from the beginning by loading your original database “WAVEDIR”.

Then, if your current directory does not contains the feature directory (that contains binary feature data generated by *KinkaSongAnalysis* or SAP), *ClassifySAPDATA* will ask you to locate the feature directory. The default directory *ClassifySAPDATA* look for is “WAVEDIR \_feature” .

Lastly, if your current directory does not contain the wave-file directory (“WAVEDIR” where your sound data (\*\*.wav) are stored), MATLAB will ask you to locate it.

* 1. Before start classifying, observe as much song as possible to have clear idea of the song structure, and potential problems of the specific birds. Common problem I had was, some birds sing syllables which tend to be fractionated by variable gap length. If that occurs too much, go back to syllable-detection part (KinkaSongAnalysis) and change the parameters of detection threshold. The smaller the threshold value, syllables can be less divided, but more chance to pick up noise. Minimum-gap duration is also useful parameter to obtain stable syllable detection results.

Manually annotate syllable classes to make teacher data like below;



Important!

Pre-determined syllable classes are;

**0: not classifiable**

**9: noise.**

You can use

1,2,…8 as identifiable syllable number.

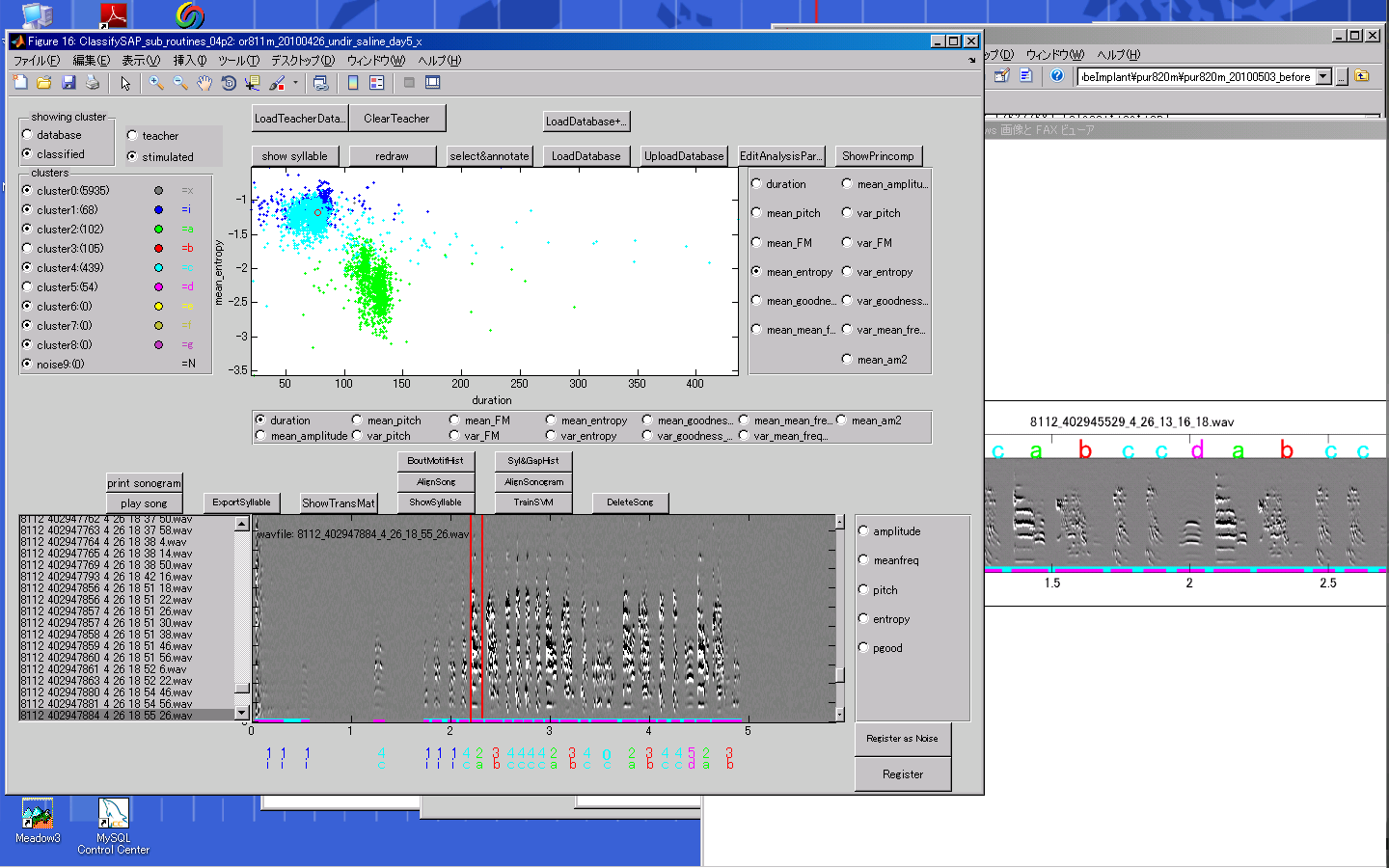
3. You can also register all the syllables in this song as teacher data, if you are confident that these classifications are correct.

