AnalysisPrograms.exe (AP.exe) Manual

QUT Ecoacoustics

Michael Towsey & Anthony Truskinger

https://ap.qut.ecoacoustics.info/

Version: 21.4.0.94
Git Hash: 2ffce6fe834b2833afb7807c13929d1bf2eb7b45
Build Date: 4/30/2021 12:04:34 AM

Table of Contents

Introduction

QUT Ecoacoustics Analysis Programs is a software package that can perform a suite of analyses on audio recordings of the environment. Although the analyses are intended for long-duration recordings (1 – 24 hours), in fact they can be performed on any audio file in a format supported by the software. Analysis Programs can:

- calculate of summary and spectral acoustic indices at variable resolutions
- produce long-duration, false-colour, multi-index spectrograms
- calculate critical statistics of annotations downloaded from an Acoustic Workbench
- run various acoustic event recognizers

All the analyses are performed by a single executable file, *AnalysisPrograms.exe*.

The project is open source and licensed under the Apache License 2.0.

Quick links

- Come chat on Gitter with us □ chat on gitter
- See these docs for instructions on
  - Downloading AnalysisPrograms.exe
  - Running the program
  - Understanding concepts
- See the **Issues list** for
  - reporting bugs
  - requesting new features
- See the **Contributing guidelines** if you want to
  - Compile the code yourself
  - Make a contribution
- Let us know if you're using AP.exe

Citation

DOI 10.5281/zenodo.4274299

This citation should be used in all publications that use data, concepts, or results generated by AnalysisPrograms.exe or from this code base:

> Michael Towsey, Anthony Truskinger, Mark Cottman-Fields, & Paul Roe. (2018, March 5). Ecoacoustics Audio Analysis Software v18.03.0.41 (Version v18.03.0.41). Zenodo. http://doi.org/10.5281/zenodo.1188744

Additionally, depending on the analysis that was run, extra work may be required to be cited. Any such additional citations will printed in the console and in the log file.

Introduction to AnalysisPrograms.exe

AnalysisPrograms.exe (AP.exe) is a software package that analyses recordings of the the environment. This tool is designed specifically to work with long recordings that are collected by passive acoustic monitors.

The name, *AnalysisPrograms.exe*, is so general because we package a variety of different analyses in one program. AP.exe can generate acoustic indices, visualise indices, run event recognisers, and more!

AP.exe is over 10 years old, has more than 200K lines of code, and has done a lot of analysis on audio data. We estimate we've analysed over 100 years of audio data, totalling more than 200TB of files.

Philosophy

One recording, one analysis

AP.exe works on one recording at a time. This is a good choice for us because it means we do not have to write code or make assumptions on how or where your audio data is stored.

In reality you will want to process many files. Similarly, AP.exe very specifically avoids batch processing of files because we don’t want to make assumptions about how your data is stored or how it should be computed. This is important in universities that use PBS (Portable Batch System) compute clusters---assumptions in how data is processed greatly vary. The solution to batch processing is to script AP.exe. You can script AP.exe with any language you like.

Longer recordings are better

The majority of the recordings we collect are long: from 30 minutes, to 2 hours (our median), to ~6.78 hours (WAVE file limit at 22050Hz, 16-bit, stereo), to 24 hour recordings!

Analysing all the data at once is in efficient and requires powerful computers. Thus AP.exe breaks up long recordings into smaller chunks---typically one minute in duration---and results are extracted from each chunk.

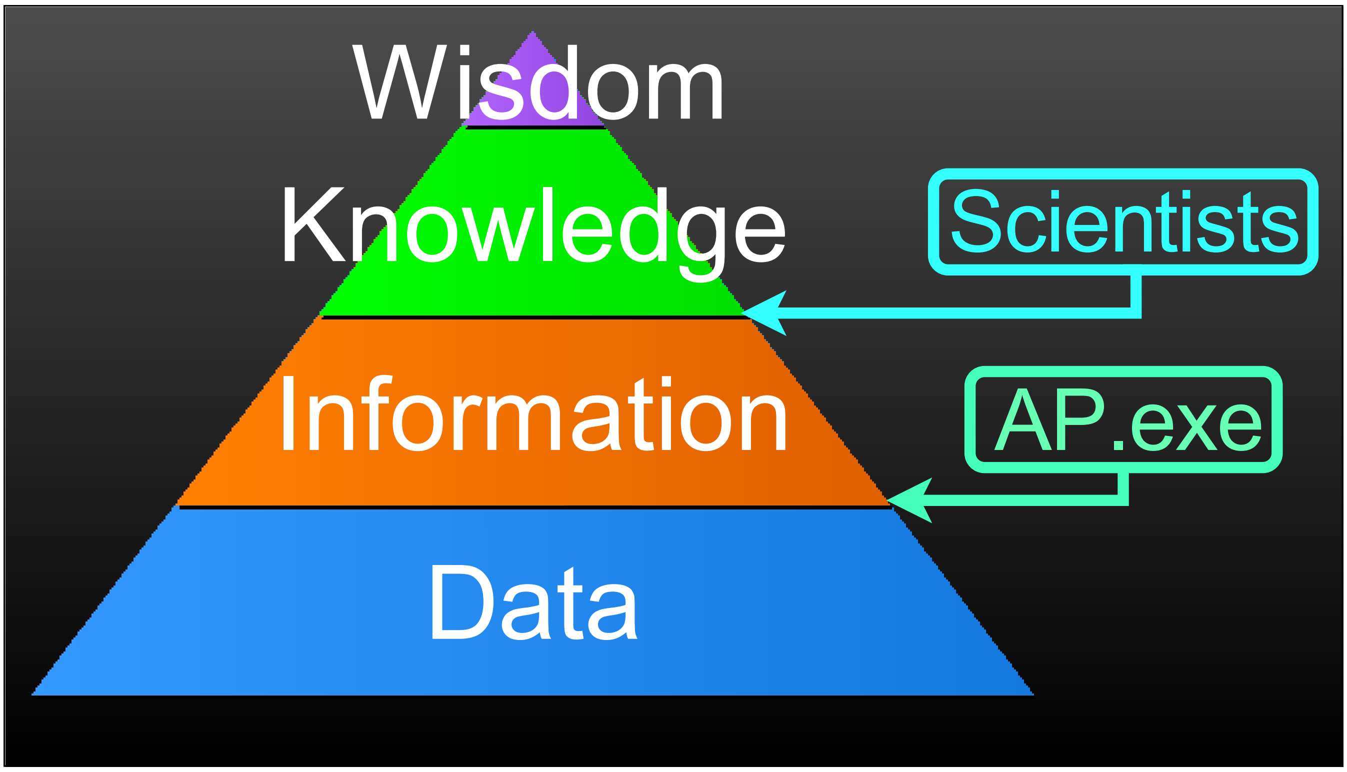
AP.exe is very good at this and can even parallelise the processing of these chunks to increase analysis speed.

Constantly changing

AP.exe is a research product and as such changes nearly weekly. You can find new releases in the releases tab on GitHub (see installing).

We're in the data transformation business

The role of AP.exe is transform raw audio data into more useful information. AP.exe will however never attempt to make ecological or scientific inferences. The data produced will almost always need post-processing, whether that be by scripted analysis or manual review.



Caveats

Focused and narrow

AP.exe is designed to do automated, unassisted, analysis of audio at massive scales. It answers the questions we need answered.

It is not a product (like SongScope, Kaleidoscope, or SoundID) where you can build or customize recognisers as an end user.

Neither is it a library (like Seewave, warbleR, monitorR) because you can't pick and choose functions to stitch together to make something new.

Made for machines

AP.exe is made for machines to use. It is not user friendly, and has no graphical user interface. This limitation is an important and necessary constraint for AP.exe as it forces the tool to remain focused.

No support

We also officially provide no support, guarantee, or warranty for AP.exe. We have released AP.exe so that the community may benefit from our work but we do not have the resources to treat AP.exe as a fully fledged product.

Having said that, we're usually interested in fixing bugs, helping people, or adding features---so please contact us!

Old code, research code

Because of the age of the code, there are many bugs, a lot of old or unmaintained code, and a rich, complex, history of changes. There be dragons.

Installing

If you're new to using *AP.exe* we recommend following the instructions in the Using AnalysisPrograms.exe (Practical) practical.

You can choose to use our automated installer script or do a manual install

Supported Platforms

- Any of the following platforms:
  - Windows 7 or newer
  - Mac OSX 10.9 or later
  - Any Debian based linux
  - Any Debian based linux container
- As well as the following CPu architectures
  - Intel x86 (64-bit only)
  - ARM (32 bit and 64-bit)

Automatic install

The automatic install will download AP.exe and may install required perquisites.

1. Prerequisite: install Powershell 7+
   - Go to https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell to find instructions for installing PowerShell
2. Then:

- Windows install
- Linux install
- MacOSX install

Run the following command in an elevated (*Run as Administrator*) prompt:

```
pwsh -nop -ex B -c '$function:i=irm "https://git.io/JtOo3";i'
```

Or, to install the prerelease version:

```
pwsh -nop -ex B -c '$function:i=irm "https://git.io/JtOo3";i -Pre'
```

NOTE

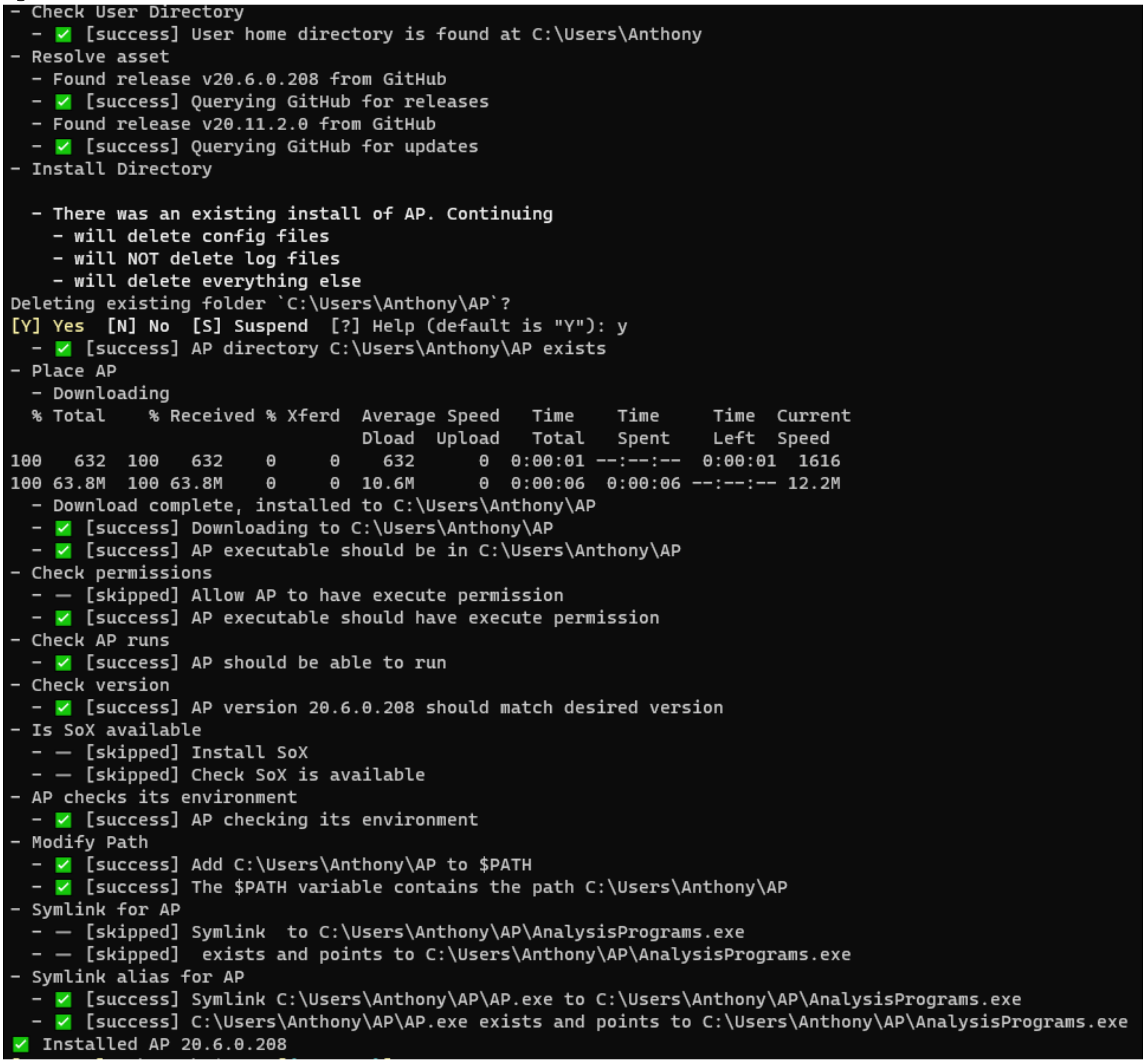
Please inspect https://git.io/JtOo3 which should point to https://github.com/QutEcoacoustics/audio-analysis/blob/master/build/download_ap.ps1 prior to running these commands.

We already know the script is safe, but you should verify the security and contents of any script from the internet you are not familiar with. The above command downloads a remote PowerShell script and executes it on your machine.

WARNING

The installer script is brand new. There may be bugs. No warranties provided.

1. The script should install or upgrade AP.exe.
   - If it is upgrading it will ask you if you want to overwrite the old installation.
     - Choose *yes* unless you have NOT stored data files or config files in the AP folder
     - Choose *no* if you have stored data files or config files in the AP folder. Copy out your files and then try installing again.



```
- Check User Directory
  - ✅ [success] User home directory is found at C:\Users\Anthony
- Resolve asset
  - Found release v20.6.0.208 from GitHub
  - ✅ [success] Querying GitHub for releases
  - Found release v20.11.2.0 from GitHub
  - ✅ [success] Querying GitHub for updates
- Install Directory

  - There was an existing install of AP. Continuing
    - will delete config files
    - will NOT delete log files
    - will delete everything else
Deleting existing folder `C:\Users\Anthony\AP`?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y
  - ✅ [success] AP directory C:\Users\Anthony\AP exists
- Place AP
  - Downloading
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   632  100   632    0     0    632      0  0:00:01 --:--:--  0:00:01  1616
100 63.8M  100 63.8M    0     0  10.6M      0  0:00:06  0:00:06 --:--:-- 12.2M
  - Download complete, installed to C:\Users\Anthony\AP
  - ✅ [success] Downloading to C:\Users\Anthony\AP
  - ✅ [success] AP executable should be in C:\Users\Anthony\AP
- Check permissions
  - — [skipped] Allow AP to have execute permission
  - ✅ [success] AP executable should have execute permission
- Check AP runs
  - ✅ [success] AP should be able to run
- Check version
  - ✅ [success] AP version 20.6.0.208 should match desired version
- Is SoX available
  - — [skipped] Install SoX
  - — [skipped] Check SoX is available
- AP checks its environment
  - ✅ [success] AP checking its environment
- Modify Path
  - ✅ [success] Add C:\Users\Anthony\AP to $PATH
  - ✅ [success] The $PATH variable contains the path C:\Users\Anthony\AP
- Symlink for AP
  - — [skipped] Symlink  to C:\Users\Anthony\AP\AnalysisPrograms.exe
  - — [skipped]  exists and points to C:\Users\Anthony\AP\AnalysisPrograms.exe
- Symlink alias for AP
  - ✅ [success] Symlink C:\Users\Anthony\AP\AP.exe to C:\Users\Anthony\AP\AnalysisPrograms.exe
  - ✅ [success] C:\Users\Anthony\AP\AP.exe exists and points to C:\Users\Anthony\AP\AnalysisPrograms.exe
✅ Installed AP 20.6.0.208
```

2. If everything went well AP should be ready to go. Try running a command:

```
AP --version
```

TIP

AP is an alias for_AnalysisPrograms.exe_ that we made to make AP easier to use.

The alias does exactly the same thing as calling *AnalysisPrograms.exe* by its full name. To learn more about how it works see Adding to PATH.

Uninstall

If you used our automatic install you can use the same script to uninstall:

- Windows uninstall
- Linux uninstall
- MacOSX uninstall

Run the following command in an elevated (*Run as Administrator*) prompt:

```
pwsh -nop -ex B -c '$function:i=irm "https://git.io/JtOo3";i -Un'
```

Manual Install

1. Go to our releases page
2. Select the version you want to download
    - Choose the *Latest release* unless you specifically need features that are available in a *Pre-release*
3. Scroll down to the assets section for your chosen release
4. Download the version of AnalysisPrograms suitable for your computer (see Choosing the asset)
5. Extract the folder
    - It can be installed in any directory
    - We typically extract to a directory named `~\AP` or `~/.local/share/AP` on Linux
6. Make sure any Prerequisites are installed
7. [Optional] Add the install directory to your `PATH` environment variable
    - Instructions in the Path document.
8. Finally, check the install by running:

Run the following command:

- Windows Check
- Linux Check
- MacOSX Check

```
C:\AP\AnalysisPrograms.exe CheckEnvironment
```

Choosing the asset

What operating system do you use?

Common choices:

[✓ Windows]
Linux
Mac OSX

Expert choices:

Linux (Alpine)

(choose for a container)

Any

(choose for a run anywhere version, requires a dotnet core runtime install)

What is your CPU architecture?

[✓ Intel x86-64 (64-bit)]

This is a good default choice; nearly all computers, laptops, and servers use x86

Arm (32-bit)

Choose this if you're using a Raspberry Pi

Arm (64-bit)

Choose this if you're using a Raspberry Pi with 64-bit Raspbian or any other ARM device

NOTE

We do not support 32-bit operating systems or CPU architectures, except for Arm devices.

Do you need to debug this?

Yes

[✓ No]

NOTE

For expert use only. A debug version will run **slower**.

Asset name

Enable JavaScript to view result. Not available in PDF manual.

Warning

A .NET Core runtime must be installed and AnalysisPrograms must be invoked with a `dotnet` prefix.

Prerequisites

Windows

None. Self contained download.

MacOSX

None. Self contained download.

Linux/Unix

The following additional dependencies may be required for Linux/Unix machines:

- **MAYBE**: ffmpeg
  - a packaged version with AP.exe should work for all platforms except ARM and ARM64
- **MAYBE**: wavpack
- libsox-fmt-all, sox
- libav-tools (on some distros only, not needed in Ubuntu 18)

Build Packages

There are two variants of AP.exe:

1. The **Stable** release is well tested used by QUT Ecoacoustics on our servers and is usually a few months old
2. The **Prerelease** release is automatically built weekly, every Monday. It has more features and bug fixes than the stable release but it also could have more bugs.

You should use the **Stable** release unless there is a recent feature implemented or bug fix in the prerelease version that you need.

Adding AP.exe to `PATH`

The `PATH` variable is a system wide variable (known as en environment variable) that any program can access. The `PATH` variable contains a list of folders that your computer can check when it searches for programs to run.

When AP's folder is added to `PATH` you or any program on your computer can run *AnalysisPrograms.exe* without knowing where AP is actually installed.

So instead of needing this:

```
> C:\Users\Anthony\AP\AnalysisPrograms.exe
```

You can instead write:

```
> AnalysisPrograms.exe
```

Setup using the installer script

The automatic installer will automatically add AP to `PATH` for you.

If you don't want this to happen use the `-DontAddToPath` switch when installing.

Add to `PATH` manually

- Windows
- Linux
- MacOSX

1. Find where AP is installed on your computer. This will be a directory (folder) where *AnalysisPrograms.exe* resides.
2. Open your *System Environment Variables*. You can type some of *Edit environment variables for your account* in the Start Menu search box to find the settings.
3. Choose *Environment Variables...* in the windows that popped up
4. In the *user variables* section, find the *Path* variable, select it, and then hit the *Edit* button
5. Add the directory from step 1 to the end
    - Ensure a semi-colon (`;`) delimits the new directory from the previous ones, if you're using an older version of Windows
6. Then click *OK* or close all windows.
7. You will have to restart any programs for which you want to see the new value

Aliasing *AnalysisPrograms.exe* to *AP*

AnalysisPrograms.exe is a long name. Tiring to type, prone to errors. It also isn't a good name for a cross-platform program; on Linux and Mac OS it is simply *AnalysisPrograms*.

To make it easier for people to experiment with and use *AP* we aliased (gave another name) to *AnalysisPrograms.exe*. We chose `AP`.

So instead of needing this:

```
> C:\Users\Anthony\AP\AnalysisPrograms.exe
```

You can instead write (assuming *AnalysisPrograms.exe* is on PATH):

```
> AP
```

Setup an alias using the installer script

If you have installed AP using the automatic installer then this alias has already been set up for you!

Setup an alias manually

This is advanced content.

1. Find the *AP* installation directory
2. Create a symbolic link
   - Windows: `<INSTALL-DIR>\AP.exe` and `<INSTALL-DIR>\_AnalysisPrograms.exe`
   - Linux/Mac: `<INSTALL-DIR>/AP` and `<INSTALL-DIR>/_AnalysisPrograms`

You'll need to ensure `<INSTALL-DIR>` in on `PATH`.

Introduction To Running Commands

You need a shell

On Windows we recommend using a PowerShell prompt, but CMD will also work. On Unix systems any terminal, like BaSH will work.

Example syntax

For examples of how to call the executable from the command line, we use the `$` symbol to denote a command line prompt. Do not type the `$` symbol yourself.

On Windows machines the standard prompt is a `>` character. We use a `$` sign in these examples only for consistency.

Special mentions:

- Powershell & Bash
    - All executables must be prefixed with a `./` if you're in the same folder as the executable

Your first execution

In your shell, type the following and press `Enter`.

```
$ AnalysisPrograms.exe
```

This will produce several lines of output, including the program version number and instructions on how to set debug levels and verbosity. These options have sensible defaults and you do not need to change them.

