1 Objective

This tutorial will help familiarize you with some of the capabilities of the Pandora toolbox. In it, I will be creating a database of birdsong syllables, then running some basic operations on it.

1.1 Get Pandora

– If you don't already have it, download Pandora from `software.incf.org/software/pandora/download` Under "Downloads & Documentation," click on "PANDORA Toolbox v1.3b" (ZIP)". Save this file somewhere relevant. □

– Make sure the Pandora folders "classes" and "functions" are in your MATLAB path. Only include those directories; don't include anything inside of them. □

1.2 Before Getting Started

– Get some data that you are interested in looking at. For this tutorial, the data will be birdsong syllables. The data can be any set of stuff that has something you're interested in looking at. □

– Think about the kinds of attributes that you want to look at in the data. For this example, we'll be looking at spectral entropy for each syllable. □

2 Creating a Database

First, we'll load all the files into a database. In this case, the files I'm using are ".mat" files with the attribute "sonogram" attached. Before loading the files in, I'd like to find the spectral entropy of the sonogram, then store each file, along with its identifying attributes, in the database.

Creating a database requires two steps. First, we need to create a "dataset" object. A dataset is an abstract base class for keeping information about what is in the database separate from the actual database. Items can be added to or removed from the dataset without adding or removing elements of the database.

There can be different implementations of datasets. For this tutorial, we're going to use the "fileset" object, which keeps track of what files we want to add to the database.

Note Methods in Pandora use the structure `method_name(param1, param2, ..., params)`, where `params1, params2, ...` are the necessary arguments for the method, and `params` is a structure containing keyword arguments, similar to Python. For example, to call the method `params_tests_fileset`,

I might do `params_tests_fileset('*.mat', 1/32000, 1, struct('key', 'value'))`. In this case, the argument `params` is the structure `struct('key', 'value')`. This passes the argument `value` to the variable `key`. If nothing is specified, it uses the default value for that parameter.

2.1 Creating a Dataset from a Group of Files

A fileset object finds all the files whose names match a particular pattern and loads them in. The constructor method for creating a fileset is `params_tests_fileset(file_pattern, dt, dy, id, params)`. The parameters are as follows:

- `file_pattern` The pattern for which files to load. For example, if you want to load all the `.mat` files in a directory, this argument could be `'*.mat'`. □
- `dt` The time resolution of the data. For example, the song data I am using is sampled at 32000 Hz, so I would use the value `1/32000`. □
- `dy` The y-axis resolution of the data. For this example, I am just going to put `1`. □
- `params` Extra arguments. Run `help params_tests_fileset` for a complete list. □

In this tutorial, the way files are loaded is important. There are two optional parameters that we need to pass:

- `fileParamsRegexp` The pattern for extracting information from the filename. We need to get two groups: `name` and `val`. For example, my filenames use the pattern `parameter-value`, so I would pass `(?<name>onset)-(?<val>.+)`. □
- `loadItemProfileFunc` If you want to change how files are loaded into the dataset, pass a custom function handle. This function needs to accept the parameters `dataset, index, param_row` and `props`. The function should return a `results_profile` object. A sample function is below. □

Sample function passed to `loadItemProfileFunc`:

```
function [ rprof ] = birdsong_mat_profile(dataset, index,
    paramRow, props)
filename = getItem(dataset, index);
file = load(fullfile(dataset.path, filename));
stim = false;
if strcmpi(file.type, 'stim')
    stim = true;
end
```

```
8 rprof = results_profile(struct('entropy', entropy(file.
    sonogram), 'stim', stim), [dataset.id, 'item=',
    num2str(index)]);
9 end
```

2.2 Creating a Database from a Dataset

Once the dataset has been specified, it is very simple to create a database. This can be done using the command `params_tests_db(my_data_set)`. This creates our database, which we can manipulate with Pandora.

Here is the sample code for this tutorial.

```
1 my_data_set = params_tests_fileset('*.mat', 1/32000, 1, '
    gr79pu85_1_20_15', struct('fileParamsRegexp', '(?<name
    >onset)-(?<val>[\d\.]+)', 'loadItemProfileFunc',
    @birdsong_mat_profile));
2 my_db = params_tests_db(my_data_set); % This step could
    take a while
```

3 Visualizing Data

Now that all of the data is in a database, we need to use it for something.

3.1 Selecting Data

In the database, each datapoint has three attributes: `onset`, `entropy` and `ItemIndex`. The first two attributes were defined by the user, and the third is a unique index. Each attribute is a column in the database, and each row is a datapoint. Selecting data is done using MatLab's matrix notation.

To select the first 10 data points in the database:

```
1 my_db(1:10, :)
```

To select particular attributes, both the attribute's name as a string or the attribute's value relative to the other attributes can be used:

```
1 my_db(:, 1)
2 my_db(:, 'entropy')
3 my_db(:, {'onset', 'entropy'})
```