Bioinformatics Institute
The University of Auckland

Tutorial Matlab functions `dtwave_cluster`

Louis Ranjard - l.ranjard@auckland.ac.nz

May 12, 2014

Introduction

This tutorial illustrates the usage of `dtwave_cluster` package for the automatic classification of bird song syllables. Please note that `dtwave_cluster` was developed to classify large amount of recordings [1] therefore the small data used in this tutorial does not constitute a representative dataset. However, this document shows how to use the package to perform a typical short and simple analysis.

1 Data set: Little Barrier Island Song Recordings

A set of 11 song recordings were collected on Little Barrier Island using a SongMeter equipment (Figure 1). Songs were first manually subdivided into their constitutive syllables and the sound signal of each syllable was saved as an independent WAV file (directory *data*). The song segmentation step can also be performed automatically, especially when analysing large dataset, but this is out of the scope of this tutorial.
The species name and syllable files for each song are as below:

- 4 saddleback songs
  - files 1.wav to 4.wav
  - files 5.wav to 9.wav
  - files 25.wav to 31.wav
  - files 37.wav to 42.wav
- 5 hihi songs
  - file 32.wav
  - file 33.wav
  - file 34.wav
  - file 35.wav
  - file 36.wav
- 1 long-tailed cuckoo song
  - files 10.wav to 19.wav
- 1 bellbird song
  - files 20.wav to 24.wav

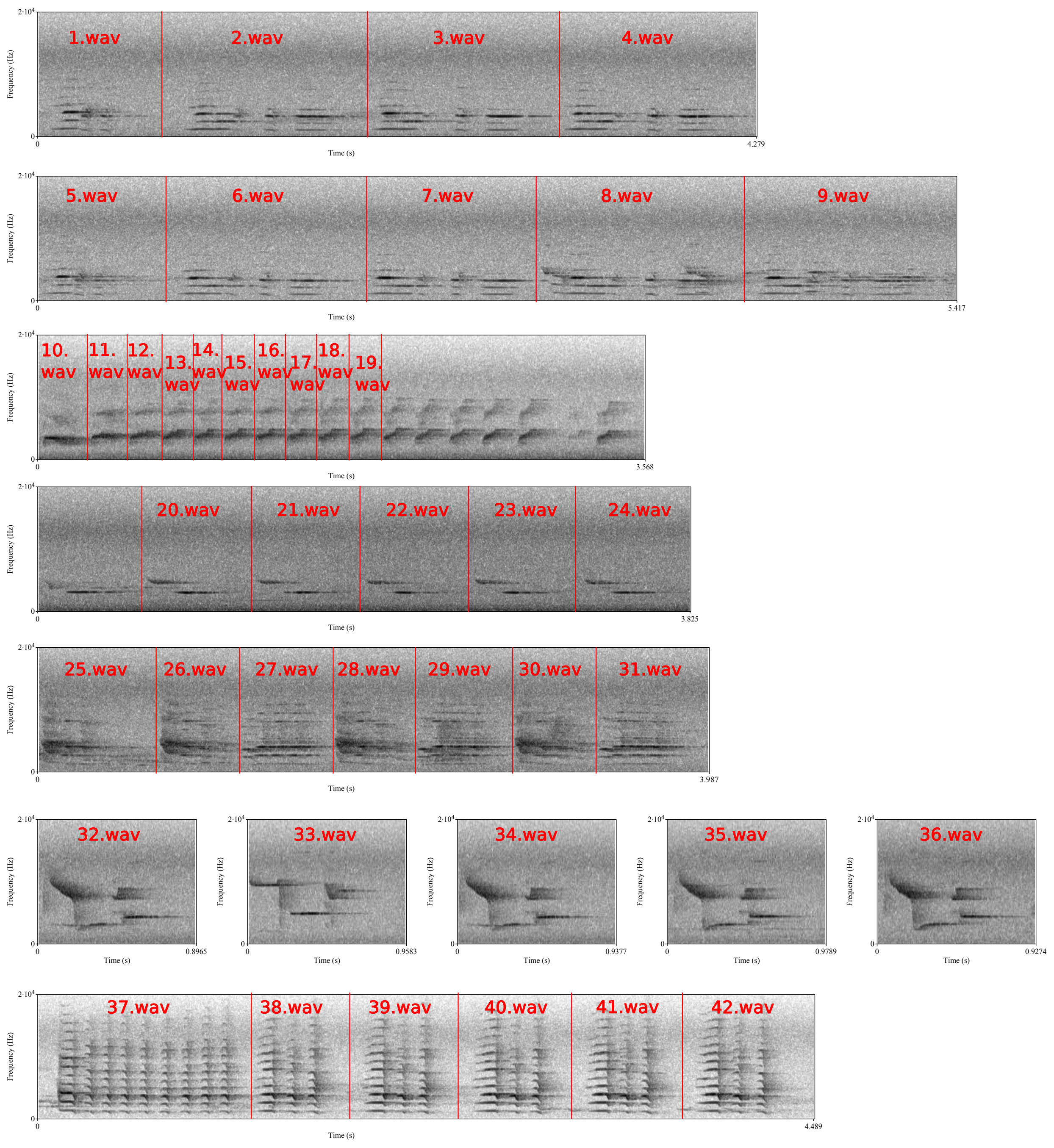

Figure 1: Spectrograms of recorded songs with syllable boundaries and file names.