Seeing what *AP.exe* can do

```
$ AnalysisPrograms.exe list
```

Will print a list of the available *commands*. Every command has an command-name by which it is invoked. Some commands perform acoustic analyses.

When constructing a command line, the first argument after the executable file name must be the command name, which will typically be followed by a list of options for that command:

```
$ AnalysisPrograms <command-name> [options...]
```

To obtain help with the options of a particular command, type:

```
$ AnalysisPrograms.exe help <command-name>
```

For example:

```
$ AnalysisPrograms.exe help audio2csv
```

In this command line, ‘help’ is a *command* and the command-name, *audio2csv*, is an option.

Options

The command line options (all prefixed with a hyphen (`-`)), have a short and long form. The short form (`-x`) is always shown to the left of the long form (`--long-option`) and using either is equivalent. The value for the option should follow the option name, separated by a space. Use double quotes to group values together.

We use long form options in this manual for clarity. See output from the `help` command for all options.

Global options

Here is a short description of some of global options. As mentioned previously, you can ignore these options until you need them.

Console and Log Verbosity

Verbosity of the logging output can be set by appending the `loglevel` options to the command line:

Valid verbosity values are:

- `None` = 0 - show nothing
- `Error` = 1 - show only errors
- `Warn` = 2 - show only warnings
- `Info` = 3 - the standard level
- `Debug` = 4 - print some debug statements that show variable state and extra information
- `Trace` = 5 - print many more debugging statements with detailed variable values
- `Verbose` = 6 - print all stdout and stderr from associated tools
- `All` = 7 - print absolutely everything

For example: `-l 4`, will give you the `debug` level of verbosity.

Alternatively, you can append one of the following switches to the command line:

- `-v` Sets the logging to verbose. Equivalent to LogLevel = `Debug` = 4
- `-vv` Sets the logging to very verbose. Equivalent to LogLevel = `Trace` = 5
- `-vvv` Sets the logging to extremely verbose. Equivalent to LogLevel = `All` = 7

Environment variables

- `AP_PLAIN_LOGGING`: `[true|false]`-- Enables simpler logging and no color output--the default is value is `false`
- [NOT YET IMPLEMENTED] `AP_DISABLE_METRICS` -- if defined will not send performance metrics back to the developers

Beware these Syntax Gotchas**

- **Never** finish a double quoted string argument with a backslash (\). In particular, do *not* end directory names like this: “C:\\Path\OutputDirectory\”. The parsing rules for such cases are complicated and outside of our control. See here for details.
- You can test arguments on Windows with the `echoargs.EXE` program
- The arguments used are one of the first lines logged in *AP.exe* log file
- If an input argument is an array (e.g. directoryinfo[]), any commas in the argument will delimit the values. For example, "Y:\Results\abc, 123, doo-dah-dee" will be parsed as "Y:\Results\abc", " 123", " doo-dah-dee".

Commands

Types

The following four categories of sub-programs (commands) are available:

- Main commands
  - Process large amounts of information (like `audio2csv`)
- Development / small scale commands
  - Small data / development entry points
  - `eventrecognizer` which is a generic program for testing different recognizers
- Utility commands
  - `DummyAnalyser` - uses CPU for a while and does nothing useful
  - `audiocutter` - cuts and converts long audio files up
- Meta commands
  - help and documentation usage (`help` & `list`)
  - `analysesavailable`

Important Commands

Analyze Long Recordings

Processes large audio recording with the specified analysis. Can run recognizers, calculate indices, or do other things, for very long recordings.

See details here Analyze Long Recordings

Colour Spectrogram

This command produces a single false-colour (FC) spectrogram, taking as input the spectral indices produced by the *Acoustic Indices* analysis on a single audio file.

See details here Draw Long Duration Spectrograms

Concatenate Index Files

This command joins together the results of several *Acoustic Indices* analysis result sets to produce data and images for 24-hour blocks of data.

See details here Concatenate Index Files

Config Files

Most commands require a configuration file which gives you access to various parameters whose values change the outcome of the analysis.

Syntax

The config file must be in strict YAML format and should have the file extension `.yml`.

- Comments in the config file give further information. All comment lines must start with a hash symbol `#`
- Be careful with the syntax. Incorrect syntax can lead to errors that are difficult to trace.
- Typically, you will only need to adjust a subset of the available parameters
- Most parameters have default values

You can find a introductions to YAML here:

- https://support.ehelp.edu.au/support/solutions/articles/6000055385-introduction-to-yaml
- https://sweetohm.net/article/introduction-yaml.en.html

You can validate YAML files (to check for syntax errors) here: http://yaml-online-parser.appspot.com/

When editing YAML files follow these rules:

- Use a good editor like Visual Studio Code which will detect mistakes and highlight different parts of the file with different colours for you
- Always indent lines with four (4) spaces (`Space Bar`)
- **Never** use the `Tab ▯` key or tab (`\t`) character to indent lines

Location

All config files are packaged with *AP.exe* releases. Inside the package you will find a `ConfigFiles` folder that contains all the config files.

IMPORTANT: Avoid editing these files directly. If you want to change a value:

1. Copy the file to another directory (a personal folder)
2. Rename the file to describe the changes
    - e.g. `Towsey.Acoustic.yml` might become `Towsey.Acoustic.HighResolution.yml`
    - See the [Naming] section below for naming rules for the config file
3. Change the values in the file
4. Remember to update the path to the config file when you run *AP.exe*

Naming

Since 72aab48 the naming format of the config files is now important. We use the name to determine which analysis to run. For any config file used by `audio2csv`/`AnalyzeLongRecording` the name of the config file must follow this format:

```
<author>.<analysis>[.<tag>]*.yml
```

The `author` and `analysis` name sections are mandatory. The `tag` section is optional, ignored by *AP*, and can be repeated.

Here are some valid examples:

- `Towsey.Acoustic.yml`
- `Towsey.Acoustic.Marine.yml`

- `Towsey.LitoriaFallax.CustomSettings_23.AnotherTag.yml`
- `Truskinger.NinoxBoobook.You.Can.Have.As.ManyDottedTags.As.YouWant.AfterTheFirstTwo.yml`

Here are some **invalid** examples:

- `TowseyAcoustic.yml`
  there's no dot (`.`) between the `Towsey` and `Acoustic` parts
- `Towsey.Acousticyml`
  there's no dot (`.`) between the `Towsey.Acoustic` and `yml` parts
- `Towsey.Acousticmarine.yml`
  AP looks for an analysis called `Towsey.Acousticmarine` which doesn't exist. It should be `Towsey.Acoustic.marine.yml`

If you find a config file that does not match this format, it will likely produce an error. If your config file must be named in a different format the `--analysis-identifier` (or the short form `-a`) argument can be used to disambiguate the analysis type you want to use.

Please note this rule does not apply to other config files not directly used by `audio2csv`. For example, `IndexProperties.yml` needs no particular naming format to valid.

Editing

Basic changes to a config file can be minimal. For example to change the resample rate for an analysis, only the number needs to be changed:

```
-ResampleRate: 22050
+ResampleRate: 16000
```

Note that the parameter name `ResampleRate` is followed by a colon, a space and then a value.

Most of our config files contain comments next to the parameters that explain what a parameter does. A comment is any line that begins with an hash symbol (`#`). You can see the text that is a comment is coloured differently from the parameter in the example below:

```
# SegmentDuration: units=seconds;
# Long duration recordings are cut into short segments for more efficient processing.
# Default segment length = 60 seconds.
# WARNING: You should not change this property!!
SegmentDuration: 60
```

Profiles

The most variable part of a config file is the `Profiles` section. Profiles allow us to add extra sections to an analysis. This can be useful for dealing with:

- Geographical variation in calls. Often a species' call will vary between regions. The same detector can work for the different variants of a call but slightly different parameters are needed. In this case we add a profile for each regional variation of the call that have slightly different parameters or thresholds.
- Generic recognition efforts. Each different type of syllable detection we want to use in a DIY Call Recognizer is added into a different profile. In this way we can detect many different syllable variants and types in a fairly generic manner.

Some analyses do not have a `Profiles` section. For those there's nothing to change.

For config files that do support a `Profiles` section, the format will be as follows:

```
# the word Profiles will always be at the start of the line
Profiles:
    # each profile will have a name
    MyName:
        # Each profile will have some parameters
        SomeParameter: 123
        AnotherParameter: "hello"
    # more than one profile can be added
    #            We use the `!type` notation to tell AP
    #            what type of parameters we're giving it
    KoalaExhale: !OscillationParameters
        ComponentName: Oscillation
        SpeciesName: PhascolarctosCinereus
        FrameSize: 512
        FrameStep: 256
        WindowFunction: HANNING
        BgNoiseThreshold: 0.0
        MinHertz: 250
        MaxHertz: 800
        MinDuration: 0.5
        MaxDuration: 2.5
        DctDuration: 0.30
        DctThreshold: 0.5
        MinOcilFreq: 20
        MaxOcilFreq: 55
        EventThreshold: 0.2
    # And another profile using the blob type
    # (!BlobParameters) parameters
    KoalaInhale: !BlobParameters
        ComponentName: Inhale
        MinHertz: 800
        MaxHertz: 8000
        MinDuration: 0.15
        MaxDuration: 0.8
        DecibelThresholds:
            - 9.0
```

Profiles can get complicated. Each configuration file should detail the different options available. If they don't, then please let us know!

For more information on constructing generic recognizers see DIY Call Recognizer.

Indentation

Whenever a line is indented (add trailing spaces) in a YAML configuration file, it must be indented to a consistent level. If it is inconsistent then *AP* will not be able to read the config file properly.

Here are some examples. Note spaces are represented with a *middle dot* (`·`).

Good

```
Profiles:
····BoobookSyllable1: !ForwardTrackParameters
········MinHertz: 400
········MaxHertz: 1100
········MinDuration: 0.1
········MaxDuration: 0.499
```

Note that each indentation step uses four (4) spaces, and each line in the same level has the same indentation.

Bad: inconsistent indentation

```
Profiles:
····BoobookSyllable1: !ForwardTrackParameters
······MinHertz: 400
········MaxHertz: 1100
········MinDuration: 0.1
········MaxDuration: 0.499
```

Note that the `MinHertz` entry is indented with six (6) spaces instead of eight (8). This means *AP* will try to read `MaxHertz` as a child (a sub-item) of `MinHertz` - which is incorrect because they are siblings belonging to the same item.

Bad: mixing tabs and spaces

Note in this example the `⇥` symbol represents pressing the `Tab ⇥` key.

```
Profiles:
····BoobookSyllable1: !ForwardTrackParameters
⇥⇥⇥⇥⇥MinHertz: 400
········MaxHertz: 1100
········MinDuration: 0.1
········MaxDuration: 0.499
```

Note that the `MinHertz` entry is indented with five (5) tabs instead of eight (8) spaces. Again, this means that `MinHertz` and `MaxHertz` are not part of the same group (they do not have the same indentation), even though it looks like they are aligned.

Supported date and time formats

Date formats:

- "yyyyMMdd[-|T|_]HHmmss" (if timezone offset hint provided),
- "yyyyMMdd[-|T|_]HHmmssZ",
- "yyyyMMdd[-|T|_]HHmmss[+|-]HH",
- "yyyyMMdd[-|T|_]HHmmss[+|-]HHmm",

Examples:

- valid: Prefix_YYYYMMDD_hhmmss.wav, Prefix_YYYYMMDD_hhmmssZ.wav
- valid: prefix_20140101_235959.mp3, a_00000000_000000.a, a_99999999_999999.dnsb48364JSFDSD, prefix_20140101_235959Z.mp3
- valid: SERF_20130314_000021_000.wav, a_20130314_000021_a.a, a_99999999_999999_a.dnsb48364JSFDSD
- valid: prefix_20140101-235959+10.mp3, a_00000000-000000+00.a, a_99999999-999999+9999.dnsb48364JSFDSD
- valid: prefix_20140101_235959+10.mp3, a_00000000_000000+00.a, a_99999999_999999+9999.dnsb48364JSFDSD
- ISO8601-ish (supports a file compatible variant of ISO8601)
- valid: prefix_20140101T235959+10.mp3, a_00000000T000000+00.a, a_99999999T999999+9999.dnsb48364JSFDSD

Problems with dates

- There is one true date format: ISO8601
- Dates and times can be ambiguous
- Time zones are silly
  - Daylight savings (e.g. AEDT)
  - Time zones change regularly
  - E.g. Sudan changed time zone from UTC +3 to UTC +2 on 1 November 2017

- Time zone offsets are very important
  - How many hours from UTC
  - Less problems than time zones

BAD EXAMPLE	PROBLEM	SOLUTION	BETTER EXAMPLE
3:45	Can occur twice a day	Use 24 hour time HH:mm	03:45
3/3/2018	Month can come first	Use ISO8601 dates	2018-03-03
16:30 03/03/2018	Not sortable	Order date components from largest to smallest (ISO8601)	2018-03-03 03:45:00
2018-03-03 03:45:00	Could refer to 37 different points in time today	Always use a time zone offset	2018-03-03 03:45:00+10:00
592D42A3	Unreadable (...but not ambiguous)	Make it readable	20180303T034500Z
2018-03-03 03:45:00+10:00	Invalid filename characters	ISO8601 alternative format	20180303T034500+1000

FAQ

How do I reproduce the results I'm given?

You need four things to reproduce a result set:

- The input data
- The config file
- The command used to run AP.exe
- And the same version of AP.exe

Getting the same input data is up to you.

The config file (and it's values), the command used, and the version of AP.exe are all recorded in the log file. Once you open the log file you can find and extract all of these values. For more information on log files see the document on log files.

Who is this program designed for?

Short answer: computers.

AP.exe is designed primarily to be used by people who have significant computing expertise. It designed specifically to be used by **other programs** for any real workloads. It is not designed to be human friendly.

We encourage anyone to give it a go--don't be daunted by these docs--but keep in mind the target audience. You're in the right ballpark if:

- your workload involves hundreds/thousands of files; or
- you need to use a script just to use *AP.exe*; or
- you have more RAM or CPU than you know what to do with!

More than likely if you're stuck we can help ▯.

Why do we analyze data in 1-minute chunks?

There are a couple reasons, but mainly, because when we first started doing this, computers were far less efficient than they are now. Computers are fundamentally limited by the amount of data they can work with at one time; in technical terms, the amount of data that can fit into main memory (RAM). By breaking the data into smaller chunks, we could *stream* the data through the analysis and our overall analysis speed was greatly improved.

We could have chosen any size from 30-seconds to 5-minutes, but one-minute blocks also had nice temporal features (they compose well with data at different resolutions) and are still detailed enough to answer questions in our multi-day analyses.

Today it seems to be the de facto standard to analyze data in one-minute blocks. We suggest that it is still a good default for most use cases:

- Computers don't have the same limitations as they did when we started, but small blocks of data allow for parallel analysis that effectively utilizes all CPU cores
- While computer's are getting better, we're also doing more complex analyses. In parallel we can use a large amount of RAM and most of the computer's CPU(s) for the quickest analysis
- One-minute blocks still retain the nice temporally composable attributes detailed above.
- And since one-minute blocks seem to be a defacto standard it does (by happenstance) provide common ground for comparing data

What effect does chunk size have on data?

For acoustic event recognition, typically only boundary effects are affected by chunk-size choice. That is, if an acoustic event occurs and is clipped by by either the start or end of the one-minute chunk, then it is now only a "partial vocalisation". A typical event recognizer may not detect such "partial vocalisations".

For acoustic indices, from a theoretical point of view, chunk-size has the same kinds of issues as the choice of FFT frame length in speech processing. Because an FFT assumes signal stationarity, one chooses a frame length over which the spectral content in the signal of interest is approximately constant. In the case of acoustic indices, one chooses an index calculation duration which captures a sufficient amount of data for the acoustic feature that you are interested in. The longer the analysis duration the more you blur or average out features of interest. However, if you choose too short an interval in then the calculated index may be dominated by "noise"---that is, features that are not of interest. We find this is particularly the case with the ACI index; one-minute seems to be an appropriate duration for the indices that are typically calculated.

Can I change the chunk size?

The config files for most of our analyses contain these common settings:

Chunk-size:

```
# SegmentDuration: units=seconds;
# Long duration recordings are cut into short segments for more
# efficient processing. Default segment length = 60 seconds.
SegmentDuration: 60
```

Here, `SegmentDuration` is our name for the chunk-size. If, for example, you wanted to process data in 5 minute chunks, you could change the configuration to `SegmentDuration: 300`.

Chunk-overlap:

```
# SegmentOverlap: units=seconds;
SegmentOverlap: 0
```

If you're doing event recognition and you're concerned about boundary effects, you could change the overlap to `SegmentOverlap: 10`, which would ensure every `SegmentDuration`-sized-chunk (typically one-minute in size) would be cut with an extra, trailing, 10-second buffer.

Note: we rarely change this setting and setting too much overlap may produce duplicate events.

Index Calculation Duration (for the indices analysis only):

For acoustic indices in particular, their calculation resolution depends on a second setting that is limited by chunk-size (`SegmentDuration`):

```
# IndexCalculationDuration: units=seconds (default=60 seconds)
# The Timespan (in seconds) over which summary and spectral indices are calculated
# This value MUST not exceed value of SegmentDuration.
# Default value = 60 seconds, however can be reduced down to 0.1 seconds for higher resolution.
# IndexCalculationDuration should divide SegmentDuration with MODULO zero
IndexCalculationDuration: 60.0
```

If you wanted indices calculated over a duration longer than one-minute, you could change the `SegmentDuration` and the `IndexCalculation` duration to higher values:

```
SegmentDuration: 300
IndexCalculationDuration: 300
```

However, we suggest that there are better methods for calculating low-resolution indices. A method we often use is to calculate indices at a 60.0 seconds resolution and aggregate the values into lower resolution chunks. The aggregation method can provide some interesting choices:

- We've seen the maximum, median, or minimum value for a block of indices chosen (and sometimes all 3).
    - though be cautious when using a mean function, it can skew the value of logarithmic indices
- And we've seen a block of indices flattened into a larger feature vector and fed to a machine learning or clustering algorithm

We collect metrics/statistics; what information is collected and how is it used?

NOTE: this is an upcoming feature and has not been released yet

AP.exe collects metrics (statistics) so that we can measure important parts of how our program is run. It lets us keep a track of how much computer resources we use, how much audio we analyze, and which commands and analyses are used the most. Metrics help us prioritize new features or important bugs and they also justify continued investment into this software.

We collect mostly anonymous information. All information collected is also embedded into the log file of each run so you can inspect it yourself. AP.exe sends a payload to our metric collector similar to the following (entirely anonymous):

```
{
  "Platform": "Microsoft Windows NT 6.2.9200.0",
  "ProcessorCount": 32,
  "ExecutionTime": 3441.2261135999997,
  "PeakWorkingSet": 239583232,
  “DurationAnalysed”: 86399.75
}
```

When the metrics are collected we also record which public IP address they were sent from.

Metrics can be disabled with an an environment variable. If you're concerned with the data collection then please contact us so we can work out a compromise.

What is a *binary*? What is an *executable*? What does *compiling* mean?

Unlike R, Python, Ruby, or JavaScript, some programming languages can not just be run straight from source code.

We call such languages *compiled* programming languages because a special program, called a *compiler* is required to transform the text-based programming source code to a low-level set of instructions that a computer understands.

Some programming languages that need to be compiled include C++, C, Java, and C#.

This compilation step is discrete and happens before the code is run. Compiling is often referred to as *building*.

The result of compilation is one or more *binary* files. We call these files binaries because the code in them is no longer readable text--rather it is just blobs of binary instructions that the computer will use.

Binaries that can be run as programs are often called *executables*.

What is a *command*?

Our program is a monolith--a very large structure. To support doing different things we have various sub-programs that can be run by themselves. We call each of these sub-programs a *command*. If you run *AP.exe* with no arguments, it will list the available commands (sub-programs) that you can run.

Formats

AP.exe supports:

- WAVE (`.wav`)
    - any sample rate
    - integer PCM only
    - bit depths of 8, 16, 24, and 32
    - multiple channels
- MP3 (`.mp3`)
    - any sample rate
    - VBR or Fixed
    - **NOTE**: MP3 compression in low bitrates creates artefacts in the reconstituted audio file to which some indices (especially ACI) are highly sensitive. For this reason, we discourage the use of MP3 recordings.
- Ogg Vorbis (`.ogg`)
- FLAC (`.flac`)
- Wave Pack (`.wv`)
- WebM (`.webm`)
- WMA (`.wma`)

Versioning

Obtaining the version of *AP.exe*

The software version number can be obtained by simply typing:

```
$ AnalysisPrograms.exe
```

The version number is also shown:

- whenever the program runs
- in the log file every time the program is run
- metadata of the file
- the filename of the release on GitHub
- the tag of the release on GitHub

Interpreting the version

The output looks like:

```
QUT Ecoacoustics Analysis Programs - version 18.03.3.5 (DEBUG build, 2018-03-19 12:23)
Git branch-version: master-517b65bca92f1ed6ce3ea207a5660ff473222424-DIRTY-CI:000
Copyright QUT 2018
```

Our program uses an automatic version numbering system. A version number such as `18.03.3.5` can be deciphered as:

<2-digit-year>.<2-digit-month>.<number-of-releases-this-month>.<commits-since-last-release>.

Thus, version `18.03.3.5` was created in 2018, in March, and is the third release made that month, and there were five changes (commits) since the last release.

The Git branch information can be deciphered as:

<git-branch-when-built>-<latest-commit-hash-when-built>

Log files

Log files record all the messages that you normally see in your terminal into a text file. Because these messages are saved to a text file we can:

- reliably understand what actions the program took
- help debug problems
- extract provenance data to make experiments repeatable.

