

BaleenWhale Acoustics Inspection Tool (BAIT)

Reference Manual

Wilfried Beslin
14 December 2023

Table of Contents

1. INTRODUCTION	2
2. GETTING AND RUNNING BAIT	2
3. BAIT PROGRAM FILES	3
3.1 MATLAB Scripts	3
3.2 Other Folders and Files	3
4. RUNNING SCRIPTS – OVERVIEW	4
4.1 Simple Usage (No Input Arguments)	4
4.2 Running Scripts with Input Arguments	4
4.3 Parameter Files	5
5. VALIDATING DETECTIONS AND MISSED CALLS (BAIT_validate)	6
5.1 BAIT_validate Input	6
5.1.1 Autodetection Spreadsheet File	6
5.1.2 Folder of WAV Files	8
5.1.3 Parameters	9
5.2. Using the BAIT_validate GUI	12
5.2.1 Keyboard Inputs	13
5.2.2 Button Inputs	14
6. IMPORTING LFDCS AUTODETECTION CSV TABLES (BAIT_convertLFDCSTable)	16
6.1 Command-Line Arguments for BAIT_convertLFDCSTable	16
6.2 Parameters for BAIT_convertLFDCSTable	18
7. ISOLATING AND PREVIEWING LFDCS DETECTIONS (BAIT_isolateLFDCSDetections)	20
7.1 Command-Line Arguments for BAIT_isolateLFDCSDetections	20
7.2 Parameters for BAIT_isolateLFDCSDetections	21
8. REFERENCES	23
APPENDIX: LIST OF SPECTROGRAM COLORMAPS	24

1. INTRODUCTION

This program started as a simple spectrogram viewing tool to inspect right whale upcall detections returned by a neural network detector that was being developed by MERIDIAN at Dalhousie University. It later evolved into a validation tool emulating the *browse_autodetections* module from LFDCS (Baumgartner and Mussoline, 2011), and has grown to support various new features, including the marking of missed calls and general annotations within files where autodetections are present. A conversion script was also added to enable detections from LFDCS to be validated using this program. Since then, its primary use case has shifted from validating MERIDIAN detector output to validating LFDCS output. Inspiration for this program was taken from both LFDCS and JASCO's PAMlab Lite.

This manual applies to the version released on MARTeamWhale's GitHub page on 2023-12-08. This version requires at least MATLAB R2018b to run. Later releases of MATLAB may also work but this has not been tested.

2. GETTING AND RUNNING BAIT

BAIT is available from the MARTeamWhale GitHub page here:

<https://github.com/MARTeamWhale/BaleenWhale-Acoustics-Inspection-Tool>.

Clone or download the BAIT repository from GitHub and place it somewhere convenient on your computer. Next, you will need to tell MATLAB where to find BAIT by adding its root folder to the MATLAB path. There are different ways of doing this, but the simplest is to use the MATLAB graphical interface, following these instructions:

- 1) Click on the *Home* tab if it isn't already open
- 2) Look for and click on the *Set Path* button (within the *ENVIRONMENT* panel)
- 3) This will bring up another window where you can add folders to MATLAB's search path. Click the *Add Folder* button and select the BAIT folder.
- 4) Click *Save*, then *Close*.

BAIT also requires another package distributed by MARTeamWhale in order to run, called *MATLAB Utilities for Cetacean Acoustics* (MUCA; <https://github.com/MARTeamWhale/MATLAB-Utilities-for-Cetacean-Acoustics>). The process for installing MUCA is the same as that for BAIT.

Once you have both BAIT and MUCA downloaded and added to the MATLAB path, you will be able to run the BAIT scripts in MATLAB

3. BAIT PROGRAM FILES

This section gives an overview of the program files present within the BAIT folder. Make sure these files are present in your root BAIT folder, and that this folder is on your MATLAB path.



Unless you wish to modify the code and know what you are doing, do not change the names or contents of any of the following files or folders, except where otherwise noted. Doing so could cause the program to stop working correctly.

3.1 MATLAB Scripts

There are currently three script files that users of BAIT may run directly.

- **BAIT_validate.m** – This is the main BAIT script. Launches a graphical user interface (GUI) application that enables users to browse through and validate autodetections, as well as mark the presence of missed calls and other signals of interest.
- **BAIT_convertLFDCSTable.m** – Converts LFDCS autodetection CSV tables into Microsoft Excel spreadsheets (XLSX) in a format that BAIT can read. Any call validations that were made in LFDCS using *browse_autodetections* can also be imported into the spreadsheet using this script.
- **BAIT_isolateLFDCSDetections.m** – Generates short audio clips and/or spectrogram images from raw audio data based on LFDCS detections. Note that this uses the original LFDCS CSV tables, not the BAIT Excel spreadsheets.

3.2 Other Folders and Files

Non-script files include the following:

- **+BAIT** – This folder contains code and other resources that are required for the BAIT scripts to run. There is no need to access this folder unless you wish to view or edit the BAIT code yourself.
- **PARAMS** – This folder stores parameter files for the various BAIT scripts. This is where default parameters are kept. Custom parameter files may be added here as well. Be careful not to rename the subfolders or the default files, as this will break functionality.
- **BAIT_ReferenceManual.pdf** – This document
- **SUMMARY_OF_IMPORTANT_CHANGES.pdf** – Provides a concise summary of the major usability changes of the detection validation tool following its transition from the MERIDIAN-focused version to the BAIT version.

4. RUNNING SCRIPTS – OVERVIEW

All three scripts may be run with or without input arguments. They also use parameter files to store certain settings. Default parameter files exist for each script, but it is possible to specify custom parameter files as well.

4.1 Simple Usage (No Input Arguments)

The most basic way to run a script is to click the *Run* button in the *Editor* tab in the MATLAB interface, or by simply typing the script name in the command line; for example:

```
BAIT_validate
```

Doing either of these will run the script with default parameters, and the user will be prompted to specify other arguments such as input file paths.