2. Double click the number of the syllable to manually annotate the class of the syllable.

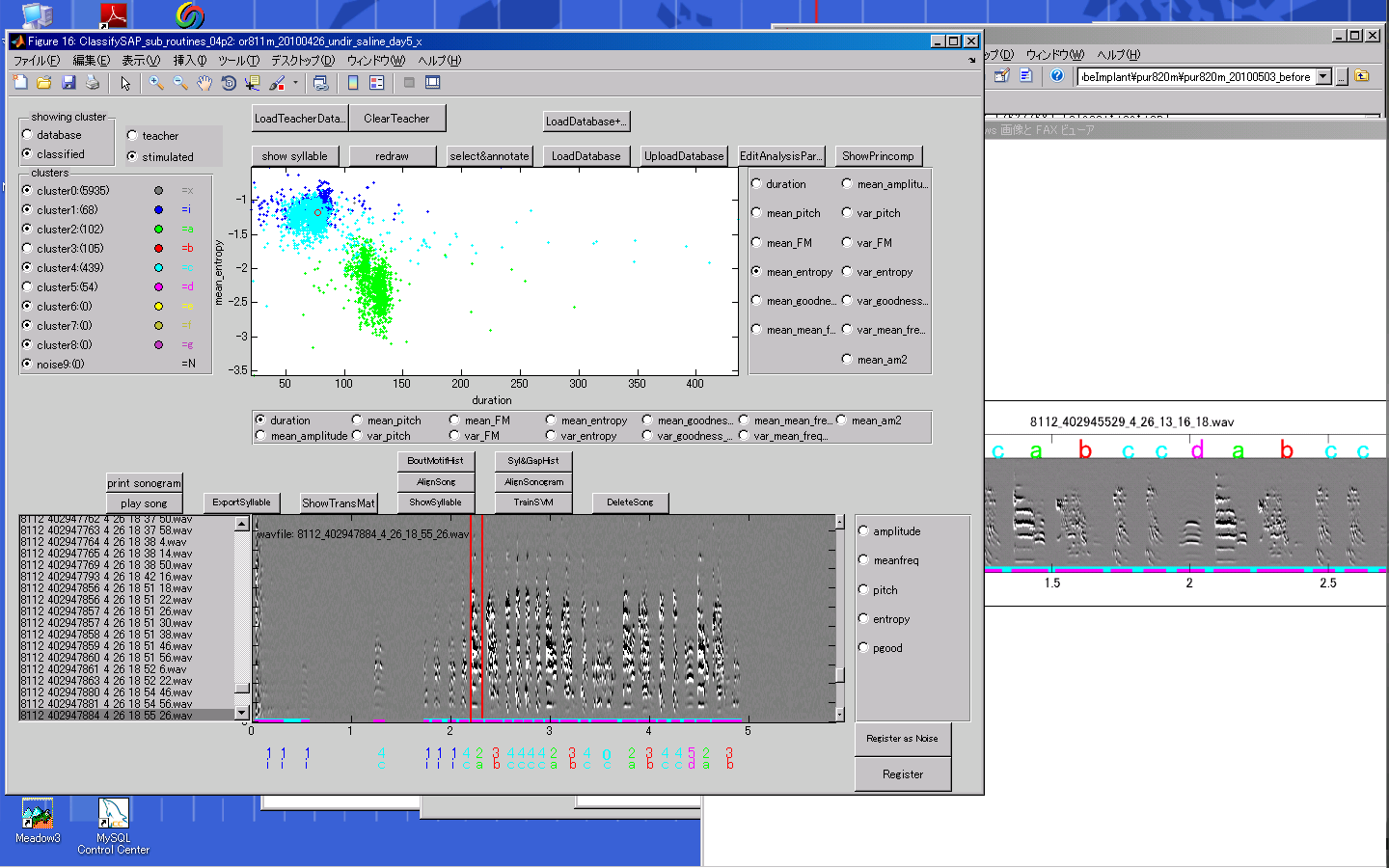
Note: If you are either in zoom mode or selection mode, you cannot select these numbers. Please unlock the zoom/select button to back to normal mode.

