Log

Data cleaning

The CSV data is supplied by Edoardo Ferrante on Kaggle. This data was created using the Librosa package for Python. Librosa outputs the intensity of a certain tone at different time intervals from the provided sound file. There is a Python script available from the author that explains how the data was transformed from sound files to numeric data points, but this script has a specific shortcoming.

First let's have a look at how Librosa normally outputs the chromogram data:

	0	1	2	3	4	5	6
chromogram_0	0.68661	0.67378	0.65758	0.66149	0.68533	0.72239	0.76395
chromogram_1	0.91368	0.88148	0.85024	0.82476	0.82282	0.83024	0.83908
chromogram_2	0.98221	0.97060	0.95834	0.94729	0.94785	0.95189	0.95408
chromogram_3	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
chromogram_4	0.96223	0.95790	0.95436	0.95403	0.94404	0.93285	0.91552
chromogram_5	0.92098	0.89960	0.88007	0.86290	0.84262	0.82280	0.80381
chromogram_6	0.87591	0.85544	0.83320	0.80870	0.78241	0.75819	0.73456
chromogram_7	0.79397	0.79418	0.79848	0.80535	0.80741	0.80750	0.80913
chromogram_8	0.62856	0.64859	0.68255	0.72441	0.76565	0.80078	0.82178
chromogram_9	0.41881	0.41914	0.44003	0.48525	0.54615	0.61608	0.68581
chromogram_10	0.36895	0.38299	0.40678	0.43114	0.45138	0.47237	0.52333
chromogram_11	0.33855	0.34676	0.35929	0.37378	0.38602	0.39931	0.42756

As you can see, the output is a neat array containing the values of 12 different tones at different time intervals. This data is sorted and can be read by Librosa.

Now onto the issue; the provided dataset contains this data in a stacked order, so each sample only takes up one row. This is a good idea, but due to sorting by alphabetical order the original order is lost. The order is important because we are working with data over time. This is not a problem if the trained model is only used on the provided test data, but we want the trained model to work in as many situations as possible.

Here is a look at the provided data:

id	chromogram_0_0	chromogram_0_1	chromogram_0_10	chromogram_0_11
0	0.997943662321316	0.832392210770135	0.7653861625931	0.70427464132375
1	0.996254885931866	0.839119599044146	0.760416790506312	0.705141765139875
2	0.970810156116343	0.823539694937237	0.759508104372184	0.709057883677716
3	1	0.855558393364941	0.752038009313116	0.710976936190937
4	1	0.884304523555434	0.741884532311754	0.714775207828629
5	0.971867873978603	0.824311712155432	0.755293860709407	0.71448132195049
6	1	0.835499361583387	0.751917158063063	0.717361992854453
7	0.978929855885584	0.827216718543843	0.751072631712318	0.718400862681119
8	1	0.895339720206626	0.733409813021178	0.722747412968086
9	0.967651828343747	0.823697857901917	0.746005680687241	0.721194823494439
10	0.993699774531599	0.847257121555946	0.734368883301346	0.726420069139032
11	0.00947350497274455	0.00699383738737368	0.372026644035831	0.0516494292032762

id	chromogram_0_0	chromogram_0_1	chromogram_0_10	chromogram_0_11
12	0.00982270123521504	0.00712337798131429	0.371129653847745	0.051631441504244

Each row contains a stack of chromogram data in a non-sequential order. The end of the array also contains the species of the corresponding bird and some spectral centroid data. These collumns need to be deleted since we want to predict using only the chromogram data.