4.2 Running Scripts with Input Arguments

A more flexible way to run scripts is by entering input arguments in the command line, using MATLAB's *Name-Value* syntax. For example:

```
dataFile = 'C:\path\to\data\file.xlsx';  
wavDir = 'D:\path\to\folder\of\WAV\files';  
BAIT_validate('data_file',dataFile, 'audio_folder',wavDir);
```

In this particular example, `'data_file'` and `'audio_folder'` correspond to the *names* of arguments supported by the script `BAIT_validate`, and `dataFile` and `wavDir` are their respective *values* assigned by the user. `dataFile` in this case is a MATLAB variable specifying the path to the input data file for `BAIT_validate` (an XLSX file); `wavDir` is a similar variable specifying the folder of WAV files associated with the relevant data.

Note that name-value pairs can be specified in any order, and that every possible pair supported by a script does not necessarily need to be specified at once. For instance, in the above example, the script `BAIT_validate` also supports a third argument, `'params'`, which was not specified. This simply means that the `'params'` argument will use default values. Any name-value pair that is not specified explicitly will either use default values or prompt the user for more information at runtime, depending on the type of argument.

Specifying input arguments in the command window this way makes it possible to run scripts without prompting the user for additional information. This method is also more efficient if scripts need to be re-run multiple times using the same files. Supported name-value pair arguments for each script will be described in later sections. They are also described within the headers (i.e., the top comment blocks) of each script file. It is possible to view the headers by either opening the script files in MATLAB, or by using the `help` command like so:

```
help BAIT_validate
```

4.3 Parameter Files

Each script has various parameters in addition to input arguments that are specified in a *parameter file*. These files are somewhat similar to the paramfiles used by the *reformat_detect_classify* module in LFDCS. They are text files (with a *.txt* extension) that define certain variables specific to each script.

Each script contains a default parameter file, but custom parameter files can be created and read as well. The default parameters are loaded automatically if an alternative file is not specified.

The root BAIT folder contains a folder called *PARAMS* that is intended to store parameter files. Each script has its own subfolder within the *PARAMS* folder; this is where default files are located. Custom parameter files may be added to these subfolders as well. While it is not strictly necessary to store custom parameter files within the *PARAMS* folder tree, doing so makes it slightly easier to load them, as will be shown later. The default parameter files can serve as a template for creating custom files.



Do not rename any of the subfolders within the *PARAMS* folder tree. Likewise, the default parameter files should not be renamed or removed. Doing so will break BAIT's parameter file system and will prevent it from working properly.

While there is nothing technically to stop you from modifying default parameter files, doing so is not recommended. Creating new parameter files is the preferred way to pass custom parameter values to scripts.

When creating custom parameter files, you may edit the parameter values as appropriate, and write whatever you wish in the comment block at the top of the file. Everything else should remain the same.

To use a custom parameter file when running a script, you must specify the file in the command line as an input argument. The argument name for parameter files for all scripts is *'params'*. If your parameter files are stored in the appropriate *PARAMS* subfolder, then you only need to specify the name of the file. For example:

```
BAIT_validate('params', 'my_params')
```

It is also possible to specify parameter files that exist outside the *PARAMS* folder tree, but in this case, you will need to specify the path to the file. For example:

```
BAIT_validate('params', 'C:/Users/BeslinW/Desktop/my_params')
```

Note that the *.txt* extension in the parameter file name does not need to be specified in the command line (though it can), but the actual file must have this extension.



Using command-line input arguments is currently the only way to specify the use of custom parameter files. **You will not be asked to choose parameters interactively.**

The specific parameters supported by each script will be covered throughout this document in the relevant sections.

5. VALIDATING DETECTIONS AND MISSED CALLS (BAIT_validate)

The GUI application for validating detections and marking missed or other calls is launched by using the script `BAIT_validate`. This section describes how to use `BAIT_validate`: its input files, parameters, and how to navigate the GUI.

5.1 BAIT_validate Input

`BAIT_validate` requires two to three pieces of information to run: a spreadsheet file of autodetections, a folder of raw audio (WAV) files, and optionally, a list of parameter values.

5.1.1 Autodetection Spreadsheet File

This must be a XLSX file supported by BAIT (i.e., using the sheets and columns specified in the file `+BAIT/OutputTemplate.xlsx`). LFDCS autodetection tables can be converted into this format using the script `BAIT_convertLFDCSTable` (validations you already made in LFDCS may also carry over when using this script); this is discussed further in section [6. IMPORTING LFDCS AUTODETECTION CSV TABLES](#). You can use either a new file, or resume working on one that you have started to process with `BAIT_validate`. In the latter case, the program will automatically pick up where you left off.



It is NOT recommended to modify the autodetection spreadsheet manually, e.g., in Microsoft Excel. Doing so could have unexpected effects or may cause the program to stop recognizing it altogether, as Excel is notorious for modifying data types without asking. If you need to edit anything manually, create a separate copy of the spreadsheet. Furthermore, if the output file is open while `BAIT_validate` is running, the program will be unable to save its updates.

The spreadsheet file can be specified in the command line as a name-value pair input argument with the name `'data_file'`. For example:

```
BAIT_validate('data_file',...  
              'C:/Users/BeslinW/Desktop/ROB_2018_04/detections.xlsx');
```

Or if using MATLAB variables:

```
in_file = 'C:/Users/BeslinW/Desktop/ROB_2018_04/detections.xlsx';  
BAIT_validate('data_file',in_file);
```

If you run `BAIT_validate` without specifying the `'data_file'` parameter, then you will be prompted to choose a file interactively instead.

Autodetection spreadsheet files contain three sheets, summarized as follows:

- **Detected** – Lists signals that have been detected by an autodetector (e.g., LFDCS)
- **Undetected** – Lists signals that have been manually marked as potential or definite calls that were missed by the autodetector
- **Annotations** – Lists arbitrary signals of interest marked by the user

Columns within each sheet are described in the next few subsections.