(This is needs to be fixed)

1. Double click the red bar to listen to the sound.
   1. “TrainSVM” to train SVM (support vector machine) to predict syllable classes. Cross-validation results will appear in matlab command window.



* 1. Hit radio-button “classified” in the left-top “Showing” panel to see the classified (predicted) results.



Radio buttons to change features of x-y dimensions.

Ratio buttons to hide/show clusters

Eye-inspect the clouds of syllable features and find suspicious mis-classifications.

To open the song file that contains a specific syllable, hit “Show syllable” button to target a syllable. You can click a nearby point of the target syllable, and *ClassifySAPData* will find the nearest syllable and open the song file that contains the targeted syllable. Back to 2.2) to annotate those syllables.

I would manually annotate 5-10 songs in this initial step, in different times of the day. Once you are confident with your results move to 5

Tips:

Miss-classifications often occur near the border of the class.

(Those are easy to fix as you increase the teacher data.)

A difficult cases of mis-classification which I often encountered was that some syllables contains similar sub-elements, and sometimes due to the fluctuation of the gap and sound amplitude, those sub-elements are separately detected as independent syllable from its original syllables. In this case, you can come back to change the detection threshold in *KinkaSongAnalysis,* or force to classify them. In this case, you may need to classify many (maybe more than 20-30) songs manually. Even in this case, it usually gives satisfactory results as long as they “really” sound different.

1. “Export syllable” to obtain the formatted syllable sequences (Important! When you update classification (hit TrainSVM with new teacher data), you need to run “Export syllable” to update the internal data to calculate transition matrix, syllable tree structure. BoutMotifHist and Syl&GapHist DOES NOT depend on syllable identification, so you don’t need to re-run it.)
2. “ShowTransMat” to calculate the transition matrix. In the transition matrix, you can find suspicious errors which often appears as blue (low probability) area. Back to 2 to correct them.)
3. For adult birds, “ShowSyllables” is also efficient way to find very minor classification errors. ShowSyllables displays the tree structure of the syllable sequence. You can click each syllable to open a song which contains a specific syllable sequence structure.
4. If you find a sound file containing just noise, you can delete it by “delete song”.
5. “BoutMotifHist” and “Syl&GapHist” will calculate bout duration histogram, and syllable&gap histogram.
6. “UploadDatabase” to upload your data.

**3. Analyze songs from different days of the same bird.**

Hit “Load teacher data” . It will load manually classified syllable data and allows you to use the same teacher data for different days.

**4. matlab commands to calculate averaged feature values**

All the data is stored in the struct (a type of variable that can hierarchically store multiple types of data) called *sap*. Typing

>> global sap;

in command window allows you to access this data.

It contains many details information of the program. Some are not relevant for the data analysis (like GUI data and their handles). Here are examples of important ones;

sap.data : a matrix of data that contains all the syllable data

M x max\_load\_N size, each column represents one syllable.

sap.pm.max\_load\_N: maximum number of syllables to download, default is 12000.

sap.pm.names : explanation of each row in sap.data.

You should be able to find the names of each row in sap.pm.names. For example,

4: 'start\_on'

5: 'duration'

6: 'mean\_pitch' ….

10: 'mean\_mean\_frequency' …

29: 'classified'

Some of these are accessible by typing sap.pm.\*\*\*\*\_index (like, sap.pm.classified\_cluster\_index returns 29.)

Therefore,

>> index\_cluster\_5=find(sap.data(sap.pm.classified\_cluster\_index,:)==5);

Returns the index of automatically classified cluster 5.

>> m\_pitch\_index=mean(sap.data(sap.pm.mean\_pitch\_index,index\_cluster\_5));

returns the mean position of the cepstrum, and actual mean pitch can be calculated

>> m\_pitch=sap.pm.Fsong/2/m\_pitch\_index;% in [Hz].

If you want to select manually clustered cluster 5,

Index\_cluster\_5=find(sap.data(sap.pm.cluster\_index,:)==5);