Important details

- Every time *AP.exe* runs a log file is saved in the `Logs` folder
- The `Logs` folder is in the same directory as the `AnalysisPrograms.exe` program
- The logs will always be named `log_<SOMEDATE>.txt` where `<SOMEDATE>` is replaced with a date time stamp.
- The latest 50 logs will be kept, after which they are deleted
- We recommend you save every log from every analysis you run
  - This will soon be easier once #157 is complete.

Detailed explanation

Because *AP.exe* saves all the messages to a text file we can:

- look at the messages anytime to sort out problems
- send the log files to other people so they can sort out problems
- refer to these logs to determine the provenance of results

The last point is really important. To support reproducible science we need to know what program, which configuration, and what data was used to generate the results.

Our log files record this information. The start of every log file has the following information:

- The date the program was started
- The version of *AP.exe*
- The type of build (either `DEBUG` or `RELEASE`)
- The date the program was compiled
- The unique id of the source code provided by Git (commonly known as the hash)
- The branch that was used to get the source code (usually this is `master`)
- The arguments the program received (which detail which input file and what config file were used)

Here is an example extract that shows all of the above information:

```
2018-03-21T13:44:45.8675459+10:00 [1] INFO  CleanLogger - QUT Ecoacoustics Analysis Programs - version
18.03.3.5 (DEBUG build, 2018-03-19 12:23)
Git branch-version: master-517b65bca92f1ed6ce3ea207a5660ff473222424-CI:123
Copyright QUT 2018
2018-03-21T13:44:45.9035316+10:00 [1] INFO  LogFileOnly - Executable called with these arguments:
"C:\Work\GitHub\audio-analysis\src\AnalysisPrograms\bin\Debug\AnalysisPrograms.exe" -l 7
```

Reporting a bug

If you think you have found a bug, please follow these steps:

1. Try and run the command again.

Often transient problems (like a bad network connection) can cause AP.exe to crash.

Make sure the output directory is empty. Sometimes when AP.exe crashes it leaves behind files that can cause trouble.

2. Double check the arguments

Make sure the arguments, options, and config files you have supplied to AP.exe are valid. Has something changed?

3. Try updating AP.exe

AP.exe is updated regularly. There is a good chance your problem might be solved in a newer version. Try downloading the latest weekly version (see installing for instructions).

4. Check the documentation, and check for issues

Check the documentation on this site for any problems that may relate to the problem you are experiencing.

Additionally, check for any open or closed issues in the AP.exe issue tracker: https://github.com/QutEcoacoustics/audio-analysis/issues?q=is:issue. These issues may contain the answers you need. If someone else has the same issue you can report your problems in the same thread.

5. File a bug report

If all else fails, we'll open a new issue.

To prepare for this, one last time, try running your command again, **but** this time run the command with very verbose logging enabled. Doing so will add much more diagnostic information to the standard logging file that is produced by AP.exe. For more information on the log file see the logs document.

To enable very verbose logging you should add the `--log-level 7` option to the end of your command. When the command has completed collect the log file.

Then open a new issue and fill out the form.

Happy bug hunting!

[WORK IN PROGRESS]: We'll place descriptions of how indices are calculated here.

```
Abbrev,Full Name,Min Value,Max Value,Units,Default Display,Default Display Min,Default Display Max,Description
avAmp-dB,Average signal amplitude,undefined,0,dB,NO,-50,-5,Average amplitude of signal envelope (max value in each frame of 512 values).
bg-dB,Background amplitude,undefined,0,dB,YES,-50,-5,Background amplitude calculated using method of Lamel et al.
snr-dB,Signal to Noise ratio,0,undefined,dB,YES,3,50,SNR = Average amplitude - background amplitude
activeSnr-dB,SNR of active frames,0,undefined,dB,YES,3,10,SNR calculated only from active frames
activity,Fraction of active frames,0,1,real,YES,0,max in array,An active frame is one whose SNR > 3.0 dB
segCount,Count of acoustic segments,0,undefined,integer,YES,0,max in array,An acoustic segment is a consecutive sequence of active frames whose duration > one frame.
avSegDur,Average duration of acoustic segments,0,undefined,milliseconds,YES,0,500,Average duration of acoustic segments as determined above.
hfCover,High frequency cover,0,1,real,YES,0,1,Fraction of high freq (>3500 Hz) spectrogram cells whose amplitude > 0.015.
mfCover,Mid frequency cover,0,1,real,YES,0,1,Fraction of mid-freq (500-3500 Hz) spectrogram cells whose amplitude > 0.015.
lfCover,Low frequency cover,0,1,real,YES,0,1,Fraction of low freq (0-500 Hz) spectrogram cells whose amplitude > 0.015.
AcComplexity,Acoustic Complexity Index,0,1,real,YES,0.3,0.7,AC Index as described by Depretere et al. (2012)
H[temporal],Temporal entropy,0,1,real,YES,0.5,1,Temporal Entropy as described by Sueur et al.
H[spectral],Spectral Entropy,0,1,real,YES,0.5,1,Spectral Entropy as described by Sueur et al.
H[spectralVar],Entropy of Variance Spectrum,0,1,real,NO,0.5,1,Entropy of the spectrum of amplitude variance in each freq bin.
clusterCount,Cluster Count,0,undefined,integer,YES,0,50,Number of mid-band spectral clusters as determined by a simple clustering algorithm
avClustDur,Average Cluster Duration,0,undefined,milliseconds,YES,50,200,Average duration of spectral clusters
3gramCount,Number of 3-grams,0,undefined,integer,YES,0.3,max in array,Number of 3-gram cluster sequences
av3gramRepetition,Av repetition of 3-grams,0,undefined,real,YES,0,max in array,Average repetition of unique 3-gram cluster sequences
SpPkTracks/Sec,Number of Spectral Peak Tracks per second,0,undefined,real,YES,0,max in array,Number of (near) horizontal tracks or whistles in spectrogram (cell ampl > 0.005).
SpPkTracks%Dur,Duration of spectral peak tracks,0,undefined,real,YES,0,max in array,Duration of spectral peak tracks as percent of recording duration
rain,Rain index,0,1,real,YES,0,1,Used to determine occurrence of rain in a recording
cicada,Cicada index,0,1,real,NO,0,1,Used to determine occurrence of cicada singing in a recording
```

Call

A *call* is taken to be any sound of animal origin (whether for communication purposes or not) and includes bird songs/calls, animal vocalizations of any kind, the stridulation of insects, the wingbeats of birds and bats and the various sounds produced by aquatic animals. Calls typically have temporal and spectral structure. For example they may consist of a temporal sequence of two or more *syllables* (with "gaps" in between) or a set of simultaneous *harmonics* or *formants*

Guides Overview

Guides are short focused examples on how to achieve something. They'll require some adaption to work for your use case.

See the tutorials section for long form working examples that use real data.

Available Guides

1. Introduction
2. Basics
    - Introduction
    - Installing
        - Manual install
        - Adding to PATH
    - Command Line Interface
    - Commands
    - Config files
    - Dates
    - FAQ
    - Supported Audio Formats
    - Versioning
    - Logs
    - Reporting bugs
3. Theory
    - Spectrograms
    - Acoustic Events
    - Acoustic Indices
    - Glossary
4. Guides
    - Overview
    - Scripting AP.exe
    - Scripting AP.exe with R
    - DIY Call Recognizer
5. Technical
    - Changelog
    - Contributing
    - Commands
        - Help
        - Analyze Long Recording
        - False Colour Spectrograms
        - Concatenate Index Files
    - Config files
        - GenericRecognizerConfig
        - CommonParameters
        - OscillationParameters
        - HarmonicParameters
        - PostProcessingConfig
        - AnalyzerConfig

Scripting AP.exe

AnalysisPrograms.exe works best when processing single audio files. This has a few advantages:

It simplifies the code. We don't need to know about the particular way you store your data, or the way you want your data processed.

It allows for greater customization. By building a set of composable tools, it allows you to choose what analysis is done, and when. You want it all? You got it. You only want the first bit done once, and the second bit done 100 times with parameter sweeps? Go for it.

It is easier to parallelize. You want to use your university HPC? Write your own distributed analysis? Or just run it in sequence? That's all your choice.

Despite these advantages, we recognize it is useful to have example of how large analyses might be completed. Thus the scripts you find in this folder contain various examples (in various languages) of composing workflows with *AnalysisPrograms.exe*.

You can find a collection of example scripts here: https://github.com/QutEcoacoustics/audio-analysis/tree/master/scripts

PowerShell

You'll see a lot of scripts in that folder that are written in PowerShell. If you're not familiar with it, you can consider it as the Windows equivalent of the Bash shell.

We like PowerShell because we think the syntax is more reasonable than Bash and the enhanced support for dates, times, and regular expressions are well worth the investment.

As of PowerShell 6.0 the shell is cross platform! If you're not convinced, the scripts should be easy enough to reimplement in your favourite language (like Bash)--and we would of course appreciate any translated scripts sent back to us as contributed examples.

Any other language

You can use any programming language to script AP.exe.

R is a popular choice. We have a short guide for using AP.exe with R

Scripting with R

For those more comfortable with R over a terminal, here is an example of an R script that runs *AP.exe*.

Using R like this, where we run *AP.exe* with `system2` is identical to using a shell; `system2` is R's method for running a command.

It is important to still know how to use a shell because you may need to get help (`--help`) for a command, or construct a command that does something else. Using R is just a wrapper around this process.

It might be helpful to think of *R interactive* as a shell (like PowerShell or Bash) and RStudio as a terminal.

Just using R to invoke AP.exe adds a lot of complexity. However, you may want to do this if you need to:

1. to integrate an *AP.exe* analysis in with the rest of your R analysis
2. or to batch analyze data.

Example: batching analysis

Almost always you'd want to run *AP.exe* on more than one file at once. You can use any programming language to do this. Here is an example using R to generate acoustic indices for a folder of audio files:

```r
# Set the directory containing the files
directory <- "C:\\Temp\\Workshop"
# The directory to store the results
base_output_directory <- "C:\\Temp\\Workshop\\BatchIndicesOutput"

# Get a list of audio files inside the directory
# (Get-ChildItem is just like ls, or dir)
files <- list.files(directory, pattern = "*.wav", full.names = TRUE)

# iterate through each file
for(file in files) {
  message("Processing ", file)

  # get just the name of the file
  file_name <- basename(file)

  # make a folder for results
  output_directory <- normalizePath(file.path(base_output_directory, file_name))
  dir.create(output_directory, recursive = TRUE)

  # prepare command
  command <- sprintf('audio2csv "%s" "Towsey.Acoustic.yml" "%s" ', file, output_directory)

  # finally, execute the command
  system2('C:\\AP\\AnalysisPrograms.exe', command)
}
```

Script explained

Set the directory containing the files

Assign using the left arrow operator `<-` the folder where the audio files are located to the variable `directory`, like this:

`directory <- "C:\\Temp\\Workshop"`

This is the directory we want to look in for files.

Storing the results

Similarly, we choose a directory (`base_output_directory`) to store the results when the analyses finish, like this:

```
base_output_directory <- "C:\\Temp\\Workshop\\BatchIndicesOutput"
```

Listing the audio files inside the directory

Which files do we analyze? We could list them all out by hand - but no one got time for that!

Instead we create a list with all the audio files inside the `directory`. In this case, we indicated we want all the files with the extension *.wav* to be listed. If your files have a different extension, you'll need to indicate the right extension after `pattern =`(and don't forget to put the extension between double quotes)

```
files <- list.files(directory, pattern = "*.wav", full.names = TRUE)
```

or, as an example, for FLAC files:

```
files <- list.files(directory, pattern = "*.flac", full.names = TRUE)
```

Iterate through each file

The for loop lets us do something for every file we found. In our example, *for each file* we:

1. Get the file's name. E.g. From `C:\Temp\Workshop\20190801_131213.wav` we get `20190801_131213.wav`
2. Create a folder for each result set inside the `base_output_directory`. The folder will have the name of audio file we're currently working with. E.g. `20190801_131213.wav`
3. Prepare the command, which means put together the bits that will form the command that R and the computer will understand. This is equivalent to typing out the command in a shell.
4. Finally, use `system2` to execute the command and run our analysis. This is equivalent to pressing `Enter` to run our command in the shell.

DIY Call Recognizers

A **DIY Call Recognizer** is a utility within *Analysis Programs* which allows you to write your own call recognizers.

A **DIY call recognizer** uses our *generic recognizer* tools. This guide will help you make your own *generic recognizer*. The generic recognizer allows a user to generically reuse and parametrize our syllable detectors. Once you can detect new syllables those syllables can be combined to form new call recognizers.

NOTE

- Incomplete parts of the manual are indicated by ***TODO***.
- Features not yet implemented are marked with a construction emoji (□).

1. Why make a DIY call recognizer?

There are three levels of sophistication in automated call recognizers:

- The simplest is the handcrafted template.
- More powerful is a *machine learned* model.
- The current cutting edge of call recognizers is *deep-learning* using a convolutional neural network.

A comparison of these recognizer types is shown in the following table and explained further in the subsequent paragraph.

Type of Recognizer	Who does the feature extraction?	Required dataset	Skill level	Accuracy
Template matching	User	Small (even 1!)	Least	Sometimes good
Supervised machine learning	User	Moderate (50-100s)	Some	Better
CNN	Part of CNN learning	Very large (10k to 1M)	A lot!	Best?

A comparison of three different kinds of call recognizer

Hand-crafted, *rule-based* templates can be built using just one or a few examples of the target call. But like any rule-based *AI* system, they are *brittle*, that is, they break easily if the target call falls even slightly outside the bounds of the rules.

A supervised machine-learning model, for example an SVM or Random Forest, is far more resilient to slight changes in the range of the target call but they require many more training examples, on the order of 100 training examples.

Finally, the convolutional neural network (CNN) is the most powerful learning machine available today (2021) but this power is achieved only by supplying thousands of examples of the each target call.

TIP

The following two rules apply to the preparation of training/test datasets, regardless of the recognizer type.

- **Rule 1.** Rubbish in □ rubbish out!
  That is, think carefully about your chosen training/test examples.
- **Rule 2.** Training and test sets should be representative (in some loose statistical sense) of the intended operational environment.

To summarize (and at the risk of over-simplification):

- a hand-crafted template has low cost and low benefit
- a machine-learned model has medium cost and medium benefit
- while a deep-learned model has high cost and high benefit

The cost/benefit ratio in each case is similar but here is the catch - the cost must be paid *before* you get the benefit! Furthermore, in a typical ecological study, a bird species is of interest precisely because it is threatened or cryptic. When not many calls are available, the more sophisticated approaches become untenable. Hence there is a place for hand-crafted templates in call recognition.

These ideas are summarized in the following table:

TYPE OF RECOGNIZER	COST	BENEFIT	COST/BENEFIT RATIO	THE CATCH !
Template matching	Low	Low	A number	You must pay ...
Machine learning	Medium	Medium	A similar number	... the cost before ...
CNN	High	High	A similar number	... you get the benefit!

To summarize, the advantages of a hand-crafted DIY call recognizer are:

1. You can do it yourself!
2. You can start with just one or two calls.
3. Allows you to collect a larger dataset (and refine it) for machine learning purposes.
4. Exposes the variability of the target call as you go.

2. Calls, syllables, harmonics

The algorithmic approach of **DIY Call Recognizer** makes particular assumptions about animals calls and how they are structured. A *call* is taken to be any sound of animal origin (whether for communication purposes or not) and include bird songs/calls, animal vocalizations of any kind, the stridulation of insects, the wingbeats of birds and bats and the various sounds produced by aquatic animals. Calls typically have temporal and spectral structure. For example they may consist of a temporal sequence of two or more *syllables* (with "gaps" in between) or a set of simultaneous *harmonics* or *formants*. (The distinction between harmonics and formants does not concern us here.)

3. Acoustic events

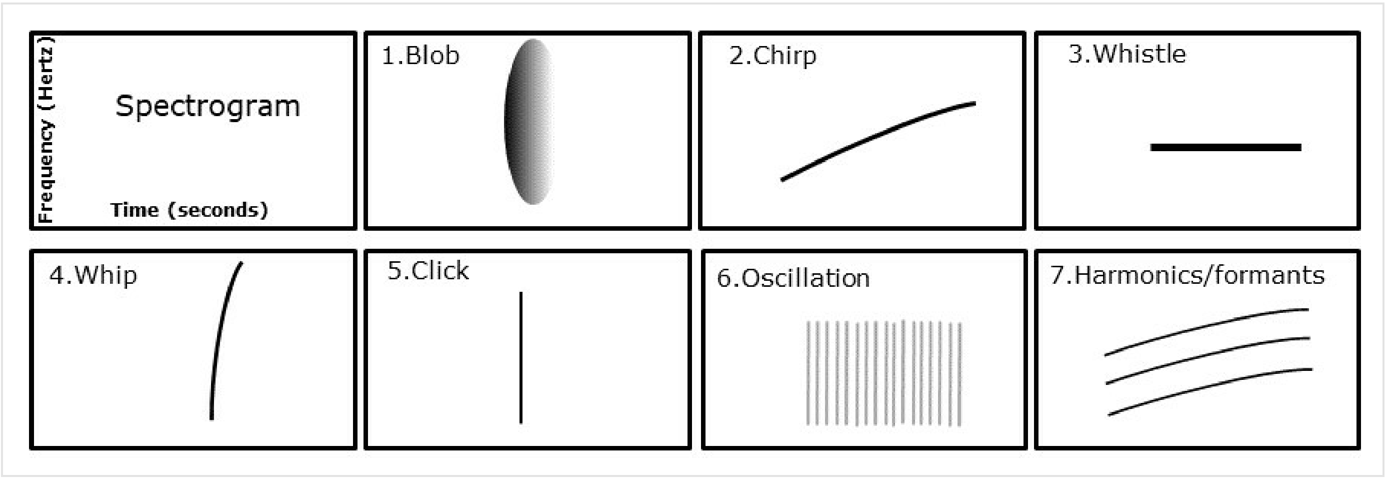
An *acoustic event* is defined as a contiguous set of spectrogram cells/pixels whose decibel values exceed some user defined threshold. In the ideal case, an acoustic event should encompass a discrete component of acoustic energy within a call, syllable or harmonic. It will be separated from other acoustic events by gaps having decibel values *below* the user defined threshold.

DIY Call Recognizer contains algorithms to recognize seven different kinds of *generic* acoustic events based on their shape in the spectrogram.

There are seven types of acoustic events:

1. Shrieks: diffuse events treated as "blobs" of acoustic energy. A typical example is a parrot shriek.
2. Whistles: "pure" tones (often imperfect) appearing as horizontal lines on a spectrogram.
3. Chirps: whistle like events that increases in frequency over time. Appears like a sloping line in a spectrogram.
4. Whips: sound like a "whip crack". They appear as steeply ascending or descending *spectral track* in the spectrogram.
5. Clicks: appear as a single vertical line in a spectrogram and sounds, like the name suggests, as a very brief click.
6. Oscillations: An oscillation is the same (or nearly the same) syllable (typically whips or clicks) repeated at a fixed periodicity over several to many time-frames.
7. Harmonics: Harmonics are the same/similar shaped *whistle* or *chirp* repeated simultaneously at multiple intervals of frequency. Typically, the frequency intervals are similar as one ascends the stack of harmonics.

For more detail on event types see *acoustic events*.



The seven kinds of generic acoustic event

4. Detecting acoustic events

A **DIY Call Recognizer** attempts to recognize calls in a noise-reduced spectrogram using a sequence of steps:

1. Preprocessing—steps to prepare the recording for subsequent analysis.
   1. Input audio is broken up into 1-minute chunks
   2. Audio resampling
2. Processing—steps to identify target syllables as "*generic*" acoustic events
   1. Spectrogram preparation
   2. Call syllable detection
3. Postprocessing—steps which simplify the output combining related acoustic events and filtering events to remove false-positives

   1. Combining syllable events into calls
   2. Syllable/call filtering
4. Saving Results

To execute these detection steps, suitable *parameter values* must be placed into a *configuration file*.

5. Configuration files

All analyses in *AP* require a *configuration file* (henceforth, *config* file) in order to tune the analysis.

It is no different for a generic recognizer. To find calls of interest in a recording *AP* reads the config file which contains *parameters* and then executes the detection steps accordingly.

IMPORTANT

If you're not familiar with AP's config files please review our Config Files page.

Naming

Configuration files must be named in a certain format. The basic format is:

```
<author>.<analysis>[.<tag>]*.yml
```

The `author` and `analysis` name sections are mandatory. The `tag` section is optional, ignored by *AP*, and can be repeated.

See Naming in the Config Files document for more details and examples.

Parameters

Config files contain a list of parameters, each of which is written as a name-value pair, for example:

```
ResampleRate: 22050
```

Changing these parameters allows for the construction of a generic recognizer. This guide will explain the various parameters than can be changed and their typical values. However, this guide will not produce a functional recognizer; each recognizer has to be "tuned" to the target syllables for species to be recognized. Only you can do that.

There are many parameters available. To make config files easier to read we order these parameters roughly in the order that they are applied. This aligns with the basic recognition steps from above.

1. Parameters for preprocessing
2. Parameters for processing
3. Parameters for postprocessing
4. Parameters for saving Results

Profiles

Profiles are a list of acoustic event detection algorithms to use in our processing stage.

> **TIP**
>
> For an introduction to profiles see the Config Files page.

Each algorithm is designed to detect a syllable type. Thus to make a generic recognizer there should be at least one (1) profile in the `Profiles` list. A config file may target more than one syllable or acoustic event, in which case there will be a profile for each target syllable or acoustic event.