Because the program deals with both detected and undetected sounds, I have opted to refer to candidate calls in general as “signals” rather than “detections”.

Data Columns for “Detected” sheet

- **FileName** – Name of the audio file containing the detection
- **FileStart** – The start time of the audio file, in seconds, relative to the date-time 1970-01-01 00:00:00.
- **SigStart** – Start time of the detection, in seconds, relative to the start of the source audio file
- **SigEnd** – End time of the detection, in seconds, relative to the start of the source audio file
- **SigStartDateTime** – Absolute date and time of occurrence of the detection. This is equivalent to $1970-01-01\ 00:00:00 + \text{FileStart} + \text{SigStart}$.
- **Class_LFDCS** – For detections that have already been validated using LFDCS and imported using `BAIT_convertLFDCSTable`, this column lists the species label assigned to the detection by the user in LFDCS. Note that only certain labels are supported at the moment, and they are converted to character strings as follows:
 - 9999 → Correct
 - -9999 → Incorrect
 - 0 → Unknown
 - -32767 → Unclassified or blank
- **Class_MATLAB** – Validation label assigned by the user within `BAIT_validate`. The options are limited to the same strings as in the `Class_LFDCS` column. If the spreadsheet was created using `BAIT_convertLFDCSTable` and the original LFDCS table already had manual labels, then both `Class_LFDCS` and `Class_MATLAB` will be auto-filled with the same value. This is to allow the LFDCS validations to be visible and potentially changed when browsing with `BAIT_validate`. Note that if the LFDCS validation labels are indeed changed using `BAIT_validate`, only the values in `Class_MATLAB` will be updated accordingly; values in the `Class_LFDCS` column are mostly there for reference and never change after they have been imported with `BAIT_convertLFDCSTable`.

- **ReasonForUNK** – Includes special comments that must be entered if the user validates a detection as *Unknown*. These comments should explain why particular detections were marked as Unknown rather than Correct.
- **Comments** – General comments entered by the user

Data Columns for “Undetected” Sheet

- **FileName** – Name of the audio file containing the undetected call
- **FileStart** – The start time of the audio file, in seconds, relative to the date-time 1970-01-01 00:00:00.
- **SigStart** – Start time of the undetected call, in seconds, relative to the start of the source audio file
- **SigEnd** – End time of the undetected call, in seconds, relative to the start of the source audio file
- **SigStartDateTime** – Absolute date and time of occurrence of the undetected call. This is equivalent to 1970-01-01 00:00:00 + *FileStart* + *SigStart*.
- **Type** – Describes the level of confidence in the undetected call. May be either *Definite* or *Potential*.
- **Comments** – General comments entered by the user

Data Columns for “Annotations” Sheet

- **FileName** – name of the audio file containing the annotation
- **FileStart** – The start time of the audio file, in seconds, relative to the date-time 1970-01-01 00:00:00.
- **SigStart** – Start time of the annotation box, in seconds, relative to the start of the source audio file
- **SigEnd** – End time of the annotation box, in seconds, relative to the start of the source audio file
- **SigMinFreq** – Lower frequency bound of the annotation box, in Hertz
- **SigMaxFreq** – Upper frequency bound of the annotation box, in Hertz
- **SigStartDateTime** – Absolute date and time of occurrence of the annotation box. This is equivalent to 1970-01-01 00:00:00 + *FileStart* + *SigStart*.
- **Comments** – General comments entered by the user

5.1.2 Folder of WAV Files

This is the folder containing all WAV files referenced in the detection spreadsheet (i.e., all WAV files within which there are detections to view or validate). **It is important that there be no missing files here.**

As with the autodetections spreadsheet, the WAV folder can be specified in the command line, using the name `'audio_folder'`. For example:

```
BAIT_validate('audio_folder', ...  
              'C:/Users/BeslinW/Desktop/ROB_2018_04/rec');
```

Or with variables:

```
in_dir = 'C:/Users/BeslinW/Desktop/ROB_2018_04/rec';  
BAIT_validate('audio_folder', in_dir)
```

Also as with the spreadsheet file, the audio folder does not necessarily need to be specified in the command line this way. If not explicitly specified, the user will be prompted to choose a folder.



The program can technically work with recordings of any sampling rate, but there are some caveats. Higher sampling rates such as 256 kHz may perform more slowly, as may sampling rates that are not powers of two. Audio playback may also not work at all for higher rates. If possible, downsampling to something like 8 kHz or lower is preferable, but this is not strictly necessary. Note also that sampling rate will limit the maximum frequency that can be displayed.

5.1.3 Parameters

BAIT_validate has various parameters that you can adjust to customize how the interface looks and operates. These are specified in a parameter file, as discussed in section [4.3 Parameter Files](#). The folder for parameter files used by BAIT_validate is `PARAMS\validate`. Specify the use of a custom parameter file by using the `'params'` input argument.

The parameters for BAIT_validate are described below.

Audio Channel

- **ChannelNumber** – Determines which channel number to use in multi-channel recordings. This may be specified as either a number, or the string *prompt*. If set to *prompt*, then the user will be prompted to choose a channel if multiple channels are detected. If it is a specific number, then the script will use that number if possible. If the number specified does not correspond to a valid channel number (i.e., if it is greater than the actual number of channels in the recording), then the script will either fall back to the first channel (if the recording is single-channel) or throw an error (if the recording is multi-channel).
The default value is *prompt*, as this is guaranteed to work in all cases.