2 Song Classification

First add the `dtwave_cluster` uncompressed directory to your Matlab `PATH`. The function `dtwave_cluster` can now be called to perform the automatic classification of the syllable recordings. Several options can be passed to this function. In particular, we encode the songs with a short time window, 0.02 seconds, and partial overlap, 0.005 seconds, because we are interested in short variation in the signal. A larger window would be more efficient if for example one is aiming at classifying whole song recordings. The number of evolving tree classifications is set to 10 and the default values for the other parameters are used. At the Matlab prompt, type:

```
[syllable,~,~,classrep] = dtwave_cluster('./data', 'wintime',0.02,'hoptime',0.005,'nrepeat',10) ;
```

```
****** dtwave_cluter: unsupervised sound classification ******

42 WAV files loaded
Parameters: 4, 0.5, 0.05, 2, 1, 4, 2, 2, 0.95

--------------------- classification tree: 1/10 ---------------------
epoch time -- Neighborhood Size, Mapping precision, Tree size, Leaves number
epoch 1/4 01-04-2014 18:32 -- 1, 10.0715, 41, 31
epoch 2/4 01-04-2014 18:32 -- 2, 10.2637, 41, 31
epoch 3/4 01-04-2014 18:32 -- 2, 10.5012, 44, 33
epoch 4/4 01-04-2014 18:32 -- 1, 9.43704, 71, 51
15 clusters
Davies-Bouldin indice: 5.15243
Classified syllable 1

--------------------- classification tree: 2/10 ---------------------
epoch time -- Neighborhood Size, Mapping precision, Tree size, Leaves number
epoch 1/4 01-04-2014 18:32 -- 1, 9.91937, 41, 31
epoch 2/4 01-04-2014 18:32 -- 2, 9.97057, 44, 33
epoch 3/4 01-04-2014 18:32 -- 2, 9.5719, 68, 49
epoch 4/4 01-04-2014 18:32 -- 1, 9.82024, 95, 67
17 clusters
Davies-Bouldin indice: 60.3539
Classified syllable 1

--------------------- classification tree: 3/10 ---------------------
...
```

A total of 10 classifications is performed. For each classification replicate, the data structure `classrep` contains the cluster number attributed to each syllable recording in column. For example, it is possible to look at the classifications of the recordings 1 to 10.

```
classrep(1:10,:)
```

```
ans =

     4     1     5     3     9     3     6     1     3     3
    13     4     8    14    11    10     3     9     1     9
    13     4     3    12    10     9     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
     8     4     3    12    10     8     3     9     1     9
    13     4     3    15    10     8     3     9     1     9
```

It appears that the recordings 4 to 9 are consistently grouped together while the recordings 2, 3 and 10 are frequently grouped in the same cluster too. On the other hand, the recording 1 is never clustered with the recordings 2 to 10. This result shows that the syllables 2 to 10 share high acoustic similarity and the syllable 1 is different, indicating that it probably belongs to a different type.
The data structure `syllable` contains the file name corresponding to each recording. Looking at the file names for the recording 1 to 10 allows us to map the recordings back to the original song files.

```
fprintf(1,'%s\n',syllable(1:10).filename)
```

```
1.wav
10.wav
11.wav
12.wav
13.wav
14.wav
15.wav
16.wav
17.wav
18.wav
```

The syllables 2 to 10 (10.wav, 11.wav,..., 18.wav) belongs to the same song recordings showing high consistency between the syllables of this song.

3 Classification analysis

The data structure `classrep` can be used to investigate biological questions related to the song recordings. These questions can be treated by integrating results over replicates. For example, one can ask how many syllables are shared between two sets of saddleback songs. We can answer that question by counting the number of times the syllables of the two different sets of songs are classified together. Let's consider set 1 as the first two saddleback songs (syllable 1.wav to 9.wav) and the two last saddleback songs (syllables 25.wav to 31.wav and syllables 37.wav to 42.wav).
First, we need to retrieve the indexes of the syllable of each set.

```
set1 = find( ismember({syllable.filename},{'1.wav';'2.wav';'3.wav';'4.wav';'9.wav';...
      '37.wav';'38.wav';'39.wav';'40.wav';'41.wav';'42.wav'})==1 ) ;
set2 = find( ismember({syllable.filename},{'5.wav';'6.wav';'7.wav';'8.wav';'25.wav';...
      '26.wav';'27.wav';'28.wav';'29.wav';'30.wav';'31.wav'})==1 ) ;
```