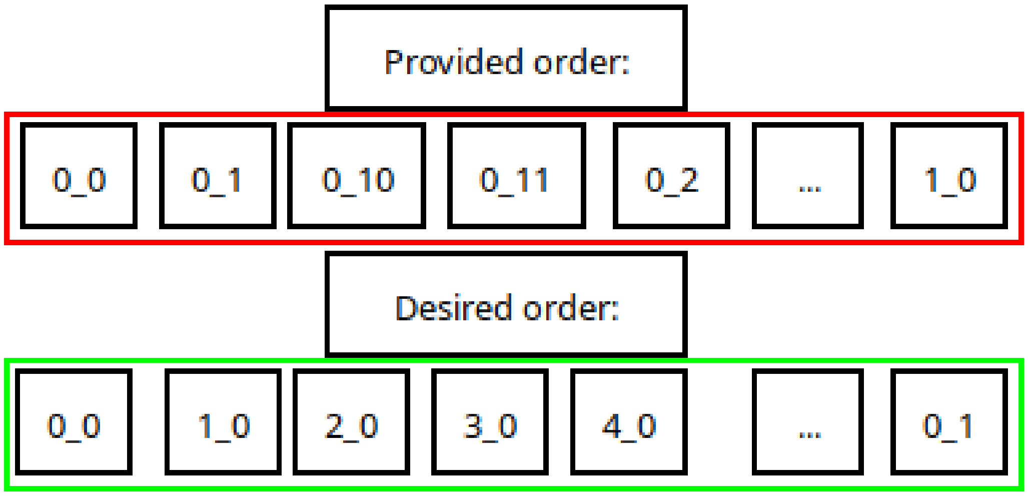


Figure 1: Comparison of the data order

Figure 1 shows a comparison of the order of provided data and the ideal order the data should be sorted in. The following script processes the provided data to the aforementioned format and deletes the unnecessary columns.