Spectrogram Settings

- **WinSize** – Spectrogram window size, measured in seconds. Essentially, this is the amount of audio data over which to calculate the Discrete Fourier Transform (DFT) for each step. Choosing an appropriate value is a tradeoff between achieving high resolution in time and high resolution in frequency: Higher values capture more frequency information, but at the cost of reduced detail in time.
The default value of 0.256 is based on Kirsebom et al. (2020).
- **StepSize** – Spectrogram stepping size, in seconds. It is the amount of time in which to step the window forward and calculate the DFT from the next chunk of data. This parameter can also be seen as the *perceived* time resolution; i.e., it is what determines where your “data points” will lie in the spectrogram on the time axis. Large values result in coarser-looking spectrograms, while smaller values result in a finer scale and are more computationally expensive. Note however that the *true* temporal resolution is dictated by *WinSize*. If you have a very large *WinSize* and a very small *StepSize*, it will *look* high-resolution, but the actual data will be very smoothed out (for example, rapid burst-pulse signals could appear continuous).
The amount of overlapping between subsequent windows is equal to $WinSize - StepSize$. *StepSize* should always be > 0 and generally $\leq WinSize/2$. Values greater than $WinSize/2$ may result in a loss of information depending on the function used for windowing (in this case, “hamming”). Values greater than *WinSize* would outright skip portions of the time series and should never be used.
The default value of 0.032 is based on Kirsebom et al. (2020).
- **NFFT_8kHz** – This is the equivalent number of points to use in the Fast Fourier Transform (FFT) algorithm used to calculate DFT, if the recording was sampled at 8 kHz. The actual number of points used will be adjusted for the data’s real sampling rate. NFFT is what determines the frequency step of the spectrogram, or *perceived* frequency resolution.
Using 8 kHz as a reference allows the frequency step to be conserved between sampling rates based on the simple calculation:

$$NFFT_{Final} = Fs(NFFT_{8kHz}/8000)$$

Where Fs is the real sampling rate. $NFFT_{Final}$ should always result in a number of samples that is greater than or equal to the number of samples resulting from *WinSize*, that is:

$$NFFT_{Final} \geq Fs \times WinSize$$

Similar to *StepSize* for the time axis, values of $NFFT_{Final}$ that are above $Fs \times WinSize$ make the spectrogram look smoother along the frequency axis, but they do not improve the *true* frequency resolution, which is determined by *WinSize*. Ideally, $NFFT_{Final}$ should be a power of two, as that is more computationally efficient.

The default value of 2048 was inferred from Kirsebom et al. (2020).

Plot Parameters

- **TSpanDefault** – This determines the time span, or duration, of the area that is plotted when you start the program or reset the plot. Measured in seconds. The time span will change as you zoom in/out across time (see section [5.2.1 Keyboard Inputs](#))
- **FMaxDefault** – This is the upper frequency limit of the plot when you start the program or reset the plot. Measured in Hertz. The upper frequency limit will change as you zoom in/out across time (see section [5.2.1 Keyboard Inputs](#))
- **TPanFactor** – Factor by which to step forward or back in time when panning across the spectrogram. This is measured relative to the current time span of the viewing window; for example, if the panning factor is 0.75 and the current view spans 8 sec, then panning right will cause the view to shift forward by 6 sec. See section [5.2.1 Keyboard Inputs](#) for more on panning.
- **TZoomFactor** – Factor by which to increase the time span of the plot when zooming out in time. Zooming in will reduce the time span by $1/TZoomFactor$. For example, if the current view spans 8 sec, zooming out with a factor of 2 will increase the time span to 16 sec, while zooming in will reduce it to 4 sec. See section [5.2.1 Keyboard Inputs](#) for more on zooming in time.
- **FZoomFactor** – Factor by which to increase the upper frequency limit of the plot when zooming out in frequency. Zooming in will reduce the upper frequency by $1/FZoomFactor$. For example, if the current upper frequency is 1000 Hz, zooming out with a factor of 2 will increase it to 2000 Hz, while zooming in will reduce it to 500 Hz. See section [5.2.1 Keyboard Inputs](#) for more on zooming in frequency.
- **DefaultColormap** – The name of the colormap to use for the spectrogram on startup. The colormap can also be changed interactively (see section [5.2 Using the BAIT_validate GUI](#)). The available colormaps are listed in the appendix [LIST OF SPECTROGRAM COLORMAPS](#).