The `Profiles` list contains one or more profile items, and each profile has several parameters. So we have a three level hierarchy:

1. The key-word `Profiles` that heads the list.
2. One or more *profile* declarations.
    - There are two parts to each profile declaration:
        1. A user defined name
        2. And the algorithm type to use with this profile (prefixed with an exclamation mark (`!`))
3. The profile *parameters* consisting of a list of name:value pairs

Here is an (abbreviated) example:

```
Profiles:
    BoobookSyllable1: !ForwardTrackParameters
        # min and max of the freq band to search
        MinHertz: 400
        MaxHertz: 1100
        # min and max time duration of call
        MinDuration: 0.1
        MaxDuration: 0.499
    BoobookSyllable2: !ForwardTrackParameters
        MinHertz: 400
        MaxHertz: 1100
        MinDuration: 0.5
        MaxDuration: 0.899
    BoobookSyllable3: !ForwardTrackParameters
        MinHertz: 400
        MaxHertz: 1100
        MinDuration: 0.9
        MaxDuration: 1.2
```

This artificial example illustrates three profiles (i.e. syllables or acoustic events) under the key word `Profiles`.

We can see one of the profiles has been given the name `BoobookSyllable3` and has the type `ForwardTrackParameters`. This means for the `BoobookSyllable3` we want *AP* to use the *forward track* algorithm to look for a *chirp*.

Each profile in this example has four parameters. All three profiles have the same values for `MinHertz` and `MaxHertz` but different values for their time duration. Each profile is processed separately by *AP*.

Algorithm types

In the above example, the line `BoobookSyllable1: !ForwardTrackParameters` is to be read as:

> the name of the target syllable is *BoobookSyllable1* and its type is *ForwardTrackParameters*

There are currently seven algorithm types, each designed to detect a different type of acoustic event. The names of the acoustic events describe what they sound like, whereas, the names of the algorithms (used to find those events) describe how the algorithms work.

This table lists the "generic" events, the algorithm used to detect the event, and the name of the parameter list needed by the algorithm.

ACOUSTIC EVENT	ALGORITHM NAME	PARAMETERS NAME
Shriek	`Blob`	`!BlobParameters`
Whistle	`OnebinTrack`	`!OnebinTrackParameters`
Chirp	`ForwardTrack`	`!ForwardTrackParameters`
Whip	`UpwardTrack`	`!UpwardTrackParameters`
Click	`VerticalTrack`	`!OneframeTrackParameters`
Oscillation	`Oscillation`	`!OscillationParameters`
Harmonic	`Harmonic`	`!HarmonicParameters`

Each of these detection algorithms has some common parameters because all "generic" events are characterized by common properties, such as their minimum and maximum temporal duration, their minimum and maximum frequencies, and their decibel intensity. In fact, every acoustic event is bounded by an *implicit* rectangle or marquee whose height represents the bandwidth of the event and whose width represents the duration of the event.

Even a *chirp* or *whip* which consists only of a single sloping *spectral track*, is enclosed by a rectangle, two of whose vertices sit at the start and end of the track.

See CommonParameters for more details.

6. Config parameters and values

This section describes how to set the parameter values for each of the call-detection steps. We use, as a concrete example, the config file for the Boobook Owl, *Ninox boobook*.

The `YAML` lines are followed by an explanation of each parameter.

Audio segmentation and resampling

Analysis of long recordings is made tractable by breaking them into shorter (typically 60-second) segments. This is done with the Analyze Long Recordings command.

The first part of a generic recognizer config file is as follows:

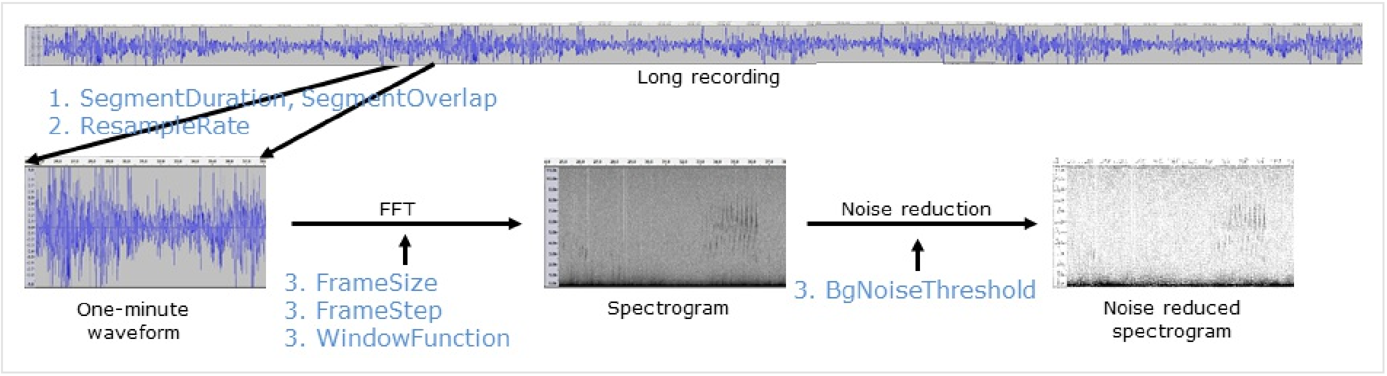
```
# Resample rate must be 2 X the desired Nyquist
ResampleRate: 22050
# SegmentDuration: units=seconds;
SegmentDuration: 60
# SegmentOverlap: units=seconds;
SegmentOverlap: 0
```

These parameters control:

- the size of the segments into which the audio file is split for analysis
- the amount of overlap between consecutive segments
- the sample rate at which the analysis is performed (22050 Hz)

For more information on these parameters see the AnalyzerConfig page.

Segment size and and overlap have good defaults set and you should not need to change them. The best value for sample rate will be analysis dependent, but will default to 22050 Hertz if not provided.



Segmenting and resampling

Adding profiles

For each acoustic event you want to detect, you need to add a profile. Each profile uses one of the generic recognizer algorithms.

Common Parameters

```
# Each of these profiles will be analyzed
# This profile is required for the species-specific recogniser and must have the current name.
Profiles:
    BoobookSyllable: !ForwardTrackParameters
        SpeciesName: NinoxBoobook
```

The key parts here are:

- the profile name (`BoobookSyllable`)
- the algorithm type (`!ForwardTrackParameters` which will detect a *chirp*)
- and an optional species name (`NinoxBoobook`)

Both the profile name and the species names can be any name you like. The names are stored in the results so you know what algorithm generated an event.

We could have a profile name of `banana` and species name of `i_like_golf`—but neither of these names are useful because they are not descriptive.

All algorithms have some common parameters. These include

- Spectrogram settings
- Noise removal settings
- Parameters that set basic limits to the allowed duration and bandwidth of an event

Each algorithm has its own spectrogram settings, so parameters such as `WindowSize` can be varied for *each* type of acoustic event you want to detect.

Common Parameters: Spectrogram preparation

By convention, we list the spectrogram parameters first (after the species name) in each algorithm entry:

```
# Each of these profiles will be analyzed
# This profile is required for the species-specific recogniser and must have the current name.
Profiles:
    BoobookSyllable: !ForwardTrackParameters
        SpeciesName: NinoxBoobook
        FrameSize: 1024
        FrameStep: 256
        WindowFunction: HANNING
        BgNoiseThreshold: 0.0
```

- `FrameSize` sets the size of the FFT window.
- `FrameStep` sets the number of samples between frame starts.
- `WindowFunction` sets the FFT window function.
- `BgNoiseThreshold` sets the degree of background noise removal.

Since these parameters are so important for the success of call detection, you are strongly advised to refer to the Spectrograms document for more information about setting their values.

Common Parameters: Call syllable limits

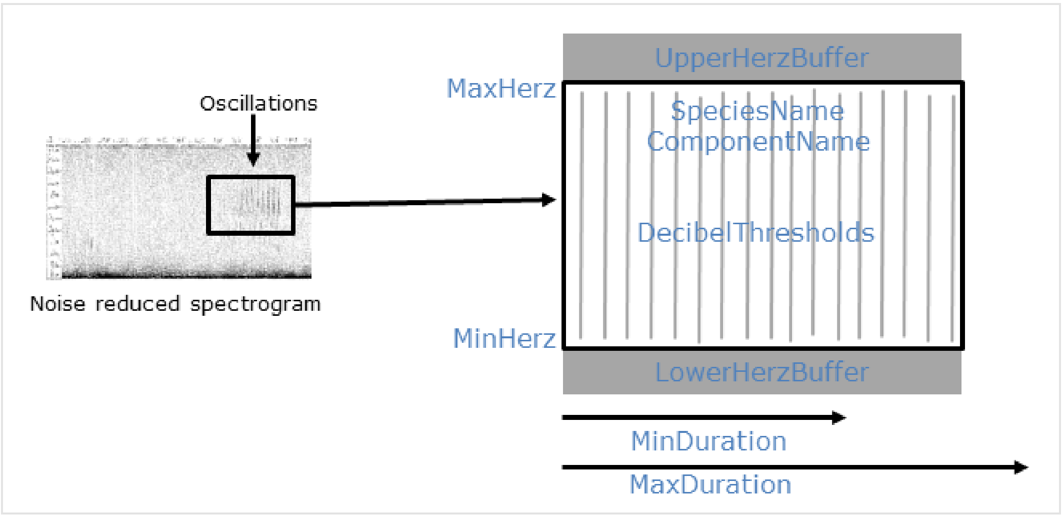
A complete definition of the `BoobookSyllable` follows.

```
# Each of these profiles will be analyzed
# This profile is required for the species-specific recogniser and must have the current name.
Profiles:
    BoobookSyllable: !ForwardTrackParameters
        SpeciesName: NinoxBoobook
        FrameSize: 1024
        FrameStep: 256
        WindowFunction: HANNING
        BgNoiseThreshold: 0.0

        # min and max of the freq band to search
        MinHertz: 400
        MaxHertz: 1100
        MinDuration: 0.17
        MaxDuration: 1.2
        # Scan the frequency band at these thresholds
        DecibelThresholds:
            - 6.0
            - 9.0
            - 12.0
```

The additional parameters direct the actual search for target syllables in the spectrogram.

- `MinHertz` and `MaxHertz` set the frequency band in which to search for the target event. Note that these parameters define the bounds of the search band, *not* the bounds of the event itself. These limits are hard bounds.
- `MinDuration` and `MaxDuration` set the minimum and maximum time duration (in seconds) of the target event. These limits are hard bounds.



Common parameters for all acoustic events, using an oscillation event as example.

Adding profiles with algorithms

For brevity, we've broken up the descriptions of each algorithm to their own pages. Some of these algorithms have extra parameters, some do not, but all do have the common parameters we've previously described.

I WANT TO FIND A	I'LL USE THIS ALGORITHM
Shriek	!BlobParameters
Whistle	🚧 !OnebinTrackAlgorithm 🚧
Chirp	!ForwardTrackParameters
Whip	🚧!UpwardTrackParameters 🚧

I WANT TO FIND A	I'LL USE THIS ALGORITHM
Click	🚧 !OneframeTrackParameters 🚧
Oscillation	!OscillationParameters
Harmonic	!HarmonicParameters

Post Processing

The post processing stage is run after event detection (the `Profiles` section). Add a post processing section to you config file by adding the `PostProcessing` parameter and indenting the sub-parameters.

```
PostProcessing:
```

Post processing is optional - you may decide to combine or filter the "raw" events using code you have written yourself.

NOTE

If you do not wish to include a post-processing step, *disable* it by deleting its keyword and all component parameters. Alternatively, you can *comment out* the relevant lines by inserting a `#` at the start of each line. Disabling a post-processing filter means that all events are accepted (not filtered out) for that step.

Order of operation

There are six post-processing steps, each of which is optional. However the order in which these steps are performed *cannot* be changed by the user. The post-processing sequence is:

1. Combine events having temporal *and* spectral overlap.
2. Combine possible sequences of events that constitute a "call".
3. Remove (filter) events whose duration is outside an acceptable range.
4. Remove (filter) events whose bandwidth is outside an acceptable range.
5. Remove (filter) events having excessive acoustic activity in their sidebands.
6. Remove (filter) events that are enclosed by another event.

Post-processing steps 1 through 5 are performed once for each of the `DecibelThresholds` used in the event detection stage. Post-processing step 6 is performed on all the events emerging from all rounds of post-processing steps 1-5.

Combine events having temporal *and* spectral overlap

```
PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
```

The `CombineOverlappingEvents` parameter is typically set to `true`, but it depends on the target call. You would typically set this to true for two reasons:

- the target call is composed of two or more overlapping syllables that you want to join as one event.
- whistle events often require this step to unite whistle fragment detections into one event.

Combine possible sequences of events that constitute a "call"

Unlike overlapping events, if you want to combine a group of events (like syllables) that are near each other but not overlapping, then make use of the `SyllableSequence` parameter. A typical example would be to join a sequence of chirps in a honeyeater call.

```
PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
    CombineOverlappingEvents: true

    # 2: Combine possible syllable sequences
    SyllableSequence:
        SyllableStartDifference: 0.6
        SyllableHertzGap: 350
        FilterSyllableSequence: true
        SyllableMaxCount: 2
        ExpectedPeriod: 0.4
```

`SyllableStartDifference` and `SyllableHertzGap` set the allowed tolerances when combining events into sequences

- `SyllableStartDifference` sets the maximum allowed time difference (in seconds) between the starts of two events.
- `SyllableHertzGap` sets the maximum allowed frequency difference (in Hertz) between the minimum frequencies of two events.

Once you have combined possible sequences, you may wish to remove sequences that do not satisfy the periodicity constraints for your target call, that is, the maximum number of syllables permitted in a sequence and the average time gap between syllables. To enable filtering on syllable periodicity, set `FilterSyllableSequence` to true and assign values to `SyllableMaxCount` and `ExpectedPeriod`.

- `SyllableMaxCount` sets an upper limit on the number of events that constitute an allowed sequence.
- `ExpectedPeriod` sets an expectation value for the average period (in seconds) of an allowed combination of events.

NOTE

The properties `ExpectedPeriod` and `SyllableStartDifference` interact.

Refer to the following documentation for more information: EventPostProcessing.SyllableSequenceConfig.

Remove events whose duration is outside an acceptable range

```
PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
    CombineOverlappingEvents: true

    # 2: Combine possible syllable sequences
    SyllableSequence:
        SyllableStartDifference: 0.6
        SyllableHertzGap: 350
        FilterSyllableSequence: true
        SyllableMaxCount: 2
        ExpectedPeriod: 0.4

    # 3: Remove events whose duration lies outside 3 SDs of an expected value.
    #Duration:
    #    ExpectedDuration: 0.14
    #    DurationStandardDeviation: 0.01

    # 4: Remove events whose bandwidth lies outside 3 SDs of an expected value.
    Bandwidth:
        ExpectedBandwidth: 280
        BandwidthStandardDeviation: 40
```

Use the parameter `Duration` to filter out events that are too long or short. There are two parameters:

- `ExpectedDuration` defines the *expected* or *average* duration (in seconds) for the target events.
- `DurationStandardDeviation` defines *one* SD of the assumed distribution.

Refer to the following documentation for more information: EventPostProcessing.DurationConfig.

Remove events whose bandwidth is outside an acceptable range

Use the parameter `Bandwidth` to filter out events whose bandwidth is too small or large. There are two parameters:

- `ExpectedBandwidth` defines the *expected* or *average* bandwidth (in Hertz) for the target events.
- `BandwidthStandardDeviation` defines one SD of the assumed distribution.

Refer to the following documentation for more information: EventPostProcessing.BandwidthConfig.

Remove events that have excessive noise or acoustic activity in their side-bands

```
PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
    CombineOverlappingEvents: true

    # 2: Combine possible syllable sequences
    SyllableSequence:
        SyllableStartDifference: 0.6
        SyllableHertzGap: 350
        FilterSyllableSequence: true
        SyllableMaxCount: 2
        ExpectedPeriod: 0.4

    # 3: Remove events whose duration lies outside 3 SDs of an expected value.
    #Duration:
    #    ExpectedDuration: 0.14
    #    DurationStandardDeviation: 0.01

    # 4: Remove events whose bandwidth lies outside 3 SDs of an expected value.
    Bandwidth:
        ExpectedBandwidth: 280
        BandwidthStandardDeviation: 40

    # 5: Filter the events for excess activity in their sidebands
    SidebandAcousticActivity:
        LowerSidebandWidth: 150
        #UpperSidebandWidth: 200
        MaxBackgroundDecibels: 12
        #MaxActivityDecibels: 12
```

The intuition of this filter is that an unambiguous event (representing a call or syllable) should have an "acoustic-free zone" above and below it. This filter removes an event that has "excessive" acoustic activity spilling into its sidebands. Such events are likely to be *broadband* events unrelated to the target event. Since this is a common occurrence, a sideband filter is useful.

For details on configuring this step see EventPostProcessing.PostProcessingConfig.

Remove events that are enclosed by other events

Running profiles with multiple decibel thresholds can produce sets of enclosed (wholly overlapped by another event) events that are actually the result of detecting the same acoustic syllable. This final (optional) post-processing step is to remove all but the outermost event of any nested set. Enable this option by setting the parameter `RemoveEnclosedEvents` to `true`. You would typically do this only after reviewing the output spectrograms to confirm that you have sets of overlapping events.

```
PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
    CombineOverlappingEvents: true

    # 2: Combine possible syllable sequences
    SyllableSequence:
        SyllableStartDifference: 0.6
        SyllableHertzGap: 350
        FilterSyllableSequence: true
        SyllableMaxCount: 2
        ExpectedPeriod: 0.4

    # 3: Remove events whose duration lies outside 3 SDs of an expected value.
    #Duration:
    #    ExpectedDuration: 0.14
    #    DurationStandardDeviation: 0.01

    # 4: Remove events whose bandwidth lies outside 3 SDs of an expected value.
    Bandwidth:
        ExpectedBandwidth: 280
        BandwidthStandardDeviation: 40

    # 5: Filter the events for excess activity in their sidebands
    SidebandAcousticActivity:
        LowerSidebandWidth: 150
        #UpperSidebandWidth: 200
        MaxBackgroundDecibels: 12
        #MaxActivityDecibels: 12

    # 6: In the case of sets of nested/enclosed events,
    # filter/remove all but the outermost event.
    RemoveEnclosedEvents: true
```

##### How `RemoveEnclosedEvents` is applied Suppose you have three decibel thresholds (6, 9 and 12 dB is a typical set of values) in each of two profiles. There will be three rounds of post-processing: 1. All the events detected (by both profiles) at threshold 6 dB will be subject to those post-processing steps 1-5 enabled by the user. 2. Next, all events detected at the 9 dB threshold will be passed through the same post-processing steps. 3. Next, all events detected at the 12 dB threshold will be passed through the same post-processing steps. The final option (step 6) is to collect all events emerging from all previous rounds of post-processing and to remove those that are enclosed by another event.

Parameters for saving results

The parameters in this final part of the config file determine what results are saved to file.

```

# Options to save results files
# Available options for saving spectrograms: [Never | Always | WhenEventsDetected]
# "Always" can be useful when debugging but "WhenEventsDetected" is a good default.
SaveSonogramImages: WhenEventsDetected

# Available options for saving data files: [Never | Always | WhenEventsDetected]
SaveIntermediateWavFiles: Never
```

Each of the parameters controls whether extra diagnostic files are saved while doing an analysis.

IMPORTANT

If you are doing a lot of analysis **you'll want to disable** this extra diagnostic output. It will produce files that are in total larger than the input audio data—you will fill your harddrive quickly!

- `SaveSonogramImages` will save a spectrogram for each analysed segment (typically one-minute)
- `SaveIntermediateWavFiles` will save the converted WAVE file used to analyze each segment

Both parameters accept three values:

- `Never`: disables the output.
- `WhenEventsDetected`: only outputs the spectrogram/WAVE file when an event is found in the current segment. This choice is the most useful for debugging a new recognizer.
- `Always`: always save the diagnostic files. Don't use this option if you're going to analyze a lot of files

The completed example

Here is a the completed config file for the hypothetical boobook recognizer we have been working with:

```
---
# This is a non-functional example used for documentation. Please see the real config file for the Boobook
recognizer.
# Boobook Owl = Towsey.NinoxBoobook
# Resample rate must be 2 X the desired Nyquist
ResampleRate: 22050
# SegmentDuration: units=seconds;
SegmentDuration: 60
# SegmentOverlap: units=seconds;
SegmentOverlap: 0

# Each of these profiles will be analyzed
# This profile is required for the species-specific recogniser and must have the current name.
Profiles:
    BoobookSyllable: !ForwardTrackParameters
        SpeciesName: NinoxBoobook
        FrameSize: 1024
        FrameStep: 256
        WindowFunction: HANNING
        BgNoiseThreshold: 0.0

        # min and max of the freq band to search
        MinHertz: 400
        MaxHertz: 1100
        MinDuration: 0.17
        MaxDuration: 1.2
        # Scan the frequency band at these thresholds
        DecibelThresholds:
            - 6.0
            - 9.0
            - 12.0

#################### POST-PROCESSING of EVENTS ###################

PostProcessing:
    # The following generic post-processing steps are determined by config settings.
    # Step 1: Combine overlapping events - events derived from all profiles.
    # Step 2: Combine possible syllable sequences and filter on excess syllable count.
    # Step 3: Remove events whose duration is too small or large.
    # Step 4: Remove events whose bandwidth is too small or large.
    # Step 5: Remove events that have excessive noise in their side-bands.

    # 1: Combine overlapping events
    CombineOverlappingEvents: true
    # 2: Combine possible syllable sequences
    SyllableSequence:
        SyllableStartDifference: 0.6
        SyllableHertzGap: 350
        FilterSyllableSequence: true
        SyllableMaxCount: 2
        ExpectedPeriod: 0.4

    # 3: Remove events whose duration lies outside 3 SDs of an expected value.
    #Duration:
    #    ExpectedDuration: 0.14
    #    DurationStandardDeviation: 0.01

    # 4: Remove events whose bandwidth lies outside 3 SDs of an expected value.
    Bandwidth:
        ExpectedBandwidth: 280
        BandwidthStandardDeviation: 40

    # 5: Filter the events for excess activity in their sidebands
    SidebandAcousticActivity:
        LowerSidebandWidth: 150
        #UpperSidebandWidth: 200
        MaxBackgroundDecibels: 12
        #MaxActivityDecibels: 12

    # 6: In the case of sets of nested/enclosed events,
    # filter/remove all but the outermost event.
    RemoveEnclosedEvents: true

# Options to save results files
# Available options for saving spectrograms: [Never | Always | WhenEventsDetected]
# "Always" can be useful when debugging but "WhenEventsDetected" is a good default.
SaveSonogramImages: WhenEventsDetected

# Available options for saving data files: [Never | Always | WhenEventsDetected]
SaveIntermediateWavFiles: Never

...
```

7. An efficient strategy to tune parameters

Tuning parameter values can be frustrating and time-consuming if a logical sequence is not followed. The idea is to tune parameters in the sequence in which they appear in the config file, keeping all "downstream" parameters as "open" or "unrestrictive" as possible. Here is a suggested tuning strategy:

1. Turn off all post-processing steps. That is, comment out all post-processing keywords/parameters.
2. Initially set all profile parameters so as to catch the maximum possible number of target calls/syllables.
    1. Set the array of decibel thresholds to cover the expected range of call amplitudes from minimum to maximum decibels.
    2. Set the minimum and maximum duration values to catch every target call by a wide margin. At this stage, do not worry that you are also catching a lot of false-positive events.
    3. Set the minimum and maximum frequency bounds to catch every target call by a wide margin. Once again, do not worry that you are also catching a lot of false-positive events.
    4. Set other parameters to their least "restrictive" values in order to catch maximum possible target events.