Then, we compare the cluster indices of the syllable of the two sets in each classification.

```
common = zeros(1,10) ;
for n=1:10
  common_syllable = numel( intersect( classrep(set2,n), classrep(set1,n)) ) ;
  unique_syllable = numel( unique( [classrep(set2,n); classrep(set1,n) ]) ) ;
  common(n) = common_syllable/unique_syllable ;
end
mean(common)
```

```
ans =

    0.1615
```

Therefore, on average the two sets share about 16% of their syllables. These results could be compared across sets of songs belonging to different populations for example.

4 Classification replicates summary

A way to summarize the classification replicates is to estimate the distances between recordings using the Jaccard distance, i.e. one minus the percentage of clusters that differ. The distances between recordings can then be represented on a summary dendrogram.

```
Z = linkage(classrep,'average','jaccard') ;
dendrogram(Z,50,'Labels',{syllab1.filename},'Orientation','left') ;
```

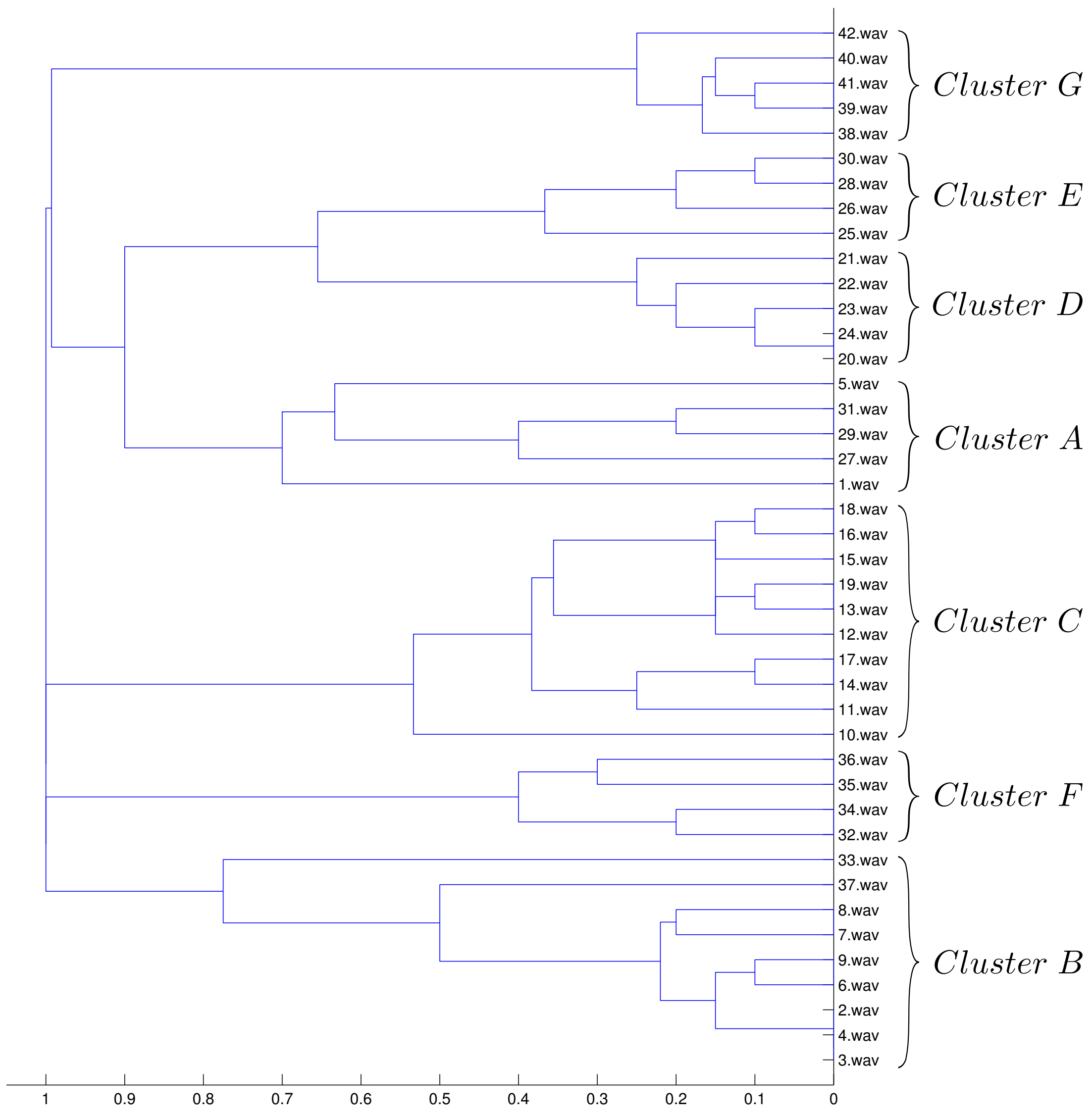

Figure 2: Syllable pairwise distances.