Marker Parameters

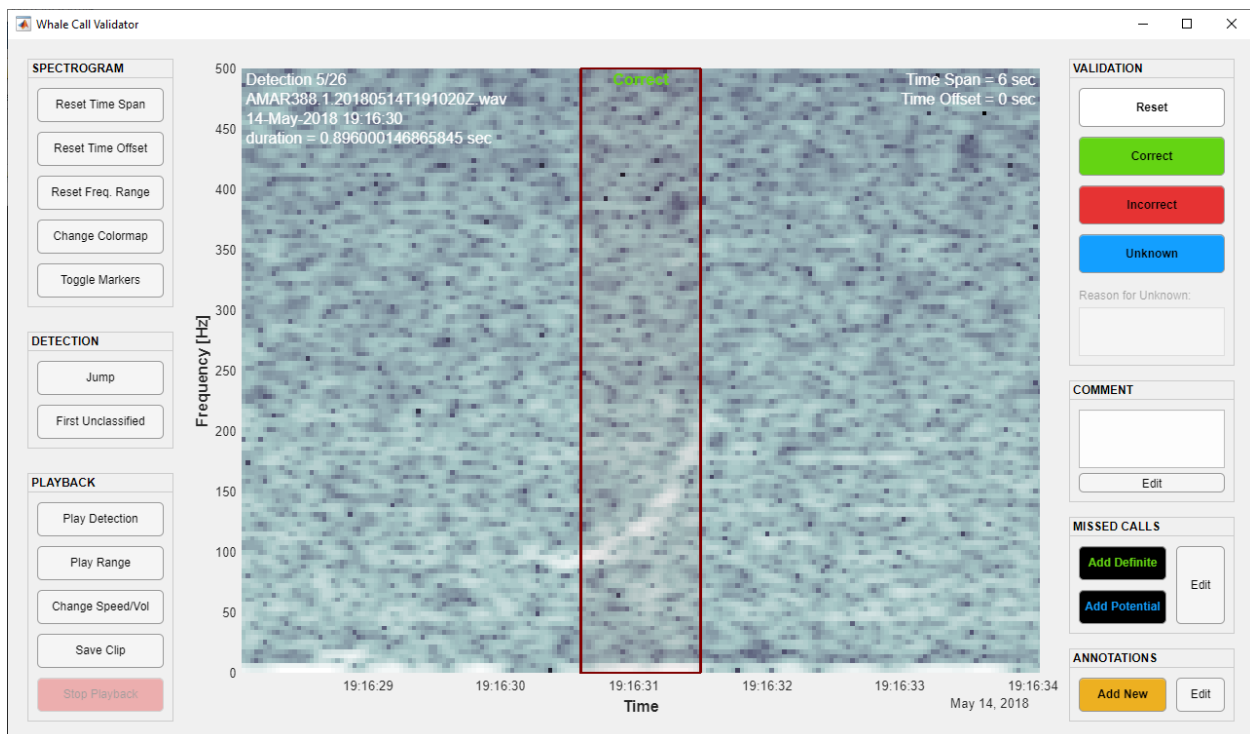
- **LineWidth** – Defines the normal thickness of the outline of the boxes marking detections or annotations.
- **FaceAlpha** – Defines the transparency of the inner face of the boxes marking detections or annotations. Must be between or equal to 0 and 1, where 0 is fully transparent (i.e., outline only) and 1 is fully opaque.
- **StandardFRange** – Frequency limits of the boxes marking detections/target calls. Measured in Hertz. Put this in brackets with the two values separated by a comma.
- **Color_DetectedFocal** – Colour of the box marking the detection in focus, specified as [R, G, B] where each RGB value is a number between 0 and 1.
- **Color_DetectedOther** – Colour of the boxes marking detections not in focus, specified as [R, G, B] where each RGB value is a number between 0 and 1.
- **Color_MissedDefinite** – Colour of the boxes marking definite missed calls, specified as an [R, G, B] vector or the string *auto*. If *auto* is specified, the colour will be the same as the button for classifying Correct calls.

- **Color_MissedPotential** – Colour of the boxes marking potential missed calls, specified as an [R, G, B] vector or the string *auto*. If *auto* is specified, the colour will be the same as the button for classifying Unknown calls.
- **Color_Annotations** – Colour of the boxes marking general annotations, specified as an [R, G, B] vector or the string *auto*. If *auto* is specified, the colour will be the same as the Annotations button.

5.2. Using the BAIT_validate GUI

BAIT_validate works similarly to the LFDCS browser: it focuses on one detection at a time and offers a number of options for validation and other tasks. Unlike LFDCS, it is possible to move forward and backward through the time series around a focal detection, and it is possible to manually mark non-detection areas of the spectrogram (albeit only within audio files that contain detections). Many options are available via GUI buttons similar to LFDCS, but navigating through the plot and stepping through detections is generally accomplished via keyboard inputs.

Here is what the main window looks like:



The user-assigned classification of the detection in focus always appears at the top centre and is colour-coded. The boxes marking signals on the spectrogram come in different colours which mean different things. Default marker colours can be changed using parameter files (see [Marker Parameters in section 5.1.3 Parameters](#)). All marker boxes that appear when processing a new dataset correspond to autodetections. Users may also draw additional boxes to mark either missed calls or general signals of interest (the latter are referred to as “annotations”).

Boxes that mark detections and missed calls currently all have fixed frequency limits – they do not represent the actual minimum and maximum frequencies of each signal within them. This feature is mostly left over from the early days of the tool, when the only purpose was to validate North Atlantic right whale upcall detections, and frequency information was not considered to be of great importance. This could change in a future version of BAIT. Note that it is possible to change the universal frequency limits of the detection and missed call boxes via the *StandardFRange* parameter (see [Marker Parameters in section 5.1.3 Parameters](#)). Marker boxes for general annotations are different, as they do support arbitrary frequency ranges that are unique to each signal.

Each button and keyboard action for the GUI is described below.



The figure window must be in focus for the keyboard inputs to register. If you find that your key presses are not doing anything, try clicking in the figure area. Note also that keyboard navigation is not available in certain situations, like during audio playback or when marking undetected calls.

5.2.1 Keyboard Inputs

Most of the following operations are exclusive to the keyboard, though a few are available via GUI buttons as well.

- **W** – Zoom in along the time axis (decrease time span)
- **S** – Zoom out along the time axis (increase time span)
- **A** – Go to previous detection
- **D** – Go to next detection
- **Shift+W/UpArrow** – Increase upper frequency limit
- **Shift+S/DownArrow** – Decrease upper frequency limit
- **Shift+A/LeftArrow** – Pan left. You can pan up to the beginning of the audio file in which the focal detection is located. Beyond that is nothing but the Void...
- **Shift+D/RightArrow** – Pan right. You can pan up to the end of the audio file in which the focal detection is located. Beyond that is nothing but the Void...
- **R** – Reset time span, time offset, and upper frequency limit to their defaults
- **T** – Toggle signal marker boxes
- **C** – Change color map of the spectrogram. Available colormaps are described in the appendix [LIST OF SPECTROGRAM COLORMAPS](#).
- **X/Delete** – Delete markers for missed calls or annotations. Only works when using “Edit Mode” for those respective signal types.
- **Q** – Close the application. You can also use the X button to do this. Either way, the program will attempt to save any changes before closing.