At this point you should have "captured" all the target calls/syllables (i.e. there should be minimal false-negatives), *but* you are likely to have many false-positives.

1. Gradually constrain the parameter bounds (i.e. increase minimum values and decrease maximum values) until you start to

lose obvious target calls/syllables. Then back off so that once again you just capture all the target events—but you will still have several to many false-positives.
2. Event combining: You are now ready to set parameters that determine the *post-processing* of events. The first post-processing steps combine events that are likely to be *syllables* that are part of the same *call*.
3. Event Filtering: Now add in the event filters in the same sequence as they appear in the config file. This sequence cannot currently be changed because it is determined by the underlying code. There are event filters for duration, bandwidth, periodicity of component syllables within a call and finally acoustic activity in the sidebands of an event.
    1. Set the `periodicity` parameters for filtering events based on syllable sequences.
    2. Set the `duration` parameters for filtering events on their time duration.
    3. Set the `bandwidth` parameters for filtering events on their bandwidth.
    4. Set the `SidebandAcousticActivity` parameters for filtering based on sideband *acoustic activity*.

NOTE

You are unlikely to want to use all filters. Some may be irrelevant to your target call.

At the end of this process, you are likely to have a mixture of true-positives, false-positives and false-negatives. The goal is to set the parameter values so that the combined FP+FN total is minimized. You should adjust parameter values so that the final FN/FP ratio reflects the relative costs of FN and FP errors. For example, lowering a decibel threshold may pick up more TPs but almost certainly at the cost of more FPs.

NOTE

A working DIY Call Recognizer can be built with just one example or training call. A machine learning algorithm typically requires 100 true and false examples. The price that you (the ecologist) pays for this simplicity is the need to exercise some of the "intelligence" that would otherwise be exercised by the machine learning algorithm. That is, you must select calls and set parameter values that reflect the variability of the target calls and the relative costs of FN and FP errors.

8. Eight steps to building a DIY Call Recognizer

We described above the steps required to tune parameter values in a recognizer config file. We now step back from this detail and take an overview of all the steps required to obtain an operational recognizer for one or more target calls.

1. Select one or more one-minute recordings that contain typical examples of your target call. It is also desirable that the background acoustic events in your chosen recordings are representative of the intended operational environment. If this is difficult, one trick to try is to play examples of your target call through a loud speaker in a location that is similar to your intended operational environment. You can then record these calls using your intended Acoustic Recording Unit (ARU).
2. Assign parameter values into your config.yml file for the target call(s).
3. Run the recognizer, using the command line described in the next section.
4. Review the detection accuracy and try to determine reasons for FP and FN detections.
5. Tune or refine parameter values in order to increase the detection accuracy.
6. Repeat steps 3, 4 and 5 until you appear to have achieved the best possible accuracy. In order to minimize the number of iterations of stages 3 to 5, it is best to tune the configuration parameters in the sequence described in the previous section.
7. At this point you should have a recognizer that performs "as accurately as possible" on your training examples. The next step is to test your recognizer on one or a few examples that it has not seen before. That is, repeat steps 3, 4, 5 and 6 adding in a new example each time as they become available. It is also useful at this stage to accumulate a set of recordings that do *not* contain the target call. See Section 10 for more suggestions on building datasets.
8. At some point you are ready to use your recognizer on recordings obtained from the operational environment.

9. Running a generic recognizer

AP performs several functions. Each function is selected by altering the command used to run *AP*.

For running a generic recognizer we need to to use the `audio2csv` command.

- For an introduction to running commands see Running commands
- For detailed help on the audio2csv command see Analyze Long Recordings

The basic form of the command line is:

```
AnalysisPrograms.exe audio2csv <input_file> <config_file> <output_folder> --analysis-identifier
"Ecosounds.GenericRecognizer"
```

When you run the command, swap out `<input_file>`, `<config_file>`, and `<output_folder>` for the actual paths to your audio, your config file, and your desired output folder respectively. For example, if the files `birds.wav` and `NinoxBoobook.yml` were in the current folder and you want to save output to the folder `/BoobookResults`, one could run:

```
AnalysisPrograms.exe audio2csv birds.wav NinoxBoobook.yml BoobookResults --analysis-identifier
"Ecosounds.GenericRecognizer"
```

NOTE

The analysis-identifier (`--analysis-identifier` followed by the `"Ecosounds.GenericRecognizer"`) is required for generic recognizers. Using `--analysis-identifier` informs *AP* that this is generic recognition task and enables it to perform the correct analysis.

If you want to run your generic recognizer more than once, you might want to use powershell or R to script *AP*.

10. Building a larger data set

As indicated above, it is useful to accumulate a set of recordings, some of which contain the target call and some of which *do not*. The *negative* examples should include acoustic events that have previously been detected as FPs.

You now have two sets of recordings, one set containing the target call(s) and one set containing previous FPs and other possibly confusing acoustic events. The idea is to tune parameter values, while carefully watching for what effect the changes have on both data sets.

Eventually, these two labelled data sets can be used for

- validating the efficacy of your recognizer
- or for machine learning purposes.

Egret is software designed to assess large datasets for recognizer performance, in an **automated** fashion. *Egret* can greatly speed up the development of a recognizer because it is easier to repeatedly test small changes to your recognizer parameters.

Egret is available from https://github.com/QutEcoacoustics/egret.

Changelog

Unreleased

Ecoacoustics Audio Analysis Software v20.11.2.0 2020-11-15

Version v20.11.2.0

Details

Compare v20.11.0.0...v20.11.2.0

- 16075878 QUT Ecoacoustics - Update changelog for v20.11.1.24
- e7e974f4 towsey - (#390) Update GenericRecognizer.cs
- f080005e towsey - (#390) Wrote unit tests for Event filters
- 7c4cf23d Michael Towsey - (#390) Refactor GenericRecognizer class and associated tests
- 493a1524 Michael Towsey - (#390) Change recognizer.config.yml files
- ecff057b Michael Towsey - (#390) Fix Aust Bittern test
- 1abf01b6 Michael Towsey - (#390) Shift array of decibel thresholds from GenericRecognizer class to CommonParameters
- e82f81a0 Michael Towsey - (#390) Fixed tests associated with the previous changes
- 7150e802 Michael Towsey - (#390) Update GenericRecognizer.cs
- 772e18d8 Michael Towsey - (#390) Changes to the sideband activity filter
- e80aa2c9 towsey - (#390) Fix unit tests for AED and Cisticola
- 947cf40a towsey - (#390) Adjust yml configs to new system of decibel thresholds.
- 4d2ba7a5 towsey - (#390) Refactor options for the Aust. Bittern
- 80a36bde towsey - (#390) Pass decibel threshold to each recognizer.
- cf30bdf5 towsey - (#390) Fixed bug that arose when first implementing multiple decibel thresholds
- 8630b971 towsey - (#390) Update CompositeEvent.cs
- 120f8cb6 towsey - (#390) Create EventFilterTests.cs
- 828347fc towsey - (#390) Shift some classes into new folder structure
- 2648ea03 towsey - (#390) Rewrite unit tests to work with new method of using multiple decibel thresholds.
- 43ab9f90 towsey - (#390) Update OnebinTrackAlgorithm.cs
- 631ec947 towsey - (#390) Update CompositeEvent.cs
- aecccbc9 towsey - (#390) Update EventFilters.cs
- 40b48251 towsey - (#390) Split the GenericRecognizer class
- 0d143de8 towsey - (#390) Refactor the sideband filter for acoustic events
- 5d0a3bd2 towsey - (#390) Refactor code for filtering on sideband activity. ## Ecoacoustics Audio Analysis Software v20.11.1.24 2020-11-08

Version v20.11.1.24

Details

Compare v20.11.0.0...v20.11.1.24

- e7e974f4 towsey - (#390) Update GenericRecognizer.cs
- f080005e towsey - (#390) Wrote unit tests for Event filters
- 7c4cf23d Michael Towsey - (#390) Refactor GenericRecognizer class and associated tests
- 493a1524 Michael Towsey - (#390) Change recognizer.config.yml files
- ecff057b Michael Towsey - (#390) Fix Aust Bittern test
- 1abf01b6 Michael Towsey - (#390) Shift array of decibel thresholds from GenericRecognizer class to CommonParameters
- e82f81a0 Michael Towsey - (#390) Fixed tests associated with the previous changes
- 7150e802 Michael Towsey - (#390) Update GenericRecognizer.cs
- 772e18d8 Michael Towsey - (#390) Changes to the sideband activity filter

- e80aa2c9 towsey - (#390) Fix unit tests for AED and Cisticola
- 947cf40a towsey - (#390) Adjust yml configs to new system of decibel thresholds.
- 4d2ba7a5 towsey - (#390) Refactor options for the Aust. Bittern
- 80a36bde towsey - (#390) Pass decibel threshold to each recognizer.
- cf30bdf5 towsey - (#390) Fixed bug that arose when first implementing multiple decibel thresholds
- 8630b971 towsey - (#390) Update CompositeEvent.cs
- 120f8cb6 towsey - (#390) Create EventFilterTests.cs
- 828347fc towsey - (#390) Shift some classes into new folder structure
- 2648ea03 towsey - (#390) Rewrite unit tests to work with new method of using multiple decibel thresholds.
- 43ab9f90 towsey - (#390) Update OnebinTrackAlgorithm.cs
- 631ec947 towsey - (#390) Update CompositeEvent.cs
- aecccbc9 towsey - (#390) Update EventFilters.cs
- 40b48251 towsey - (#390) Split the GenericRecognizer class
- 0d143de8 towsey - (#390) Refactor the sideband filter for acoustic events
- 5d0a3bd2 towsey - (#390) Refactor code for filtering on sideband activity. ## Ecoacoustics Audio Analysis Software v20.11.0.0 2020-11-01

Version v20.11.0.0

Details

Compare v20.10.1.58...v20.11.0.0

- 388daf70 QUT Ecoacoustics - Update changelog for v20.10.2.1
- f7d6e8a2 dependabot-preview[bot] - Bump FSharp.Core from 4.7.2 to 5.0.0 ## Ecoacoustics Audio Analysis Software v20.10.2.1 2020-10-25

Version v20.10.2.1

Details

Compare v20.10.1.58...v20.10.2.1

- f7d6e8a2 dependabot-preview[bot] - Bump FSharp.Core from 4.7.2 to 5.0.0 ## Ecoacoustics Audio Analysis Software v20.10.1.58 2020-10-18

Version v20.10.1.58

Details

Compare v20.10.0.3...v20.10.1.58

- cc619074 dependabot-preview[bot] - Bump CsvHelper from 15.0.6 to 15.0.8
- 224f4e5f dependabot-preview[bot] - Bump System.IO.Abstractions from 12.2.5 to 12.2.7
- 724aee85 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration from 3.1.8 to 3.1.9
- 466757e6 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration.Json from 3.1.8 to 3.1.9
- 53d4cf66 dependabot-preview[bot] - Bump System.IO.Abstractions from 12.2.1 to 12.2.5
- 625ed67e towsey - (#370) Update two config files
- ea1f1dbe towsey - Improve result on Cisticola recognizer test. Now get 19 TP detections
- 282d5a33 towsey - (#370) Fix Boobook Owl test
- 28c2eda0 towsey - (#370) Fix unit test for Cisticola
- 2b417829 towsey - (#370) Add new filter method to generic filters
- 53e9c4dc towsey - (#370) Fix unit test for Australasian Pipit.
- dee47b7a towsey - (#370) Revise unit tests for the Bittern
- 773181de towsey - (#370) Change the filter for syllable sequences
- 55ddfad3 towsey - (#370) Update GenericRecognizer.cs
- c4b01133 towsey - Update NormalDist.cs

- ea810744 towsey - (#370) Update Towsey.BotaurusPoiciloptilus.yml
- 5f0ddbe5 towsey - (#370) Update NoiseRemoval_Briggs.cs
- bbdedfd8 towsey - (#370) Update GenericRecognizerTests.cs
- ff97af89 towsey - (#370) Change yml file to be consistent with new post-processing hierarchy.
- 751d9e92 towsey - (#370) Set up PostProcessing as a top level key in generic recognizers
- 5acab67a towsey - (#370) Change method names to follow accepted convention
- e45a575f towsey - (#370) Refactor methods for recognition of the main generic acoustic events.
- ee3707f3 towsey - (#370) Refactor code for the recognition of blob events
- cbab908c towsey - (#370) Fix tests for powerfull owl
- 9df9bf5c towsey - (#370) Update Plot.cs
- 5958a596 towsey - (#370) Enable Oscillation recognizer to utilize an array of decibel thresholds
- 24cdc16b towsey - (#370) Enable Harmonic recognizer to utilize an array of decibel trhsholds.
- cc951abb towsey - (#370) Create a new class to contain methods that filter lists of acoustic Events
- 55fda898 towsey - (#370) Update GenericRecognizerTests.cs
- 5fb425d7 towsey - (#370) Change Track Algorithm classes to accept an array of decibel thresholds
- 676725d9 towsey - (#370) Update CommonParameters.cs
- 4664320e towsey - (#370) Update Plot.cs
- 9c0abc70 towsey - (#370) Shift method PreparePlot()
- 009ce06f towsey - (#370) Fixing broken Unit tests
- 2a75cc60 towsey - (#370) Fix unit test for Powerful owl
- 0a05f78c towsey - (#370) Update NinoxStrenua.cs
- da87a2a1 towsey - (#370) Update ForwardTrackAlgorithm.cs
- abc0199d towsey - (#370) Update GenericRecognizer.cs
- 9575f90f towsey - (#370) Update ForwardTrackAlgorithm.cs
- c3bfb483 towsey - Adjust parameters for powerful Owl Test
- c78e0186 towsey - (#370) Rework Powerful Owl recognizer
- 851aebeb towsey - (#370) Revise and refactor methods to do with Local Contrast Normalisation
- c09d8c4d towsey - (#370) Try new test file for Powerful Owl
- 5db186e7 towsey - (#370) Small changes
- efde855f towsey - (#370) Change config file to catch wider range of recordings.
- 7d0f6bfb towsey - (#370) Update EventExtentions.cs
- d8ec8a96 towsey - (#370) Optimise parameter values for Australasian Bittern.
- b8c4d5ba towsey - (#370) Small changes to config files
- c0e20822 towsey - (#370) Refactored code for the Powerful Owl
- aef31d3f towsey - (#370) Refactored code for the Boobook owl
- def462ac towsey - (#370) Refactored code for the Australasian Bittern
- 87ae78c4 towsey - (#370) Create a new class SyllableSequenceConfig
- 4c674bfb towsey - (#370) Write methods to calculate the periodicity of syllable sequence
- 0070fb03 towsey - (#370) Update NinoxStrenua.cs
- 0bcdedef towsey - (#370) Update PowerfulOwlTests.cs
- 0560c616 towsey - (#370) Rearrange the post-processing steps.
- 2ee86d86 towsey - (#370) Create two new generic post-processing steps
- 6ec55c69 towsey - (#370) Set up recognizer for Powerful Owl ## Ecoacoustics Audio Analysis Software v20.10.0.3 2020-10-11

Version v20.10.0.3

Details

Compare v20.9.3.2...v20.10.0.3

- 2f983a42 dependabot-preview[bot] - Bump Moq from 4.14.5 to 4.14.6
- 9e8f0847 dependabot-preview[bot] - Bump CsvHelper from 15.0.5 to 15.0.6
- 025eafce dependabot-preview[bot] - Bump System.IO.Abstractions from 12.1.9 to 12.2.1 ## Ecoacoustics Audio Analysis

Software v20.9.3.2 2020-09-27

Version v20.9.3.2

Details

Compare v20.9.2.2...v20.9.3.2

- f78c09a0 Anthony Truskinger - Update DrawLongDurationSpectrograms.cs
- 18c21d0e Anthony Truskinger - Update .zenodo.json ## Ecoacoustics Audio Analysis Software v20.9.2.2 2020-09-20

Version v20.9.2.2

Details

Compare v20.9.1.1...v20.9.2.2

- a5934517 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration.Json from 3.1.7 to 3.1.8
- 6de06109 dependabot-preview[bot] - Bump System.IO.Abstractions from 12.1.1 to 12.1.9 ## Ecoacoustics Audio Analysis Software v20.9.1.1 2020-09-13

Version v20.9.1.1

Details

Compare v20.9.0.2...v20.9.1.1

- f1ae620c dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration from 3.1.7 to 3.1.8 ## Ecoacoustics Audio Analysis Software v20.9.0.2 2020-09-06

Version v20.9.0.2

Details

Compare v20.8.1.5...v20.9.0.2

- 2c4cba67 dependabot-preview[bot] - Bump SQLitePCLRaw.bundle_green from 2.0.3 to 2.0.4
- 7582ae84 dependabot-preview[bot] - Bump ImmediateReflection from 1.5.0 to 1.6.0 ## Ecoacoustics Audio Analysis Software v20.8.1.5 2020-08-30

Version v20.8.1.5

Details

Compare v20.8.0.63...v20.8.1.5

- 7501ad59 dependabot-preview[bot] - Bump log4net from 2.0.8 to 2.0.9
- e20911b3 dependabot-preview[bot] - Bump Microsoft.NET.Test.Sdk from 16.6.1 to 16.7.1
- 5d8c010f dependabot-preview[bot] - Bump MathNet.Numerics.FSharp from 4.11.0 to 4.12.0
- a645dc57 dependabot-preview[bot] - Bump xunit.runner.visualstudio from 2.4.2 to 2.4.3
- 5f5a990f dependabot-preview[bot] - Bump SixLabors.ImageSharp from 1.0.0 to 1.0.1 ## Ecoacoustics Audio Analysis Software v20.8.0.63 2020-08-23

Version v20.8.0.63

Details

Compare v20.7.1.10...v20.8.0.63

- 66f7b757 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration from 3.1.6 to 3.1.7
- c5d5518d dependabot-preview[bot] - Bump MSTest.TestFramework from 2.1.0 to 2.1.2
- c7f3ebd8 dependabot-preview[bot] - Bump System.IO.Abstractions from 12.0.8 to 12.1.1

- 23240324 dependabot-preview[bot] - Bump MSTest.TestAdapter from 2.1.0 to 2.1.2
- 4f5a71a7 dependabot-preview[bot] - (#362) Bump YamlDotNet from 8.1.0 to 8.1.2 (#362)
- 4fd5f387 Anthony Truskinger - Fixes unit tests
- 2b6821f2 towsey - (#332) Fixing Unit tests
- 13968a3d towsey - (#332) Fix up octave data reduction
- 6acfa88e towsey - (#332) Shift methods to better locations
- 621ad0c9 dependabot-preview[bot] - (#358) Bump MathNet.Numerics from 4.9.0 to 4.12.0 (#358)
- 50847680 dependabot-preview[bot] - (#359) Bump FsUnit.xUnit from 3.9.0 to 4.0.1 (#359)
- 005466e7 dependabot-preview[bot] - (#348) Bump CsvHelper from 14.0.0 to 15.0.5 (#348)
- 60a76093 dependabot-preview[bot] - (#360) Bump JetBrains.Annotations from 2019.1.3 to 2020.1.0 (#360)
- 93c2703f towsey - (#332) Update FrequencyScaleTests.cs
- 01aa9be1 towsey - (#332) Fix errors in octave frequency scale
- f10ea682 towsey - (#332) New methods for rescaling linear spectrograms
- 5fa1979b towsey - Add in new spectrogram image.
- 5f26e3ff towsey - (#332) Work on Octave frequency scale
- 5031e9a1 towsey - (#332) Remove unnecssary code, change method names
- 071143c6 dependabot-preview[bot] - (#355) Bump coverlet.collector from 1.2.0 to 1.3.0 (#355)
- e843ef55 dependabot-preview[bot] - (#356) Bump Microsoft.Extensions.Configuration.Json from 3.1.6 to 3.1.7 (#356)
- bec695d2 dependabot-preview[bot] - (#347) Bump SixLabors.ImageSharp.Drawing from 1.0.0-unstable0421 to 1.0.0-unstable0456 (#347)
- 3fd6bba1 dependabot-preview[bot] - Bump ConsoleTools from 0.5.3 to 0.5.4
- 0e429f7c towsey - (#332) Update FrequencyScaleTests.cs
- adcab044 towsey - (#332) Fix generation of additional spectrogram types.
- 2a8c3e2a towsey - (#332) More epsilon/log of zero fixes.
- 6dbe9bae towsey - (#332) Incorporate epsilon in to log transform
- 7905d4f5 Anthony Truskinger - Fix CLI deprecated API errors
- e7ef6998 dependabot-preview[bot] - Bump McMaster.Extensions.CommandLineUtils from 2.5.0 to 3.0.0
- 5143bdc9 Anthony Truskinger - Attempt to make fragile tests less demanding for CI
- 7fb64df9 dependabot-preview[bot] - Bump FSharp.Core from 4.7.0 to 4.7.2
- 0a83479d towsey - (#291) Fix cluster tests
- a577f49d towsey - (#291) Fis oscillation tests
- 4dafd0cc towsey - (#291) Fixing tests for IndexCalculateTests.cs
- 2638e75f towsey - (#291) Fix tests of Long duration recordings
- 8a891005 towsey - (#291) Setting default FFT window to Hanning
- b4230447 towsey - (#332) Rework Ocatve scale when calculating acoustic indices
- 28832b52 towsey - (#332) New tests and revise tests for Octave scale
- 5908e272 towsey - (#332) More work on Octave Frequency scales
- 0a643f18 towsey - (#332) Automate calculation of Octave scale bounds and its gridlines.
- f3eb3ed8 towsey - (#332) Fix production of Octave scale spectrogram
- 7ed5281c towsey - (#332) Remove unused code
- 834fa952 towsey - (#332) Fix production of Octave spectrogram
- a3778e3d towsey - (#332) Update BaseSonogram.cs
- 9f2904f0 towsey - (#332) Change Hamming to Hanning Window
- 277a0def towsey - (#332) Mostly remove unused code and tidy comments.
- 3efb6332 towsey - (#332) Work on Unit tests for mel and octave scale spectrograms.
- 743fda81 towsey - (#332) Work on Octave scale spectrograms
- b62e20e4 towsey - (#332) Work on Mel scale spectrograms
- 0347e3f3 towsey - (#332) Create new unit test for spectral data reduction
- c16ab0d2 towsey - (#332) Add tests and rework existing tests for frequency scales
- 1598a40b towsey - (#332) Make Octave Scale methods more efficient
- 52556ac1 towsey - (#332) Shift methods to do with Mel scale
- e847cb70 towsey - (#332) Change Scale Type names