Figure 4 represents the syllable pairwise distances as defined by the classification replicates. From the dendrogram it clearly appears that the syllables group into clusters. Each group constitutes a syllable cluster and is refereed to with a letter from A to G.
It is now possible to encode the songs as a sequence of syllables using the summary dendrogram clusters. Figure 4 shows the song spectrograms with the cluster indicated for each syllable (A to G).

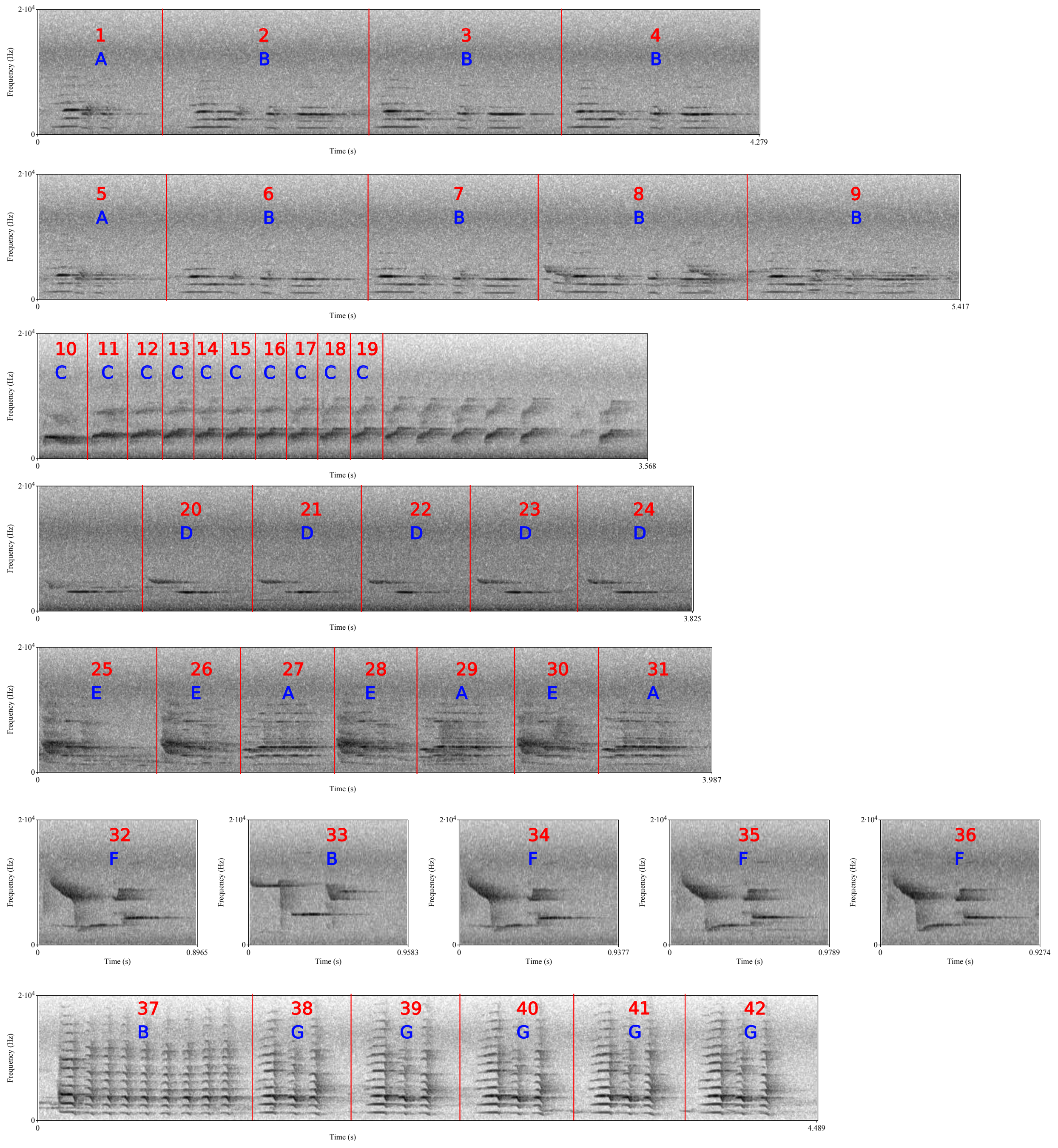

Figure 3: Song with encoded syllables following classification replicates summary.

5 Classification of the original songs

directory $/home/louis/Documents/DTWave/test/Little_Barrier_Island/extracted_songs$