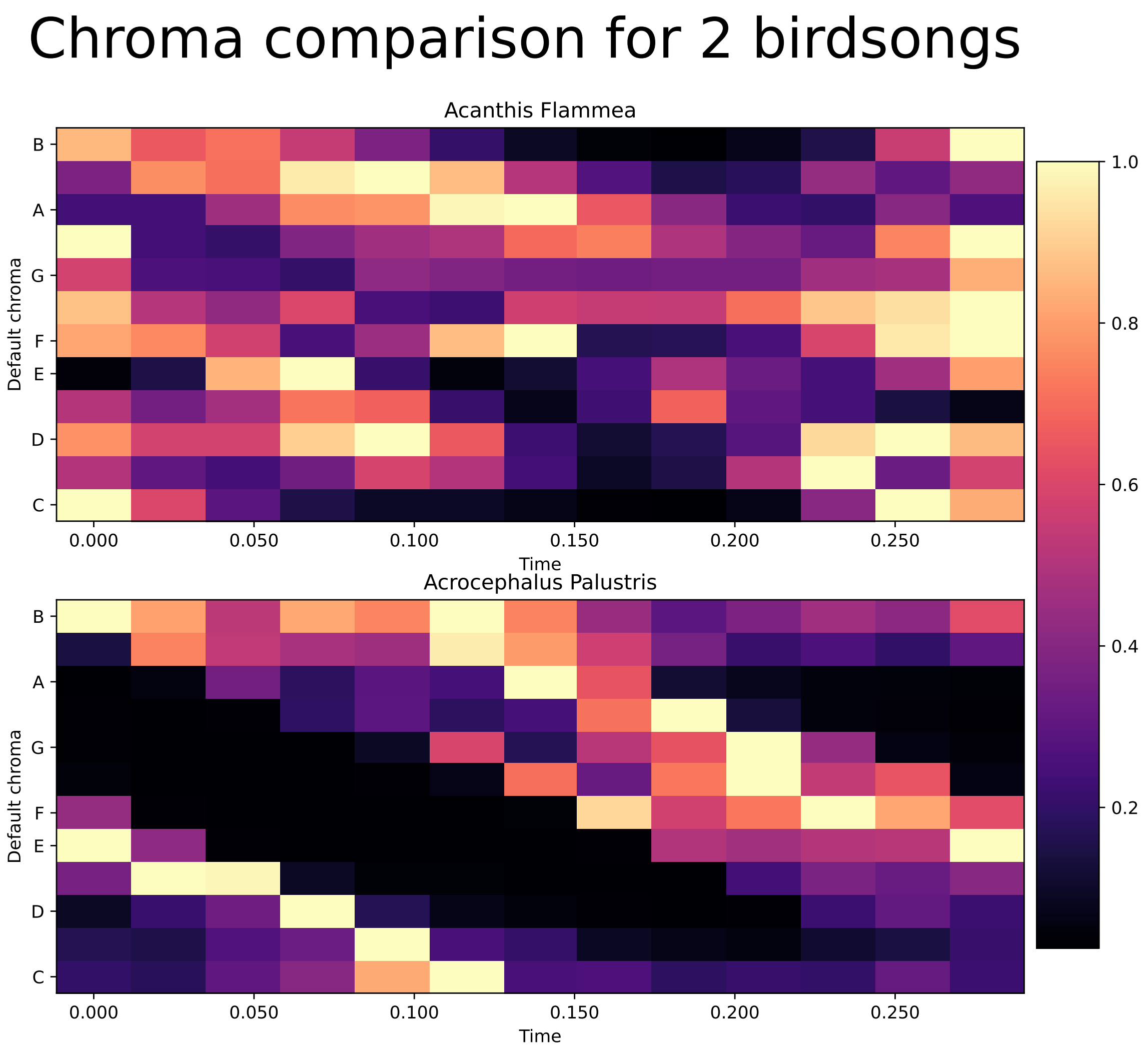
```
#!/usr/bin/env python3

"""
Script for cleaning and visualising the provided birdsong data.
"""
```

```
__author__ = "Niek Scholten"

# Imports
import librosa
from librosa import display
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

with open('data/train.csv', newline='') as csvfile:  # Open CSV and load the data into numpy
    my_data_train = np.genfromtxt(csvfile, delimiter=',')
    csvfile.close()
# Delete the last columns containing other data
my_data_train = np.delete(my_data_train, slice(-1, -16, -1), axis=1)
my_data_train = np.delete(my_data_train, 0, axis=1)  # Delete ID's
my_data_train = np.delete(my_data_train, 0, axis=0)  # Delete column names

with open('data/test.csv', newline='') as csvfile:  # Open CSV and load the data into numpy
```

```
    my_data_test = np.genfromtxt(csvfile, delimiter=',')
    csvfile.close()
# Delete the last columns containing other data
my_data_test = np.delete(my_data_test, slice(-1, -16, -1), axis=1)
my_data_test = np.delete(my_data_test, 0, axis=1)  # Delete ID's
my_data_test = np.delete(my_data_test, 0, axis=0)  # Delete column names

with open('data/train.csv', newline='') as csvfile:  # Open CSV and load the column names into numpy
    my_column_names = np.genfromtxt(csvfile, dtype=str, delimiter=',', skip_footer=1760)
    csvfile.close()
# Delete the last columns containing other data
my_column_names = np.delete(my_column_names, slice(-1, -16, -1))
my_column_names = np.delete(my_column_names, 0)  # Delete ID's

# The given data was sorted by alphabetical order, but this results in broken sequences
# Rearrange the data to the correct format for librosa
index = [0, 39, 52, 65, 78, 91, 104, 117, 130, 143, 13, 26,  # Chromogram 1
         1, 40, 53, 66, 79, 92, 105, 118, 131, 144, 14, 27,  # Chromogram 2
         5, 44, 57, 70, 83, 96, 109, 122, 135, 148, 18, 31,  # Chromogram 3
         6, 45, 58, 71, 84, 97, 110, 123, 136, 149, 19, 32,  # Chromogram 4
         7, 46, 59, 72, 85, 98, 111, 124, 137, 150, 20, 33,  # Chromogram 5
         8, 47, 60, 73, 86, 99, 112, 125, 138, 151, 21, 34,  # Chromogram 6
         9, 48, 61, 74, 87, 100, 113, 126, 139, 152, 22, 35,  # Chromogram 7
         10, 49, 62, 75, 88, 101, 114, 127, 140, 153, 23, 36,  # Chromogram 8
         11, 50, 63, 76, 89, 102, 115, 128, 141, 154, 24, 37,  # Chromogram 9
         12, 51, 64, 77, 90, 103, 116, 129, 142, 155, 25, 38,  # Chromogram 10
         2, 41, 54, 67, 80, 93, 106, 119, 132, 145, 15, 28,  # Chromogram 11
         3, 42, 55, 68, 81, 94, 107, 120, 133, 146, 16, 29,  # Chromogram 12
         4, 43, 56, 69, 82, 95, 108, 121, 134, 147, 17, 30]  # Chromogram 13
my_data_train = my_data_train[:, index]  # Apply index to the train data
my_data_test = my_data_test[:, index]  # Apply index to the test data
my_column_names = my_column_names[index]  # Apply the index to the collumn names

flammea_1 = np.empty((12, 13), int)  # Create empty array for this birdsong