- ea1e55f2 towsey - (#332) Start setting up more detailed tests for octave frequency scales
- c20e4ca4 towsey - (#332) More work on Octave scales
- a7ca52f9 towsey - (#332) Work on Octave Spectrograms
- 055571e9 towsey - (#331) Update ActivityAndCover.cs
- 57cdec69 towsey - (#332) Produce Mel frequency scale spectorgrams
- 68a0ba2b towsey - (#332) Fix bug concerning assigning colour codes to spectrogram types
- d9688f90 towsey - (#332) Set up structure for two new spectrograms ## Ecoacoustics Audio Analysis Software v20.7.1.10 2020-07-26

Version v20.7.1.10

Details

Compare v20.7.0.4...v20.7.1.10

- 5eec1807 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration.Json from 3.1.2 to 3.1.6
- 116cbaeb dependabot-preview[bot] - Bump System.IO.Abstractions from 9.0.5 to 12.0.8
- 054747c1 dependabot-preview[bot] - Bump Moq from 4.13.1 to 4.14.5
- 2a047967 dependabot-preview[bot] - Bump SQLitePCLRaw.bundle_green from 2.0.2 to 2.0.3
- c180ebee dependabot-preview[bot] - Bump FsUnit.xUnit from 3.8.0 to 3.9.0
- eee79590 dependabot-preview[bot] - Bump Microsoft.DotNet.PlatformAbstractions from 3.1.3 to 3.1.6
- 74ae08f3 dependabot-preview[bot] - Bump Microsoft.NET.Test.Sdk from 16.6.0-preview-20200309-01 to 16.6.1
- 543b1a33 dependabot-preview[bot] - Bump xunit.runner.visualstudio from 2.4.1 to 2.4.2
- f8c25939 dependabot-preview[bot] - Bump MathNet.Numerics.FSharp from 4.9.0 to 4.11.0
- 31657330 dependabot-preview[bot] - Bump Microsoft.Extensions.Configuration from 3.1.2 to 3.1.6 ## Ecoacoustics Audio Analysis Software v20.7.0.4 2020-07-12

Version v20.7.0.4

Details

Compare v20.6.0.208...v20.7.0.4

- f3a2b112 towsey - (#292) Final changes to spectral centroid classes
- 40c843b7 towsey - (#292) Incorporate calculation of the Spectral Centroid into calculation of the Summary Indices
- a74690c9 towsey - (#292) Calculation of the Spectral Centroid
- 178af3ed towsey - (#292) Start work on calculating the spectral centroid ## Ecoacoustics Audio Analysis Software v20.6.0.208 2020-06-14

Version v20.6.0.208

Details

Compare v20.4.0.142...v20.6.0.208

- a9d94d22 Anthony Truskinger - Make GetCompositeTrack generic
- 093551ed Michael Towsey - (#321) Remove unneeded code
- 36f47b86 towsey - (#321) Finished debugging Cisticola and tests
- d14d06c4 towsey - (#321) Finished debugging Australasian Pipit and tests
- b7f74c4c towsey - (#321) Finished Boobook Owl debug and test
- ea6b3364 towsey - (#321) Finished debugging Australasian Bittern and tests
- daabd7e8 towsey - (#321) Write tests for MatrixTools
- 4433ec44 towsey - (#321) Fix unit tests after recognizer experimentation
- 6feb37c2 towsey - (#321) Update Towsey.CisticolaExilis.yml
- f38bda43 towsey - (#321) Set up the Cisticola recognizer
- 5fdd229d towsey - (#321) Update AnthusNovaeseelandiae.cs
- a29c3c28 towsey - (#321) Finish work on Australasian Pipit recognizer

- 0cefa329 towsey - (#321) Work on Austral Pipit
- 8e9f84c1 towsey - (#321) Update AnthusNovaeseelandiae.cs
- e76f2217 towsey - (#321) Work on Australiasian Pipit recognizer
- 86e471b6 towsey - (#321) Started set up of the Cisticola recognizer
- 0e718d65 towsey - (#321) Set up the Australiasian Pipit recognizer
- a9d903ec towsey - (#321) Fix AustBittern recognizer test
- ed5579f3 Anthony Truskinger - Disable koala v3 test
- 80cc5a4e Anthony Truskinger - Fix bad escape in debug message
- 55c288bd Anthony Truskinger - Fixes unit tests
- f9c33885 Michael Towsey - (#319) Changes requested by Anthony as part of Pull Request.
- 3d352631 towsey - (#319) Add documentation
- b6098eb9 towsey - (#319) Changed parameters values to ease neighbourhood test.
- 52bba1fb towsey - (#319) Add new events filter
- 08427d14 towsey - (#319) Update BoobookOwlTests.cs
- a0a57cd0 towsey - (#319) Finalise boobook recognizer
- c6e1b8b7 towsey - (#319) Work on accuracy of Boobook Owl recognizer
- 8c941431 towsey - (#319) Set up post-processing options for Boobook Owl
- c680408a towsey - (#319) Work on AUst bittern recognizer
- 68182665 towsey - (#319) Work on Boobook Owl recognizer
- b4cf5b88 towsey - (#319) Started work on test and recognizer classes for Koala V3
- 500010ea towsey - (#319) Changed namespaces for test classes
- a3cd4795 towsey - (#319) Finished work on the Aust Bittern tests
- 0219d269 towsey - (#319) Finished work on the Boobook Owl recognizer test
- 5e3743ec Anthony Truskinger - Simplified boobook owl test
- 866b9337 towsey - (#319) Set up Boobook Owl test
- 0bd49d34 Phil Eichinski - updated test to detect whether dev build or not on mac, and check the relevant file extension
- 796057e7 Anthony Truskinger - Makes working with test result artefacts easier
- f1f705de Anthony Truskinger - Fixes more unit tests
- 59b256e4 Anthony Truskinger - Fixes unit tests
- 10df97ae Anthony Truskinger - Adds more extension methods to interval
- 587d6457 Anthony Truskinger - Switches pdb types to portable
- c63aa6ce Anthony Truskinger - Add json output for new events
- d291705e Anthony Truskinger - Fix error in docs generation script
- 9ab9c455 Anthony Truskinger - Add example scripts and documentation
- 6f4eda61 towsey - (#297) Final edits after Anthony review.
- 16bc6695 Michael Towsey - Update src/AudioAnalysisTools/Events/Types/CompositeEvent.cs
- 79e8f359 Michael Towsey - Update src/AnalysisPrograms/Recognizers/Birds/BotaurusPoiciloptilus.cs
- b28d093d Anthony Truskinger - Fix spelling mistake
- 89904f96 Anthony Truskinger - Update scripting with R document
- 5cbf0174 towsey - Update PteropusSpTests.cs
- deee0172 towsey - (#297) Changes as part of code review.
- 35c74472 Marina D. A. Scarpelli - Update ScriptingwithR.md
- 3a47495b Marina D. A. Scarpelli - Create ScriptingwithR.md
- 78527438 Anthony Truskinger - Update indices.md
- 8ec397c1 towsey - (#297) Prepare config file for Koala recognizer v3
- e52411a9 Michael Towsey - (#297) tweak bittern recognizer
- bf2bd6f5 towsey - (#297) Update EventCommon.cs
- b8880db7 towsey - (#297) Fix methods for post-processing of whistle events
- 04c95d54 towsey - (#297) Write methods that detect acoustic activity in band above whistle
- 5fa7cfba towsey - (#297) Work on reducing false positives in whistle recognizers.
- 4fecb8cf Anthony Truskinger - Update README.md
- 2ddf3fbb Michael Towsey - Update Towsey.BotaurusPoiciloptilus.yml

- 09ee1756 Michael Towsey - Update BotaurusPoiciloptilus.cs
- d929e75b towsey - (#297) Final tweaks to Aust Bittern recognizer
- 5d2e8cc8 towsey - (#297) Change type casting between event types
- e1ea09d0 towsey - (#297) Wirte new class EventExtensions.cs
- a11b60c9 towsey - (#297) Set up the Australasian Bittern recognizer
- b44b7dd3 towsey - (#297) Finish work on Boobook owl recognizer
- 279c487b towsey - (#297) Koala yml file
- aec25840 towsey - (#297) Create KoalaV3.cs
- e79f6e2b towsey - Create Australasian Bittern recognizer
- c90f8fc1 towsey - (#297) Shift more frog yml files
- 4f487469 towsey - (#297) Put frog yml files in frog folder
- 24e560ce towsey - (#297) Update NinoxBoobook.cs
- d5567105 towsey - (#297) Update PezoporusOccidentalis.cs
- b5e7169d towsey - (#297) Set up the boobook owl recognizer.
- 925e2ca5 towsey - (#297) Update Track.cs
- 66d41d0b towsey - (#297) Start work on Boobook recognizer
- acbc0d1d Anthony Truskinger - Add another test case for DrawTextSafe
- 034d3966 towsey - (#297) Create NinoxBoobook.cs
- 65a980c5 towsey - (#297) Fix various bugs in the drawing of events on spectrograms.
- c2d3dc3e Anthony Truskinger - DrawTextSafe now accounts for kerning
- c193a383 Anthony Truskinger - Fix old mono reference
- 5afbf114 towsey - (#297) Reworked calculation of normalised score
- 5547a625 Anthony Truskinger - Switch all text rendering to draw text safe
- d69099c2 towsey - (#297) Update CompositeEvent.cs
- 78572a87 towsey - (#297) Fixed the drawing of labels and score bars for spectral events.
- 76573b7d towsey - (#297) Get unit tests working for spectral events
- 269edd17 towsey - (#297) Get code to build after major rework re scores and track drawing etc
- 331b713f towsey - (#297) Create four Algorithm classes for track events
- 5b351a64 Anthony Truskinger - Reverted changes to score, improved drawing methods
- e464f635 towsey - (#297) More work on calculation and normalisation of event scores.
- bca9917c towsey - (#297) Fix the normalisastion of event scores
- ffe546c9 Anthony Truskinger - Fixes faulty LFS file check error
- 7e4a8e95 towsey - (#297) Remove a duplicated draw method in SpectrogramTools
- fa3afbde towsey - (#297) Get tests working for drawing of Forward events.
- 20b29f00 towsey - (#297) Add methods to draw events on spectrograms
- 1b83c469 towsey - (#297) Fix unit test for darwing of SpectralEvent on spectrogram
- 6a35ba0c towsey - (#297) Get one test working for SpectralEvent class
- 45ed89d0 towsey - (#297) Remove old AcousticEvent constructor
- 2274b91b towsey - (#297) Create new name spaces for frog and bird recognizers.
- 9ce3db6a Anthony Truskinger - (#310) Fix build
- 30f5d30a Anthony Truskinger - Removed default docs metadata file
- 1e8e33ce Anthony Truskinger - Updates docs
- 08a8645f towsey - (#297) Start work on the new Event hierarchy
- 9a976371 Anthony Truskinger - Create apMetadata.json
- 254172d4 Anthony Truskinger - (#313) Add docs website link □
- 59219a12 Anthony Truskinger - (#313) Publishes our documentation https://ap.qut.ecoacoustics.info
- 7d529e69 towsey - (#297) Miinor changes
- a76b2487 towsey - (#297) Update Track.cs
- 07ac99f0 towsey - (#297) Update AcousticEvent.cs
- 99d3394d towsey - (#297) Fix test classes
- aaa3fa78 towsey - (#297) Iron bugs out of track extraction
- faf39a57 towsey - (#297) Correct spelling of class name

- 96a797e4 towsey - (#297) More work on setting up event types
- bff0696d towsey - (#297) Changes to event/track rendering
- 15cebf95 towsey - FUrther changes of method and class names
- 3f0e9c64 towsey - (#297) Create two new event typ0es
- 04a0808a towsey - Rename some of the track/event recognizers
- d0dbcf7f towsey - Update AcousticEvent.cs
- 7ea4b8ce towsey - Get upwards tracks drawing correctly.
- 83eb3271 towsey - (#297) Debug UnitConverters
- 1cd84120 towsey - (#297) Renamed class and methods to reflect new nomenclature
- f537c365 towsey - (#297) Getting track drawing to work after merge.
- b5d04000 towsey - (#297) Update VerticalTrackParameters.cs
- c5111ef6 Anthony Truskinger - (#310) [WIP] Tracks drawing tests begun
- 3663cbb9 towsey - (#297) Adjust unit tests to accomodate new Track hierarchy
- 1954a7da towsey - (#297) Adapt two classes to new Track hierarchy
- 728d5406 towsey - (#297) Collateral changes required while working on Track classes
- 7446c49d towsey - (#297) New work on tracks
- b4d2ecdc towsey - (#297) New work on Unit conversion.
- b3741577 Anthony Truskinger - Update README.md
- 3aaef32e Anthony Truskinger - (#310) Fixes drawing for new events
- 6666293b towsey - (#297) Complete tests for Track.cs
- de6a8225 towsey - (#297) Create new tests for tracks
- 0d8ba634 towsey - (#297) Set up test for Track class and UnitConverters class
- 689921fe Anthony Truskinger - [WIP] Add draw border inset method
- 8a1539a3 towsey - (#297) More work on Track class
- f9292e32 towsey - (#297) Create new Track class
- 43eeae19 towsey - (#297) Prepare for removal of the old SpectralTrack class
- 117994eb towsey - (#297) Augment unit converters class
- bfba2b2a Anthony Truskinger - (#310) Add tests for spectral event
- 1caf8f1d Anthony Truskinger - (#310) Continue working on new event architecture
- 0d19f9ce Anthony Truskinger - Improved instructions for setting up dev software
- e2436167 Anthony Truskinger - (#310) [WIP] Started new event archtiecture
- 420e0337 towsey - (#297) Set up methods to draw tracks and acoustic events on spectrograms.
- f2bb1e5c Anthony Truskinger - Bulk style changes
- f072e0fd Anthony Truskinger - (#311) Avoid duplicate copy of audio files
- 297729ea towsey - (#297) Set up new methods for drawing spectral tracks
- 6ce4b9af Anthony Truskinger - Fix merge error
- 3f1ff1b0 Anthony Truskinger - (#196) Undo debugging switches for CI build
- 5db59e7c towsey - (#297) Enable user to specify whether to combine events
- 50a49140 towsey - (#297) Change SpectralTrack method to return complete track data
- 82f78cdc towsey - (#297) Fix bugs in SpectralTrack class
- f6b90d9d towsey - Set up another unit test for Vertical tracks
- 8fde859b Anthony Truskinger - Add gitter chat bade
- 3c55a34a towsey - (#297) Update VerticalTrackParameters.cs
- 6f2c6160 towsey - (#297) Fix out of range bugs
- 1fa84b16 towsey - (#297) Get new VerticalTrack component recognizer working
- 4a167586 towsey - (#297) Update SpectralPeakTrackParameters.cs
- 0dcdb43b towsey - (#297) Rework the old SpectralTrack class.
- d5bf5d76 towsey - (#297) Write new recognizer for VerticalTrack components
- 2658be2b towsey - (#297) Write method to combine proximal similar events
- a43134a9 towsey - (#297) Finish unit test for click events
- f941679f towsey - (#297) Write and Test Click recognizer
- a81b9065 towsey - (#297) Update GenericRecognizer.cs

- cc44224d towsey - (#297) Set up new event type - (#297) the Click
- 17bf9f4e towsey - (#297) Define a set of event types
- 5ae9cc50 towsey - (#297) Retain track info to store in acoustic event
- b6037033 towsey - (#297, #297) Issue#297 Tweak the spectral peak track recognizer
- 7ea052d5 towsey - (#297) Work on spectral peak track recognizer
- 827a5ba9 towsey - (#297) Update HarmonicParameters.cs
- 4a3916ec towsey - (#297, #307) Find bug re window overlap
- 2b9fa60c towsey - (#297) Work on the harmonic recognizer
- fca8484c Anthony Truskinger - Update PULL_REQUEST_TEMPLATE.md
- 12398bb6 towsey - (#297) Set up generic recognizer for stone curlew call
- ad7b7dc1 towsey - (#297) Establish unit test for generic recognition of harmonics
- 767dd013 towsey - (#297) Set up artificial spectrogram
- e2df1984 Charles Alleman - (#305) Created PR Template (#305)
- 2bbb60bc Anthony Truskinger - (#301) Cleaned up unconventional code
- 4d6e3c7e Anthony Truskinger - Cleaned up event test
- 6d970e39 towsey - (#300) Fixed test of drawing events
- af4813e6 towsey - (#300) Fix anti-aliasing when drawing event
- 8a5a6f0e towsey - (#300) Changes as requested by Anthony
- 6de76cc7 towsey - (#300) Update AcousticEvent.cs
- 1df70c59 towsey - (#300) Update AcousticEvent.cs
- 4cd48e80 towsey - (#300) Remove a call to one of my do-it-yourself test methods.
- 71a53b8d towsey - (#300) Fixing bugs in merging of acoustic events
- 79620e68 towsey - (#300) Unit tests for drawing events on sonograms
- 5a3f9afe towsey - (#300) Write one unit test for overlay of hit scores
- b05bbb0c towsey - Update SpectrogramSettings.cs ## Ecoacoustics Audio Analysis Software v20.4.0.142 2020-04-07

Version v20.4.0.142

Notes

MAJOR CHANGE: Now runs on .NET Core

They way you use AnalysisPrograms will not be the same if you are on Linux or MaxOSX. You will no longer need to install mono, or prefix commands with `mono`. Please see our installing documentation for more details!