- **Esc** – Cancel the following operations:
 - marking/unmarking of undetected calls or annotations
 - editing of undetected call or annotation markers
 - audio playback
- **Enter/Return** – Exit editing of missed call or annotation markers

5.2.2 Button Inputs

SPECTROGRAM panel

- **Reset Time Span** – Reset time zoom to its default
- **Reset Time Offset** – Reset time translation to its default
- **Reset Freq. Range** – Reset frequency range to its default
- **Change Colormap** – Change color map of the spectrogram. Available colormaps are described in the appendix [LIST OF SPECTROGRAM COLORMAPS](#).
- **Toggle Markers** – Activate/deactivate all boxes marking auto-detected or manually marked calls

DETECTION panel

- **Jump** – Go to a specific detection
- **First Unclassified** – Go to the first detection in the list that has yet to be classified

PLAYBACK panel

- **Play Detection** – Play back audio of the target detection area
- **Play Range** – Play back audio over an arbitrary range of your choosing. Specify the range by drawing a temporary box over an area of the spectrogram.
- **Change Speed/Vol** – Change playback speed and volume
- **Stop Playback** – Stop a currently playing clip. This button is active only when audio is playing.

VALIDATION panel

- **Reset** – Clear classification for current detection, including “Reason for Unknown” if there is one
- **Correct** – Label detection as a true call
- **Incorrect** – Label detection as a false positive
- **Unknown** – Label detection as uncertain. You will be prompted to specify a reason why.

COMMENT panel

- **Edit** – Enter or edit comment for current detection

MISSED CALLS panel

- **Add Definite** – Mark a definite undetected call if you see one. After this button is pressed, you can draw a window over an area of the spectrogram to mark the start/end times of the call. You will also be prompted to enter a comment. Pressing “OK” after the comment will add your selection to the list of missed calls (sheet 2 in the output file).
- **Add Potential** – Mark a potential undetected call if you see one. This works the same way as for “Definite” calls, but will be labeled as “Potential” in the spreadsheet.
- **Edit** – Enter “Edit Mode” for missed calls. In this mode, you can click on missed call boxes and do the following:
 - Resize boxes by clicking on and dragging their edges (draggable edges will highlight when you hover the cursor over them). For missed calls, you can only edit the left and right edges.
 - Delete the box by pressing the **Delete** or **X** key. This will remove the entry from the spreadsheet.

ANNOTATIONS panel

- **Add New** – Mark an arbitrary area on the spectrogram to indicate something of interest. This works similarly to the *Missed Calls* buttons in that you will be prompted to draw a box, but differs in that you can set the frequency limits as well. In other words, you can set all four edges of an annotation box to any position you want, unlike the *Missed Calls* boxes which have fixed upper and lower limits.
- **Edit** – Enter “Edit Mode” for annotations. This works similarly to the edit mode for missed calls – you can click on any annotation box and do the following:
 - Resize boxes by clicking on and dragging their edges (draggable edges will highlight when you hover the cursor over them). You can resize all four edges for annotations.
 - Delete the box by pressing the **Delete** or **X** key. This will remove the entry from the spreadsheet.

6. IMPORTING LFDCS AUTODETECTION CSV TABLES (`BAIT_convertLFDCSTable`)

The script `BAIT_convertLFDCSTable` allows you to import an LFDCS CSV file generated using the LFDCS command `export_autodetections`, and convert it into an Excel spreadsheet that can then be passed as input to `BAIT_validate`.



In the past, there have been several cases in which LFDCS autodetection CSV files passed to prior versions of BAIT contained imprecise detection times, due to those times being inadvertently rounded to the second. The culprit behind this rounding was eventually found to be Microsoft Excel.

If you need to open CSV files using Excel, be extremely cautious. **NEVER press “Save” when viewing a CSV file produced by LFDCS (or other programs) using Microsoft Excel.** Excel is a bully when it comes to formats other than XLSX and is capable of altering data in unexpected ways, even if you have not made any changes. In this case, it was found that Excel was saving data within CSV files based on its *display* format, which did not include milliseconds for the detection times by default. As a result, merely opening an LFDCS CSV file in Excel, doing nothing, and pressing *Save* would permanently strip away the milliseconds.

BAIT has since been updated to detect if a file has been corrupted in this way. If the problem is detected, BAIT will generate a warning discouraging you from using the file. It is still possible to use these files, but doing so is not recommended. Files with imprecise detection times will lead to bigger data analysis problems later on.

It is possible to run the script `BAIT_convertLFDCSTable` with no input arguments, in which case you will be prompted to specify certain files and folders interactively. However, the script does support various name-value pair arguments, as well as parameters specified within a parameter file. These are described in the following subsections.

6.1 Command-Line Arguments for `BAIT_convertLFDCSTable`

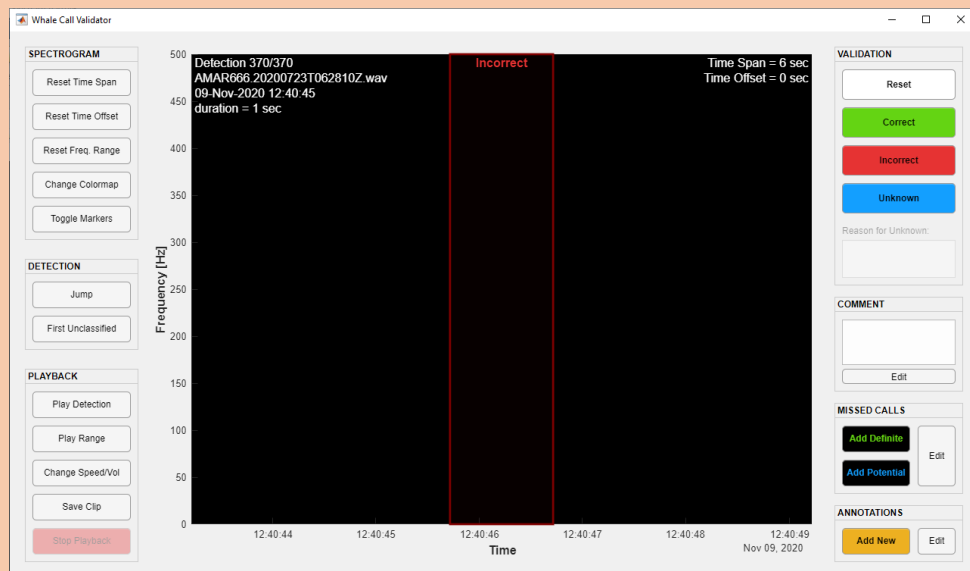
These input arguments must be specified as name-value pairs, following the syntax explained in section [4.2 Running Scripts with Input Arguments](#).