# Add multiple columns form the original data as a new row
flammea_1 = np.append(flammea_1, my_data_train[0:1, 0:13], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 13:26], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 26:39], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 39:52], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 52:65], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 65:78], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 78:91], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 91:104], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 104:117], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 117:130], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 130:143], axis=0)
flammea_1 = np.append(flammea_1, my_data_train[0:1, 143:], axis=0)

flammea_1 = np.delete(flammea_1, slice(0, 12), axis=0)  # Delete empty cells

palustris_1 = np.empty((12, 13), int)  # Create empty array for this birdsong
# Add multiple columns form the original data as a new row
palustris_1 = np.append(palustris_1, my_data_train[20:21, 0:13], axis=0)
```

```python
palustris_1 = np.append(palustris_1, my_data_train[20:21, 13:26], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 26:39], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 39:52], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 52:65], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 65:78], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 78:91], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 91:104], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 104:117], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 117:130], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 130:143], axis=0)
palustris_1 = np.append(palustris_1, my_data_train[20:21, 143:], axis=0)

palustris_1 = np.delete(palustris_1, slice(0, 12), axis=0)  # Delete empty cells

fig, ax = plt.subplots(nrows=2, figsize=(10, 9))  # Create empty canvas for plots
img1 = librosa.display.specshow(flammea_1, y_axis='chroma', x_axis='time', ax=ax[0])
ax[0].set_title('Acanthis Flammea')
ax[0].set(ylabel='Default chroma')
ax[0].set(xlabel='Time')

img2 = librosa.display.specshow(palustris_1, y_axis='chroma', x_axis='time', ax=ax[1])
ax[1].set_title('Acrocephalus Palustris')
ax[1].set(ylabel='Default chroma')
ax[1].set(xlabel='Time')

cbar_ax = fig.add_axes([0.91, 0.15, 0.05, 0.7])  # Set axis for the colorbar
fig.colorbar(mappable=img1, cax=cbar_ax)
```

```python
fig.suptitle('Chroma comparison for 2 birdsongs', fontsize=32)

df = pd.DataFrame(my_data_train, columns=my_column_names)  # Export clean training data to csv
df.to_csv('data/dataframe_train.csv', index=False, header=True, sep=',')

df = pd.DataFrame(my_data_test, columns=my_column_names)  # Export clean testing data to csv
df.to_csv('data/dataframe_test.csv', index=False, header=True, sep=',')

plt.show()
```



This figure shows the chroma signature comparison for 2 fragments of different bird-species songs. It was created by transforming the input data to the correct format that is normally outputted by librosa, because the sound data is created using librosa. The x-axis shows the time of the sound fragment, while the y-axis shows the intensity of different tones on the given time.