Details

Compare v20.2.0.99...v20.4.0.142

- 1a273de9 Anthony Truskinger - (#196) Adds GitHub service connection to pipelines release task
- d1629bc9 Anthony Truskinger - (#196) Yet another method of resolving the branch name ▯
- 17847379 Anthony Truskinger - (#196) Change git branch detection in git version script again
- 8f2315ca Anthony Truskinger - (#196) Add debugging information to git version script
- 4176273d Anthony Truskinger - (#196) Debug release script
- b6519400 Anthony Truskinger - (#196) Fix spelling mistake
- 1e77ebcc Anthony Truskinger - Add dependency betwen release jobs
- 90e70e30 Anthony Truskinger - Add names to release jobs and change release condition to use those names
- de17dc6a Anthony Truskinger - Fix bad powershell like
- 24c0b81f Anthony Truskinger - (#196) fix line break in github credentials
- ecfd17c1 Anthony Truskinger - Revert changes to ACT wait/pump
- 791c44e3 Anthony Truskinger - Battle harden AC tests
- 6d2d3ed5 Anthony Truskinger - (#196) Remove redundant artifact publish step
- 4b8bd727 Anthony Truskinger - (#196) Attempt to get release stage to Work
- bd652094 Anthony Truskinger - Fixes for unit tests
- 45e6232b Anthony Truskinger - Fix bad option in publish test results

- 2fd130ec Anthony Truskinger - (#169) Fix arguments for git_version script
- b1cf305a Anthony Truskinger - (#196) Adds repeat test attribute for flaky tests
- db5118ad Anthony Truskinger - (#196, #196) [WIP] Add release stage to pipeline
- 53e9efbe Anthony Truskinger - (#196) Attempts to save version vars for release stage
- e2a5a6bd Anthony Truskinger - (#196) Try renabling AED code coverage
- 2161ce36 Anthony Truskinger - (#196) Fix bad variable
- f5f0b330 Anthony Truskinger - (#196) Fix unblanced parantheses
- 83b6d20c Anthony Truskinger - (#196) Try and enhance code coverage
- 66651010 Anthony Truskinger - (#196) Up async wait for test, remove self contianed arg for fx-dependent build
- 074e24fd Anthony Truskinger - (#196) One last tweak to tests
- fd5162f5 Anthony Truskinger - (#196) More CI Fixes
- 7cf239ad Anthony Truskinger - (#196) Fix tests for Mac CI
- 4432180d Anthony Truskinger - (#196) Add missing if to azure pipelines yaml
- de523ecc Anthony Truskinger - (#196) Fix bad null literal in buid yaml
- 2a3f6545 Anthony Truskinger - (#196) Removes support for MP3 for SoX on OSX
- a3091c02 Anthony Truskinger - (#196) Remove appveyor references
- e196c076 Anthony Truskinger - (#196) Bump SDK Version
- ab7f71cb Anthony Truskinger - (#196) Adds LFS file checkout check to build
- da575741 Anthony Truskinger - (#196) Fix string quoting issues.
- 69f4d483 Anthony Truskinger - (#196) Adds random MP3 C libs to SoX in attempt to get it work on OSX
- dd28a847 Anthony Truskinger - (#196) Adds framework dependent build
- 8abafea9 Anthony Truskinger - (#196) Add extra logging to log file cleanup
- 3381ee93 Anthony Truskinger - (#196) Disabel musl tests and and more timeouts
- 50fe4497 Anthony Truskinger - (#196) Attempt to get more information about failing tests
- ebfe8e86 Anthony Truskinger - (#196) Fixes runtime identifer code generation
- d32268d3 Anthony Truskinger - (#196) Fixes for CI and tests
- ce8b1996 Anthony Truskinger - (#196) Fix mistake in azure pipeliens yml
- 7da16240 Anthony Truskinger - (#196) Ensure wavpack is installed on azure
- c8c8a09c Anthony Truskinger - (#196) Fix missing rid arguments, add fallback for rid source
- 0358b8c6 Anthony Truskinger - (#196) Add unit tests for missing wavpack binary
- 6d122983 Anthony Truskinger - (#196) Fix bad runtime identifer source for CheckExecutePermission test
- 4c02f917 Anthony Truskinger - (#196) Fixes issue with publish step
- 6673ed40 Anthony Truskinger - (#196) Fixes stack overflow in AED data generator
- 7a5a13e1 Anthony Truskinger - (#196) Fix missing arg on RuntimeIdentifierSpecificDataTestMethod
- d0f4d308 Anthony Truskinger - (#196) Add SoX install script for linux CI
- 62b03206 Anthony Truskinger - (#196) Fix style warning in SoX Audio utility
- 3c50c3cf Anthony Truskinger - (#196) More cross-platform test fixes
- 9e78583c Anthony Truskinger - (#196) Fix more tests
- 14859ffa Anthony Truskinger - (#196) Fix template syntax
- 0bd8eca7 Anthony Truskinger - (#196) Fix test, attempt different job factorization strategy
- d8cb6b4b Anthony Truskinger - (#196) Clean up solution, remove some mono refs in docs
- 7534b47b Anthony Truskinger - (#196) Fixes various cross platform issues
- 3269c785 Anthony Truskinger - (#196) Fixes tests for SelfContained/platform-specific builds
- 0ba342ab Anthony Truskinger - (#196) Attempt to fix CI
- b5a43e8c Anthony Truskinger - (#196) Attempts to fix tests on linux
- 2b655643 Anthony Truskinger - (#196) Fix AED tests
- 628d95ce Anthony Truskinger - (#196) Fixes ribbon plot tests
- 310b38a0 Anthony Truskinger - (#196) Fixes unit test with hardcoded windows path
- 0a58db50 Charles Alleman - (#196) Solved AED.Test error on test
- fbe9a784 Anthony Truskinger - (#196) Simplified path diagnostics test for directories with trailing spaces
- 894012b6 Charles Alleman - (#304, #196) Process extension (#304)
- 9b444ff0 Anthony Truskinger - (#196) Fixes stack overflow exception in ReadSpectrogram

- 25a77b21 Anthony Truskinger - (#196) Fixes process runner tests
- 11c9199a Anthony Truskinger - (#196) Fixes tiler tests
- 75901282 Anthony Truskinger - (#196) Fixes path diagnostics for Linux
- 71ee22eb Anthony Truskinger - (#196) Fixes PathUtils tests for Linux
- f952de36 Anthony Truskinger - (#196) Fixes font fallback option
- cf15a270 Anthony Truskinger - (#196) Fix Assembly load bug resulting in CSV class maps not getting registered
- 85c5ec50 Anthony Truskinger - (#196) [WIP] Add negatated platform specific test option
- 01405d0f Anthony Truskinger - (#196) Removes Zio
- 12f0324d Anthony Truskinger - (#196) Finishes adding support for Roboto font
- bd7db260 Anthony Truskinger - (#196) Bump .NET SDK to 3.1.200
- 61be2eca Anthony Truskinger - (#196) [WIP] Bundles a font to use with AP.exe
- ac009d81 Anthony Truskinger - (#196) Updates namespaces on FSharp projects
- 5f08faa9 Anthony Truskinger - (#196) Initial changes to get build working on Linux
- 6e806a40 Anthony Truskinger - (#196) Adjusts timeouts for tests on CI
- 6876db51 Anthony Truskinger - (#196) Adds executable check to exe reosolution
- 922027db Anthony Truskinger - (#196) Cleaned up solution
- a6117a6c Anthony Truskinger - (#196) Up execution time for test
- 822c3ae5 Anthony Truskinger - (#196) Fix permissions of included binaries
- 1dca915b Anthony Truskinger - (#196) Add path diagnostics method
- 671d93f3 Anthony Truskinger - (#196) Battle hardening ProcessRunner for .NET Core
- ec4502fc Anthony Truskinger - (#196) Fixing platform breaking unit tests
- b310984c Anthony Truskinger - (#196) Fix failing unit tests
- 6dd705b8 Anthony Truskinger - (#196) More hardening for solution root helper
- b01a1b0e Anthony Truskinger - (#196) fix no build switch
- 1dad15c6 Anthony Truskinger - (#196) Attempt to get coverlet working, add more diagnostics to test runtime
- 6d38388d Anthony Truskinger - (#196) Changes unit test solution directory finder method
- b1bca4a0 Anthony Truskinger - (#196) Fixes bad test invocation for azure CI
- 103ad4d5 Anthony Truskinger - (#196) Removed and deprecated no longer used audio tools
- 433f4bfa Anthony Truskinger - (#196) [WIP] refactoring audio utils
- 2735d443 Anthony Truskinger - (#196) Flesh out azure build
- 69dc04da Anthony Truskinger - (#196) Change runsettings configuration
- c7c9d28f Anthony Truskinger - (#196) Vastly simplifies ap version metadata generation
- a3bc20ea Anthony Truskinger - (#196) More work on CI
- 60dc9d33 Anthony Truskinger - (#196) Removes auto attach feature
- a0e3d58a Anthony Truskinger - (#196) Conntinued work on CI
- 02f4cae2 Anthony Truskinger - (#196) More work on CI
- 5a328826 Anthony Truskinger - (#196) First (failing attempt) to make Azure Pipelines work
- a0cddca0 Anthony Truskinger - Update contributing guide
- 752eaa46 Anthony Truskinger - Fix clean missing project import reference
- a0bc5ab0 Anthony Truskinger - Fixes all unit tests for .NET Core
- 47ee8ff7 Anthony Truskinger - Fixed metadata generation
- 0d08e64a Anthony Truskinger - Updates ImageSharp libraries
- 9ca5467e Anthony Truskinger - Cleaned up solution file
- 64c2913e towsey - Mostly cosmetic changes.
- 7b4ef823 Anthony Truskinger - Refactors Audio2Sonogram
- 4d67dbf8 Anthony Truskinger - Improves image testing and test output directories
- 7670fe75 Anthony Truskinger - Adds textual pattern test image generation to TestImage
- 4ce35449 Anthony Truskinger - Added wrapper methods for no antialias code
- 4e7b4988 towsey - Fix merging of overlapped acoustic events
- ce503316 towsey - Fix spectrogram image generation from GenericRecognizers
- 5e51bb7a Anthony Truskinger - Wrote tests for combine images methods
- 1117d4e9 Anthony Truskinger - Refactored drawing methods and extensions

- 6c52072e towsey - Trying to get spectrogram images working
- 5c085d9e towsey - Update Audio2Sonogram.cs
- dadf0bfb towsey - Update ImageTools.cs
- df5606d7 Anthony Truskinger - Refactored solution build and tests
- 3f195bec towsey - Fixed various post merge issues
- 1f33214a Anthony Truskinger - Fixes a bug in ImageTools
- bb1c4c67 Anthony Truskinger - (#241) Refactored .NET Core solution
- 248163f6 Anthony Truskinger - Removes outdated packages directory references
- 2c134cbd Anthony Truskinger - Fixes errors from previous merge commits
- d880a997 Anthony Truskinger - Adds download URL for .NET Core SDK
- 63e6299b Anthony Truskinger - (#289, #288, #158, #159) Almost completes port to .NET Core
- ecad1de1 Anthony Truskinger - (#158, #159) .NET Core and ImageSharp: fixing unit tests
- fee81bd6 Anthony Truskinger - (#158, #159) Converting to .NET Core and ImageSharp
- a4d25e9d Anthony Truskinger - (#159, #158) Fixing .NET Core incompatibilities
- 79d7ae03 Anthony Truskinger - (#159) Converted Solution to PackageReferences format ## Version v20.2.0.99 - QUT Ecoacoustics Workshop 2020 Version (14/02/2020)

Notes

This version of AP was bundled in the USB sticks used in the tutorial of the 2020 QUT Ecoacoustics Workshop.

IMPORTANT: This will be the last stable version of AP.exe to be released for some time. The project is changing its fundamental architecture (see #159) from the .NET Framework to .NET Core.

After this point, mono will no longer be required to use AP.exe on Mac and Linux and instead, standalone platform-independent packages will be produced.

This will be an exciting time for us but be wary, there will be dragons ahead.

Details

Compare v19.12.0.1...v20.2.0.99

- 8bc90089 Anthony Truskinger - Adds useful defaults to ConcatenateIndexFiles command
- ae7501e2 Anthony Truskinger - No not do appconfig name check on mono
- 398bd4f6 Michael Towsey - Update UseModel.cs
- c39bb95a Anthony Truskinger - Updated default BlueEnhanceParameter in Towsey.Acoustic.yml
- 08cb0fe6 Michael Towsey - Work on the harmonic recognizer
- 4571c2de Michael Towsey - a little work on harmonic recognizer
- 4955f30e Michael Towsey - Update ConcatenationTests.cs
- 1dfaa928 Michael Towsey - Get Generic recognizer tests working
- 39e6a38d Michael Towsey - Set up config for doing test of GenericRecognizers
- f9d4f9dc Michael Towsey - Work on Harmonic and Whistle recognizers
- 9511e918 Michael Towsey - Work on Whistle recognizer
- feb72a07 towsey - Change config files
- 5c99a195 towsey - Start on Oscillation recognizer
- 21f89d33 Anthony Truskinger - Fixes window overlap calculation for generic recognizer
- 5b1e4002 Anthony Truskinger - Improves method of registering yaml tag types
- 072bfcf5 towsey - Change two config file names
- 5a37019b Anthony Truskinger - Ensures GenericRecognizerConfig static constructor is invoked
- 4b6a453f towsey - Update LDSpectrogramRGB.cs
- 663b9ef3 Anthony Truskinger - Finished Generic recognizer and tests
- 00086b0b Anthony Truskinger - [WIP] Finished rewwriting generic recognizer
- a2db97af Anthony Truskinger - [WIP] Refactoring generic recognizer
- 262dde4e Michael Towsey - insert configs into generic recogniser classes

- 0d255d4a Michael Towsey - Set up classes
- 2bfa8895 Michael Towsey - Set up Generic recognizers
- c5ab40d3 Michael Towsey - Conflict resolution
- 701b9053 Michael Towsey - Updated image sharp packages
- 66d3caea towsey - Update ConcatenateIndexFiles.cs
- 1e0079e8 towsey - Attempt to fix bug in UseModel.cs
- 59672cc7 towsey - Read parameters from the config.yml file.
- 996d5648 towsey - Try versions of difference spectrogram
- b1ba0c8c towsey - Enable recording name to be passed to spectrograms.
- 664597ad towsey - Set up generation of Cepstral spectrogram.
- 0bf4f6c2 towsey - Create Towsey.SpectrogramGenerator.yml
- 3c544d93 towsey - Add TODO notes to two classes
- 5752c027 towsey - Replace config dictionary with typed dictionary
- de70d9b9 towsey - Add back second time scale to spectrograms.
- 85f32ce3 towsey - Trial removal of obsolete class
- 99c4bfce towsey - Get four spectrograms displaying correctly
- 8bbd8722 towsey - Further work to get more spectrograms
- 9e72c71f towsey - Get Waveform image working.
- d39e9820 towsey - Refactor the method to produce spectrograms
- 1c26d469 towsey - Fix unit tests
- 18c7bb6c towsey - Made correction of blue colour an optional parameter
- e69360fc towsey - Adjusted display of indices rendered in blue.
- 9e523266 towsey - Update LDSpectrogramRGB.cs
- 9ad5c900 towsey - Work done by Anthony to implement a new Results class for 6 indices.
- 376a5039 towsey - Update UseModel.cs
- 7a0364fa towsey - Update Sandpit.cs
- e095850a towsey - Add testing capcability to the Build/Make templates.
- d9db4ae0 towsey - Implement entry point for building content description models
- c15800b1 Michael Towsey - Create new SpectralIndexvalues class
- c342e0b8 Michael Towsey - Shift Content description analysis to main analysis loop
- 79f939b2 Michael Towsey - Update Towsey.TemplateDefinitions.json
- be0abf9b Anthony Truskinger - Code review for content description
- 89355ce8 Michael Towsey - add documentation
- c8f22515 Michael Towsey - Update TemplateManifest.cs
- e1465f57 Michael Towsey - Delete unnecesary files
- 129fb7f4 towsey - Fix up ContentSignatures.cs
- 51b29d88 towsey - derive more params from config file
- b8a84bcb towsey - Access params from the config.yml file
- 9c13b79e towsey - Finally get content description working in IAnalyzer2 environment.
- 516abf0b towsey - integrating Content Descirption into IAnalyzer2.
- a52c9b02 towsey - Content description now works using IAnalyzer2
- b3e568e9 towsey - Set up IAnalyzer2 system ready for Content description
- d765d942 towsey - More work on content description
- e7a4052b towsey - Set up class to calculate only six spectral indices
- 42b429ad towsey - Set up Content Description as IAnalyzer2.
- 17a56e35 towsey - Testing that manifest to tesmplates works
- 82f03295 towsey - resharper stuff
- a1fbbc9b towsey - Improve score normalisation
- d471feb8 towsey - Experiment with different score normalisations
- 7f1c5041 towsey - Complete refactoring of code to separate manifests from template definitions.
- d683f115 towsey - Set up Json reading and writing of template files
- 6bd20d29 towsey - Issue #252 Refactor code as per Anthony suggestions

- cb7f80a7 towsey - Work on template creation
- 418032d5 towsey - Issue #252 Start work on template editing
- 1b8e7764 towsey - Start work on Template Creation
- 1ac77f0f towsey - Issue #252 Finalise three content algoithms
- 0132e83a towsey - Issue #252 Set up new data structures for content description
- 6a319be1 towsey - Added visualisation methods to Plot.cs
- e5645d13 towsey - Created DataProcessing class plus two new test classes
- 7747fd56 towsey - Add another way of calculating distance
- 45aeb8be towsey - Set up two new content types, Morning Chorus and Silver Eye
- 2e4c4c1a towsey - Set up new classes to recognise acoustic types
- c2ec082a towsey - Set up wind and rain detection methods
- d2511f4c towsey - Building more of code structure for content descirption
- 4847c108 towsey - Continue setting up Content description framework.
- 99a27ff7 towsey - Set up methods for validating the output from content description classes
- d1d9f9f0 towsey - Write methods to read in acoustic index spectrograms.

- 857b9f72 towsey - Set up ConentDescription project

Ecoacoustics Audio Analysis Software v19.12.0.1 (09/12/2019)

Version v19.12.0.1

Compare v19.12.0.5...v19.12.0.1

- b93a667 towsey - Changes to standard spectrogram config file

Ecoacoustics Audio Analysis Software v19.12.0.5 (02/12/2019)

Version v19.12.0.5

Compare v19.11.1.1...v19.12.0.5

- b45652f towsey - Update TestAnalyzeLongRecording.cs
- 1907094 towsey - Fix five more broken tests
- 4dd38e0 towsey - Removed references to R3D identified by Anthony

- 96354c9 towsey - removed calculation of R3D

Ecoacoustics Audio Analysis Software v19.11.1.1 (25/11/2019)

Version v19.11.1.1

Compare v19.11.0.30...v19.11.1.1

- 781d0a3 towsey - Update Sandpit.cs

Ecoacoustics Audio Analysis Software v19.11.0.30 (18/11/2019)

Version v19.11.0.30

Compare v19.10.0.5...v19.11.0.30

- 33c5094 Michael Towsey - Issue-#238 Fix failed tests for flying fox

- cc8b834 Anthony Truskinger - Update CONTRIBUTING.md
- 856da7a Michael Towsey - Fixed unit test
- 0a5ec7d Michael Towsey - Update AcousticComplexityIndex.cs
- b31598b towsey - More work on Oscillation recognizer
- a0e643b towsey - Reworking the oscillation detection component of Flying Foxrecognizer
- e807743 towsey - Update PulseTrain.cs
- 1228ff3 towsey - Update SonogramTests.cs
- 1228b46 towsey - Finish Unit tests for Flying FOx class
- 70caaaf towsey - Update SonogramTests.cs
- 2f8947d towsey - More work on Pteropus recogniser tests
- 93cb93b towsey - More work on FlyingFox tests
- 67e023b towsey - Set up unit tests for the flying fox recogniser
- 03a304d towsey - More changes to FF code
- 1d1c552 towsey - Finished response to requests from Anthony
- 877703f towsey - Responding to comments from Anthony
- 10ad62a towsey - Fine adjustment of parameters
- 23429bd towsey - Update PteropusSpecies.cs
- 007d150 towsey - Attempt to implement pulase train detection
- 6e6cce8 towsey - Add wingbeat profile to Flying Fox
- 7a2c5e1 towsey - Penultimate work on FF recogniser
- 5e5016e towsey - Refactor acoustic events method
- a07ae44 towsey - cleaned up the PreopusSpecies class
- 3a5db0a towsey - New method to find events
- d8813c2 towsey - Further work on the flying fox recogniser.
- d7ac462 Michael Towsey - Begin work on flying fox recogniser
- 3f24758 towsey - Set up Flying Fox Recogniser classes

- 56c0725 towsey - Projects & Packages required to get recogniser working

Ecoacoustics Audio Analysis Software v19.10.0.5 (21/10/2019)

Version v19.10.0.5

Compare v19.10.0.3...v19.10.0.5

- e96fb1e dependabot-preview[bot] - Bump Microsoft.Extensions.FileProviders.Physical
- 733ef00 dependabot-preview[bot] - Bump System.Diagnostics.DiagnosticSource from 4.3.0 to 4.6.0
- 2d51909 dependabot-preview[bot] - Bump xunit.extensibility.core from 2.2.0 to 2.4.1
- 4164f40 dependabot-preview[bot] - Bump SixLabors.ImageSharp.Drawing from 1.0.0-beta0007 to 1.0.0-dev000893

- e6a208d dependabot-preview[bot] - Bump Zio from 0.3.6 to 0.7.4

Ecoacoustics Audio Analysis Software v19.10.0.3 (07/10/2019)

Version v19.10.0.3

Compare v19.9.2.18...v19.10.0.3

- bea4285 Anthony Truskinger - Allow fo detection of AudioMoth dates in result file names
- ce28732 Anthony Truskinger - Added FxCopAnalyzers package

- 3657e16 Anthony Truskinger - Added support for parsing AudioMoth V1 dates

Ecoacoustics Audio Analysis Software v19.9.2.18 (30/09/2019)

Version v19.9.2.18

Compare v19.9.1.3...v19.9.2.18

- ea59702 Anthony Truskinger - Fix syntax error from merge conflict
- ded36f9 towsey - Update ACI.bin
- 2c4599a towsey - Fix problem ACI value in top freq bin
- 53def22 towsey - Fix Unit tests for Concatenation
- 498c468 towsey - More experiments with rendering of zooming spectrograms
- 37cf217 towsey - Experiment with combined BGN index
- b8b39aa towsey - Check zooming spgm code is working OK
- 2c6578b Michael Towsey - Fix ColourMap weightings
- f9ed53d Michael Towsey - Work on ZoomTiledSpectrograms
- 3f9fe4f Michael Towsey - new method to load index matrices
- 4c55275 Michael Towsey - Shift directory SearchOption one level higher
- 1d78306 Michael Towsey - Edit code to adjust to config changes
- b6f64ca Michael Towsey - edit IndexProperties files
- 52cc26e Michael Towsey - Fix treatment of degenerate distributions
- d2e3377 Michael Towsey - Experiments with concatenate

- 9f3455c Michael Towsey - Deleted config files no longer relevant

Ecoacoustics Audio Analysis Software v19.9.1.3 (23/09/2019)

Version v19.9.1.3

Compare v19.9.0.3...v19.9.1.3

- 4462211 Anthony Truskinger - Fixed errant tilde
- e579aba Anthony Truskinger - Updated Plot.cs with changes from content-description branch

- b141a00 Anthony Truskinger - Updated image sharp and other dependencies

Ecoacoustics Audio Analysis Software v19.9.0.3 (16/09/2019)

Version v19.9.0.3

Compare v19.9.0.0...v19.9.0.3

- d4fe77f Anthony Truskinger - Cleaned up old string extension methods

- 92c371e Anthony Truskinger - Test and patch for short-name-app-config bug

Ecoacoustics Audio Analysis Software v19.9.0.0 (02/09/2019)