- `'input_file'` – This specifies the path to a LFDCS autodetections CSV file to be converted to the XLSX format. If not specified in the command line, the user will be prompted to select a file.
- `'wav_dir'` – Specifies the path to the folder containing the audio files (WAV files) from which the LFDCS detections were generated. This is used to associate audio files with each detection, which is required information for the `BAIT_validate` script. If not specified in the command line, the user will be prompted to select a folder.



Audio files currently **MUST** have a timestamp embedded somewhere in the filename with the format `YYYYMMDD.hhmmss`, where `.` is any non-digit character. The conversion script will not work if it cannot find timestamps.

`BAIT_convertLFDCSTable` determines which audio files each detection originates from by comparing the start times of each detection to the start times of each audio file. However, it currently does not evaluate the end time of the audio files. Therefore, it is very important that there be **NO MISSING AUDIO FILES**, especially in the middle or at the end of the sequence. If there are, then some detections may end up with the wrong file being assigned to them. In the validation app, this typically manifests as a detection marker appearing against a black background:



- `'output_file'` – Specifies the path (including filename) for the XLSX file that will be created. If not specified in the command line, the user will be prompted to create the file interactively.
- `'wav_subfolders'` – Boolean value (i.e., `true` or `false`) that determines whether subfolders nested within the root audio folder should be searched or not. Use this if your audio dataset is divided into multiple folders; disable it if you have subfolders that contain irrelevant data. The default is `true`.



BAIT will not work as expected if the audio time series is not monotonic (i.e., if there are audio files that overlap in time or appear out of order). Be especially careful of this if your audio files are organized into subfolders, because the order of the subfolders is important. All audio files must be able to appear in a single list from earliest to latest.

6.2 Parameters for BAIT_convertLFDCSTable

Parameters for the script `BAIT_convertLFDCSTable` control which detections from an LFDCS autodetections CSV file should be included in the XLSX file. These are specified in a parameter file as discussed in section [4.3 Parameter Files](#). The folder for parameter files used by `BAIT_convertLFDCSTable` is `PARAMS\convertLFDCSTable`. Specify the use of a custom parameter file by using the '`params`' input argument.

The parameters for `BAIT_convertLFDCSTable` are described below.

Species and Call Type Codes

- **ManualSpeciesCodes** – This consists of an array of numbers corresponding to manually-validated LFDCS species codes, assuming that detections were pre-validated using LFDCS. Only detections that contain codes included in this array will be included in the XLSX spreadsheet. Currently, only the following codes are supported:
 - 9999 (Correct)
 - -9999 (Incorrect)
 - 0 (Unknown)
 - -32767 (Unclassified)This parameter can be used, for example, to omit all detections that were classified as incorrect. The default is to include all detections.
- **AutoCallType** – This consists of an array of numbers corresponding to call type codes that LFDCS has automatically assigned to detections. Only detections containing one of the call type codes included in this array will be included in the XLSX spreadsheet. The default value for this parameter is `[5, 6, 7, 8, 9]`, which corresponds to all North Atlantic right whale upcall call types. Edit this parameter if you wish to analyze other call types, such as those produced by other species.

Date and Time Range

- **StartDateTime** – Corresponds to the starting date-time, i.e., the time from which detections should be included in the XLSX spreadsheet. Any detection that occurs before this time will be omitted. The format for this parameter is *[YYYY, MM, DD, hh, mm, ss]*. It is also possible to specify *NaN*, which effectively disables this parameter and includes all detections (this is the default).
- **StopDateTime** – Corresponds to the ending date-time, i.e., the time up to which detections should be included in the XLSX spreadsheet. Any detection that occurs after this time will be omitted. The format for this parameter is *[YYYY, MM, DD, hh, mm, ss]*. It is also possible to specify *NaN*, which effectively disables this parameter and includes all detections (this is the default).

7. ISOLATING AND PREVIEWING LFDCS DETECTIONS (BAIT_isolateLFDCSDetections)

The script `BAIT_isolateLFDCSDetections` may be used for previewing LFDCS autodetections as spectrograms and/or creating short audio clip files that capture these detections. The process is referred to as *isolation*, because it relies on the original audio files and *isolates* the regions within those files that contain detections.



`BAIT_isolateLFDCSDetections` attempts to keep detections centred within clips and spectrogram images, but this may not always be the case in the final output. It is common for there to be an offset between LFDCS detection times and the actual times that detections occur within the WAV files. This seems to originate from LFDCS: it has been shown that LFDCS will sometimes add small delays within an audio time series that may change over time. The reason for this is not fully understood – it may be a bug in LFDCS or IDL that has not been resolved.

This issue is separate from the Microsoft Excel precision loss issues mentioned in section [6. IMPORTING LFDCS AUTODETECTION CSV TABLES](#). However, the two issues can co-exist and exacerbate the problem.

It is possible to run the script `BAIT_isolateLFDCSDetections` with no input arguments, in which case you will be prompted to specify certain files and folders interactively. However, the script does support various name-value pair arguments, as well as parameters specified within a parameter file. These are described in the following subsections.

7.1 Command-Line Arguments for `BAIT_isolateLFDCSDetections`

These input arguments must be specified as name-value pairs, following the syntax explained in section [4.2 Running Scripts with Input Arguments](#).

- `'input_file'` – This specifies the path to a LFDCS autodetections CSV file listing the detections to isolate. If not specified in the command line, the user will be prompted to select a file.
- `'audio_dir'` – Specifies the path to the folder containing the audio files (WAV files) from which the LFDCS detections were generated. If not specified in the command line, the user will be prompted to select a folder.



Audio files currently **MUST** have a timestamp embedded somewhere in the filename with the format `YYYYMMDD.hhmmss`, where `.` is any non-digit character. The isolation script will not work if it cannot find timestamps.

- `'output_dir'` – Specifies the path to a root folder within which output will be saved. The actual output files will be written to subfolders within the `output_dir`, called “spectrograms” (for

spectrogram image files) and “clips” (for audio clip files). If not specified in the command line, the user will be prompted to choose a folder interactively.

- **'overwrite'** – Boolean value (i.e., **true** or **false**) that determines whether clip or image files that already exist with the same names should be overwritten or not. The default is **false**.

7.2 Parameters for BAIT_isolateLFDCSDetections

Parameters for the script BAIT_isolateLFDCSDetections control which types of files are generated and how. These are specified in a parameter file as discussed in section [4.3 Parameter Files](#). The folder for parameter files used by BAIT_isolateLFDCSDetections is `PARAMS\isolateLFDCSDetections`. Specify the use of a custom parameter file by using the **'params'** input argument.

The parameters for BAIT_isolateLFDCSDetections are described below.

Input Processing

- **Channel** – The audio file channel number to use. Must be less than or equal to the number of channels in the recording. The default is 1.



Unlike BAIT_validate, there is currently no option to choose the channel interactively in BAIT_isolateLFDCSDetections. To use a channel other than the first, this needs to be specified in the parameter file.

- **RecursiveSearch** – Boolean value (i.e., **true** or **false**) that determines whether subfolders nested within the root audio folder should be searched or not. Use this if your audio dataset is divided into multiple folders; disable it if you have subfolders that contain irrelevant data. The default is **true**.



BAIT will not work as expected if the audio time series is not monotonic (i.e., if there are audio files that overlap in time or appear out of order). Be especially careful of this if your audio files are organized into subfolders, because the order of the subfolders is important. All audio files must be able to appear in a single list from earliest to latest.

Output File Types

- **SaveClips** – Boolean value (i.e., **true** or **false**) that determines if audio clips of the detections should be saved or not (WAV file format). Default is **true**.
- **SaveSpecs** – Boolean value (i.e., **true** or **false**) that determines if spectrogram images of the detections should be saved or not (PNG file format). Default is **true**.



If *SaveClips* is `false` but *SaveSpecs* is `true`, the script will start by searching the output folder to check if there happen to be any existing clips. If clips exist, they will be used to create the spectrograms; in this case, no audio folder needs to be specified, and the user will not be prompted for one. However, if clips are not found, then an input audio folder is required.

General Output Properties

- **SnippetDur** – Number that determines the time span (in seconds) of output audio clips and/or spectrograms. The default is 4, which means that each clip and/or spectrogram will span 4 seconds in total, centred about their detections.



Be careful not to make *SnippetDur* too small, as this can result in detections being clipped, if not entirely absent (due to the complications with centring detections mentioned previously in this section).

Spectrogram Image Properties

These are only used if spectrogram images are being generated.

- **SpecMaxFreq** – Number specifying the upper frequency limit of spectrograms, in Hertz. Change this in accordance with call types that you expect to see. The default is 400, which is optimized for North Atlantic right whale upcalls. Note that it is currently not possible to adjust the minimum frequency (that is always locked at 0 Hz).
- **SpecColorMap** – String specifying the spectrogram colormap to use. The viable options include all those listed in the appendix [LIST OF SPECTROGRAM COLORMAPS](#). The default is *parula*.
- **SpecFigSize** – Array of two numbers corresponding to the width and height of output spectrogram image files, in pixels. The default is [960, 540]. Note that the image resolution is currently locked at 150 DPI (Dots Per Inch).

8. REFERENCES

- Baumgartner, M.F., and Mussoline, S.E. **(2011)**. "A generalized baleen whale call detection and classification system," *J. Acoust. Soc. Am.* **129**, 2889-2902
- Kirsebom, O.S., Frazao, F., Simard, Y., Roy, N., Matwin, S., and Giard, S. **(2020)**. "Performance of a deep neural network at detecting North Atlantic right whale upcalls," *J. Acoust. Soc. Am.* **147** (4), 2636–2646

APPENDIX: LIST OF SPECTROGRAM COLORMAPS









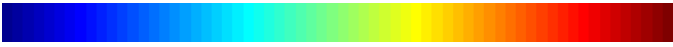
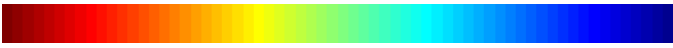

Spectrogram colormaps supported by BAIT consist largely of all continuous colormaps built-in to MATLAB, with the exception of the `blue` colormap, which was adapted from the Scripps *Triton* project (<https://github.com/MarineBioAcousticsRC/Triton>). It is also possible to use inverted versions of each colormap that reverse the order in which colours are associated with intensity. Each colormap is referenced by a unique name, in accordance with the table below.



You may be wondering why the American spelling of *Color* is used when referring to colormaps, whereas the Canadian spelling is used otherwise. This is because MATLAB, being the product of an American company, requires the American spelling in all code that deals with colormaps. There is no `colourmap` command in MATLAB; trying to use that will generate an error.

Colormap Name	Color Scale
parula	
parula_inverted	
hsv	
hsv_inverted	
hot	
hot_inverted	
cool	
cool_inverted	
spring	
spring_inverted	
summer	
summer_inverted	
autumn	
autumn_inverted	
winter	
winter_inverted	

Colormaps continued

Colormap Name	Color Scale
gray	
gray_inverted	
bone	
bone_inverted	
copper	
copper_inverted	
pink	
pink_inverted	
jet	
jet_inverted	
blue	
blue_inverted	