Version v19.9.0.0

Compare v19.8.2.5...v19.9.0.0

- 7542295 Anthony Truskinger - temporarily disable azure pipelines
- dd04350 Anthony Truskinger - minor docs change
- d2b94f0 towsey - Responding to comments from Anthony
- f6537bf towsey - Fine adjustment of parameters
- a1eb4e7 towsey - Update PteropusSpecies.cs
- f1a5a9f towsey - Attempt to implement pulase train detection
- 14a3214 towsey - Add wingbeat profile to Flying Fox
- 2cf3847 towsey - Penultimate work on FF recogniser
- f74098d towsey - Refactor acoustic events method
- aa85432 towsey - cleaned up the PreopusSpecies class
- c86caaa towsey - New method to find events
- d123d30 towsey - Further work on the flying fox recogniser.
- 414b392 Michael Towsey - Begin work on flying fox recogniser
- 6481e0e towsey - Set up Flying Fox Recogniser classes
- 4bb818c towsey - Projects & Packages required to get recogniser working
- bd00af0 Anthony Truskinger - Missing changes to previous fix

- efc4a44 Anthony Truskinger - Adds test for BARLT metadata parsing

Ecoacoustics Audio Analysis Software v19.8.3.17 (29/08/2019)

Version v19.8.3.17

Compare v19.8.2.5...v19.8.3.17

- 7542295 Anthony Truskinger - temporarily disable azure pipelines
- dd04350 Anthony Truskinger - minor docs change
- d2b94f0 towsey - Responding to comments from Anthony
- f6537bf towsey - Fine adjustment of parameters
- a1eb4e7 towsey - Update PteropusSpecies.cs
- f1a5a9f towsey - Attempt to implement pulase train detection
- 14a3214 towsey - Add wingbeat profile to Flying Fox
- 2cf3847 towsey - Penultimate work on FF recogniser
- f74098d towsey - Refactor acoustic events method
- aa85432 towsey - cleaned up the PreopusSpecies class
- c86caaa towsey - New method to find events
- d123d30 towsey - Further work on the flying fox recogniser.
- 414b392 Michael Towsey - Begin work on flying fox recogniser
- 6481e0e towsey - Set up Flying Fox Recogniser classes
- 4bb818c towsey - Projects & Packages required to get recogniser working
- bd00af0 Anthony Truskinger - Missing changes to previous fix

- efc4a44 Anthony Truskinger - Adds test for BARLT metadata parsing

Ecoacoustics Audio Analysis Software v19.8.2.5 (26/08/2019)

Version v19.8.2.5

Compare v19.8.1.1...v19.8.2.5

- 68d58fe towsey - Update LdSpectrogramRibbons.cs
- 70277d4 towsey - Clean up some build problems
- 734954b towsey - Set up methods for reading indices from Spectrogram ribbon images
- d0b228d towsey - Write methods to read ribbon images

- 9b40034 Anthony Truskinger - Update faq.md

Ecoacoustics Audio Analysis Software v19.8.1.1 (12/08/2019)

Version v19.8.1.1

Compare v19.8.0.1...v19.8.1.1

- abb00dd Anthony Truskinger - Added notes on chunk size choices

Ecoacoustics Audio Analysis Software v19.8.0.1 (05/08/2019)

Version v19.8.0.1

Compare v19.7.2.3...v19.8.0.1

- 0ba6fb0 Anthony Truskinger - Update installing.md

Ecoacoustics Audio Analysis Software v19.7.2.3 (29/07/2019)

Version v19.7.2.3

Compare v19.7.1.3...v19.7.2.3

- f47ea18 Anthony Truskinger - Fixes spelling mistakes in bug template ðŸ™„
- 6aa4089 Anthony Truskinger - Update bug_report.md

- 79a2f00 Anthony Truskinger - Set up CI with Azure Pipelines

Ecoacoustics Audio Analysis Software v19.7.1.3 (22/07/2019)

Version v19.7.1.3

Compare v19.7.0.1...v19.7.1.3

- 5f1d30f Anthony Truskinger - Updated bug report issue template
- 69ee220 Anthony Truskinger - Create a question issue template

- 089e148 Anthony Truskinger - Fixes System.Numerics.Vectors not loading on some systems and produces better errors for reflection loading exceptions (#244)

Ecoacoustics Audio Analysis Software v19.7.0.1 (08/07/2019)

Version v19.7.0.1

Compare v19.6.1.1...v19.7.0.1

- b1045dd Anthony Truskinger - Adds documentation to SaveIntermediateCsvFiles config option

Ecoacoustics Audio Analysis Software v19.6.1.1 (01/07/2019)

Version v19.6.1.1

Compare v19.6.0.1...v19.6.1.1

- 30eb5bd Anthony Truskinger - Fixes culture formatting bug for SoX commands (#243)

Ecoacoustics Audio Analysis Software v19.6.0.1 (24/06/2019)

Version v19.6.0.1

Compare v19.5.1.1...v19.6.0.1

- 484f922 Anthony Truskinger - Add notes on procuring Visual Studio

Ecoacoustics Audio Analysis Software v19.5.1.1 (13/05/2019)

Version v19.5.1.1

Compare v19.5.0.1...v19.5.1.1

- 6973fd5 Anthony Truskinger - More ap_download script updates

Ecoacoustics Audio Analysis Software v19.5.0.1 (06/05/2019)

Version v19.5.0.1

Compare v19.4.1.4...v19.5.0.1

- 7dc4f0e Anthony Truskinger - Updated download_ap.ps1 to accept github api token

Ecoacoustics Audio Analysis Software v19.4.1.4 (29/04/2019)

Version v19.4.1.4

Compare v19.4.0.1...v19.4.1.4

- ff0e123 Anthony Truskinger - Adds unit tests for ribbon plots
- 8f9e727 Anthony Truskinger - [WIP] Finished ribbon plots
- 2005bc8 Anthony Truskinger - [WIP] Almost got ribbons working

- 51f27bd Anthony Truskinger - [WIP] Initial work for drawing ribbon plots

Ecoacoustics Audio Analysis Software v19.4.0.1 (08/04/2019)

Version v19.4.0.1

Compare v19.3.3.39...v19.4.0.1

- 3c64b3c Anthony Truskinger - Updated fs compiler tools

Ecoacoustics Audio Analysis Software v19.3.3.39 (01/04/2019)

Version v19.3.3.39

Compare v19.3.2.1...v19.3.3.39

- 3bc6a90 Michael Towsey - Mostly resharper stuff
- 934f466 Michael Towsey - Change config file
- 5630edd Michael Towsey - Add new property to AcousticIndicesConfig class
- 03f7190 Anthony Truskinger - Upped timeout for log clearing test
- f25b053 Michael Towsey - Adjust tests affected by Issue #217
- 8c4a0fb Michael Towsey - Small changes to Sandpit
- c7c2a17 Michael Towsey - Update SummaryIndexValues.cs
- 910ff24 Michael Towsey - Final commit for issue #217
- 21d8925 Michael Towsey - remove option to include Sunrise data
- b9fe5db Michael Towsey - Remove translation dictionary
- 369d3a3 Michael Towsey - Removed unnecessary Indices & properties
- 3d949e0 Michael Towsey - Split the IndexProperties file in two
- e35260d Anthony Truskinger - Fixed minor concat bugs
- cedad4c Michael Towsey - Update ConcatenationTests.cs
- 252c58f Michael Towsey - Changed way mode of distribution calculated
- 57de07e Michael Towsey - Deal with case where histogram mode = 0.0
- d0228f1 Michael Towsey - Refactor code which draws histograms
- d690c35 Michael Towsey - Rework rendering of Concatenated Indices
- 047e36c Michael Towsey - Four minor changes for checking purposes
- 9691546 Michael Towsey - Revert "Update Sandpit.cs"
- c5fa3cb Michael Towsey - Update Sandpit.cs

- 668e48b Michael Towsey - Update GapsAndJoins.cs

Ecoacoustics Audio Analysis Software v19.3.2.1 (25/03/2019)

Version v19.3.2.1

Compare v19.3.1.9...v19.3.2.1

- 6cbf3da Anthony Truskinger - Fixes spelling mistakes in CLI help

Ecoacoustics Audio Analysis Software v19.3.1.9 (18/03/2019)

Version v19.3.1.9

Compare v19.3.0.5...v19.3.1.9

- 64e0f98 Anthony Truskinger - Adds checkes for critical DLLs
- e3317bd Anthony Truskinger - Use a nuget version of System.ComponentModel.Annotations
- 3d0d284 Anthony Truskinger - More fixes to download script

- 5463137 Anthony Truskinger - Fully disables CLIpager
- b89a5d5 Anthony Truskinger - Tried to improve check environment
- ee00f00 Anthony Truskinger - Fixes log naming bug
- 478f910 Anthony Truskinger - Disables CLI pager

- b6a34a5 Anthony Truskinger - Improves downloader script

Ecoacoustics Audio Analysis Software v19.2.2.1 (25/02/2019)

Version v19.2.2.1

Compare v19.2.1.10...v19.2.2.1

- e336ab8 Anthony Truskinger - Fixes a bug in CLI parsing

Ecoacoustics Audio Analysis Software v19.2.1.10 (18/02/2019)

Version v19.2.1.10

Compare v19.2.0.90...v19.2.1.10

- f998a4f Anthony Truskinger - Fixes extra bad new line in header
- 799a653 Anthony Truskinger - Splits up copyright text
- 9e9e56d Anthony Truskinger - More logging patches
- c96ea6e Anthony Truskinger - Fixes CLI duplicate value parser bug
- 3e1a67a Anthony Truskinger - Refactored static Logging class
- ce80cbc Michael Towsey - Fixes and closes Issue #170 concat joins
- 981696d Michael Towsey - Concat crashing Issue #170

- 2917519 Anthony Truskinger - Fixes CLI pager bug

Ecoacoustics Audio Analysis Software v19.2.0.90 (11/02/2019)

Version v19.2.0.90

Compare v19.2.0.3...v19.2.0.90

- e47eeff Anthony Truskinger - Fix unit tests from previous merge
- 1372bd6 Anthony Truskinger - Updates mstest and adapters
- 8259fee Anthony Truskinger - Improves perf for zoom tests
- 2ad4da1 Mahnoosh Kholghi - removed unnecessary references
- 557e5de Mahnoosh Kholghi - removing some unnecessary references
- d7e5a7c Mahnoosh Kholghi - added comments to the spectral peak tracking method
- 29af151 Mahnoosh Kholghi - fixed bugs in new peak tracking method
- 2af60bf Mahnoosh Kholghi - added a new method for peak tracking
- 01054f2 Mahnoosh Kholghi - fixed a bug in the config file
- 789e618 Mahnoosh Kholghi - added a method to draw spectral tracks
- 470283d Mahnoosh Kholghi - extended the drawing method to highlight bands bounadries
- e3577e1 Mahnoosh Kholghi - added methods to draw peak hits on spectrogram
- 36e234e Mahnoosh Kholghi - added energy spectrogram class
- 0c84f1b Mahnoosh Kholghi - cleaning the code
- b5f8125 Mahnoosh Kholghi - added FindLocalSpectralPeaks method and corresponding test
- 85c8f74 Mahnoosh Kholghi - added GetPeakBinsIndex method and corresponding test

- e7a59d9 Mahnoosh Kholghi - added spectral peak tracking for night parrot
- 6f194f2 Anthony Truskinger - Fixes dotsettings
- 71d6f14 mkholghi - cleaning feature learning and extraction process
- 30272e0 mkholghi - removing unnecessary comments and lines
- ec2e73e Anthony Truskinger - Unit test fixes for rendering all gray FCS
- 8043425 Michael Towsey - Reworked code to draw grey scale spectrograms
- 6ec16c6 Michael Towsey - Adds rendering of ALL grayscale LD spectrograms to standard indices generation
- 44ac920 Mahnoosh Kholghi - amended patch sampling approach in semi-supervised clustering process
- 8f9dac6 Mahnoosh Kholghi - amended ListOf2DArrayToOne2DArray method, so that it can merge matrices with different number of rows
- 9981af3 Mahnoosh Kholghi - debugged the semi-supervised feature learning method
- 73b2b5f Mahnoosh Kholghi - added semi-supervised fearture learning
- 0d64531 Mahnoosh Kholghi - fixed bug in frame window length and fixed window overlap
- b52daae Mahnoosh Kholghi - fixed bug in feature extraction
- 895396f Mahnoosh Kholghi - Fixed bug in audio segmentation
- 2eb488b Mahnoosh Kholghi - added audio segmentation and generate features for any desired resolution
- e9be1d4 Mahnoosh Kholghi - added downsampling step to feature extraction and feature learning process
- 7c915e6 Mahnoosh Kholghi - Fixed test methods due to build failed: commit 780a171 by @towsey
- e452e3a Mahnoosh Kholghi - Editted MahnooshSandpit
- e5eec5c Mahnoosh Kholghi - saved similarity vectors to a csv file
- ed2a488 Mahnoosh Kholghi - added random sampling without replacement
- c951ea4 Mahnoosh Kholghi - added ExtractClusteringFeatures and GenerateSpectrograms to MahnooshSandpit
- cf0c3b9 Mahnoosh Kholghi - added frame window length and step size
- ddeba37 Mahnoosh Kholghi - updated project files
- a2154b5 Mahnoosh Kholghi - added two parameters for making a window of group of frames
- 203e07e Mahnoosh Kholghi - more recording samples tested for different spectrogram classes
- bfe2369 Mahnoosh Kholghi - revised drawing method
- 7c27469 Mahnoosh Kholghi - revised noise reduction method
- 554dbd2 Mahnoosh Kholghi - added a method to calculate percentile noise profile
- 4527865 Mahnoosh Kholghi - adding a class for feature learning settings and configurations
- d27c09c Mahnoosh Kholghi - revised sandpit based on new classes
- ad0e209 Mahnoosh Kholghi - adding two parameters to config file
- f05446d Mahnoosh Kholghi - Adding two class for feature learning and feature extrcation processes
- 21ba023 Mahnoosh Kholghi - changed settings
- 9f1831e Michael Towsey - More work on Spectrogram clsases
- 98a948c Mahnoosh Kholghi - Fixed bug
- 719bf88 Michael Towsey - work on new standard spectrogram classes
- f7168dc Mahnoosh Kholghi - Added test for different spectrograms
- a79bbdc Mahnoosh Kholghi - cleaning the code and notes
- 9bf06be Mahnoosh Kholghi - Added a constructor to EnergySpectrogram
- 918b8b3 Mahnoosh Kholghi - Added get image methods
- 9cf361c Mahnoosh Kholghi - Added Amplitude and Decibel Spectrograms
- da2ea55 Mahnoosh Kholghi - changes to config file
- 780a171 Michael Towsey - Changes to EnergySpectogram class to remove dependence on Base class.
- 5bb02f2 Mahnoosh Kholghi - added ignore attribute to PSD test
- df76bf7 Mahnoosh Kholghi - specified path to feature learning config file
- 9ed6cae Mahnoosh Kholghi - modified PSD test
- 69a83c1 Mahnoosh Kholghi - modified config file
- 078ce16 Mahnoosh Kholghi - Added energy spectrogram
- c0e47a5 Mahnoosh Kholghi - added a test for PSD class
- 6e566bc Mahnoosh Kholghi - Added class to calculate power spectrum
- a2a8af3 Mahnoosh Kholghi - added class for PSD

- c70b164 Mahnoosh Kholghi - Read parameters from config file
- 433ff34 Mahnoosh Kholghi - added config file for feature learning
- 4757dc9 Mahnoosh Kholghi - fixed bug in feature pooling step
- db4d8d7 Mahnoosh Kholghi - added min pooling
- d6ad371 Mahnoosh Kholghi - fixed a bug in writing the features to file
- a58fa6e Mahnoosh Kholghi - added a method to get the min value of a vactor
- 1c05ae0 Mahnoosh Kholghi - Added a condition for vector normalization
- 991bdd1 Mahnoosh Kholghi - updated config for Acoustic Test packages
- f2b7649 Mahnoosh Kholghi - Updated MSTest
- 28f4be7 Mahnoosh Kholghi - Added skewness measure
- ba7d158 Mahnoosh Kholghi - updated McMaster
- 6cbe20e Mahnoosh Kholghi - update MSTest
- 8b19583 Mahnoosh Kholghi - writing all feature vectors to one file
- 15888da Mahnoosh Kholghi - cleaning the code
- 512f8b4 Mahnoosh Kholghi - generating features for a set of recordings
- 9b7a178 Mahnoosh Kholghi - fixing the error in directory
- 3fdb19c Mahnoosh Kholghi - Adding a method for arbitrary freq bins

Contributing to audio-analysis

Best Practices

- Set the git `autocrlf` config setting. See https://help.github.com/articles/dealing-with-line-endings/#global-settings-for-line-endings for instructions.
- Avoid adding binary content to this repository. If it must be added, ensure git-lfs is tracking it.
- NEVER commit if the code does not build to the `master` branch
- Try to work on branches if your code negatively affects production code
- Write code in American English. Documentation may be written in Australian English.
- Wherever possible **use un-prefixed SI units for variables**
    1. Variables with no unit **MUST** be standard units
        - `var duration = 30` should **always** be 30 seconds
        - `var bandwidth = 50` should **always be hertz**
    2. **Never** use imperial units
    3. **Always** include the full unit in the name if it does not follow our standard
        - avoid this if possible, see first point
        - e.g. `var minKiloHertz = 3.5`
        - e.g. `var limitHours = 6`
    4. **Do not** abbreviate units
    5. It is **recommended** that full units be used in any user facing field name
        - e.g. `EventEndSeconds` in a CSV file
- Dates and durations:
    1. **ONLY** format dates in an ISO8601 format
        - a modified ISO8601 format with all punctuation removed is acceptable for file/folder names. Example format string: `yyyyMMddTHHmmssZ`
    2. **ALWAYS** format dates with UTC offset information if available
    3. **PREFER** formatting dates in the UTC timezone
    4. **AVOID** exposing `TimeSpan`s to user facing fields (use seconds instead)
        - if a `TimeSpan` needs to be formatted
            - in a log: the default formatting is acceptable
            - in a filename: use ISO8601 duration formatting
- New code or bug fixes must be covered by unit tests. Our project has a historically bad relationship with unit testing and we want to try and improve on this.
  codecov 1%

Required Software

The **required** software for developing new code (not running the program) includes:

1. PowerShell Core (version 6+)

You can install it from here: https://github.com/powershell/powershell#get-powershell

2. A .NET Core SDK

- We aim to use the latest stable version
- You can verify the version our project is using by looking in the global.json file

Alternately, you can download a SDK from here: https://dotnet.microsoft.com/download/dotnet-core/. Note: you want the *Build* *apps - SDK* download.

If you receive the following error message, you need to update your SDK:

> Unable to locate the .NET Core SDK. Check that it is installed and that the version specified in global.json (if any) matches the installed version.

3. An IDE (code editor)

There are three options:

1. **Visual Studio 2019** (Windows only)
   - Install features:
     - C# Development
     - .NET Core SDK for Visual Studio 2019
   - If you're at a university that has an Office365 Subscription you can download software from https://azureforeducation.microsoft.com/devtools
   - The community edition of Visual Studio should work fine and is totally free
   - [Optional] Resharper Ulitmate (Academic License)
   - Install these plugins (*ReSharper* menu > *Extension Manager*)
     - ReSpeller Free
2. **VS Code** (recommended for Mac and Linux, works on Windows too)
   - Install from here: https://code.visualstudio.com/
   - Open the `ap.code-workspace`
   - Install the recommended workspace extensions
3. **JetBrains Rider** (Windows only)

4. Git

A recent version of the `git` executable must be on your `PATH` (the standard install should do this).

https://git-scm.com/downloads

5. Git LFS

https://git-lfs.github.com/

Binary Large Objects (BLOBs)

We use git-lfs to store BLOBs for testing audio file converters. If you want to run the unit tests you need to have git-lfs installed.

Not all BLOBs are stored in git-lfs. See the `.gitattributes` file to list what files are included.

You can check the status of LFS files with the command `git lfs status`.

AP001

If you cloned the repository before LFS was installed you will need to:

1. Install Git LFS
2. `cd` to the audio-analysis directory
3. Run `git lfs install` to set up Git LFS
4. Run `git lfs pull` to download the LFS BLOBs
5. Use `git lfs ls-files` to verify the files have been restored.

> An asterisk (`*`) after the OID indicates a full object, a minus (`-`) indicates an LFS pointer.

Third party contributions

Third party contributions should be made by:

- forking the repository
- making changes in a branch
- submitting a pull-request to merge those changes from your-fork and branch, to our copy and master branch

Help wanted

We mark the most straightforward issues as "up for grabs". This set of issues is the place to start if you are interested in contributing but new to the codebase.

- QutEcoacoustics/audio-analysis - "up for grabs"

Contribution "Bar"

Project maintainers will merge changes that improve the product significantly and broadly and that align with our roadmap.

Contributions must also satisfy the other published guidelines defined in this document.

We will gladly accept any documentation or script enhancements.

DOs and DON'Ts

Please do:

- **DO** follow our style (enforced by StyleCop)
- **DO** give priority to the current style of the project or file you're changing even if it diverges from the general guidelines.
- **DO** include tests when adding new features. When fixing bugs, start with adding a test that highlights how the current behavior is broken.
- **DO** keep the discussions focused. When a new or related topic comes up it's often better to create new issue than to side track the discussion.
- **DO** blog and tweet (or whatever) about your contributions, frequently!

Please do not:

- **DON'T** make PRs for style changes.
- **DON'T** surprise us with big pull requests. Instead, file an issue and start a discussion so we can agree on a direction before you invest a large amount of time.
- **DON'T** commit code that you didn't write. If you find code that you think is a good fit, file an issue and start a discussion before proceeding.
- **DON'T** submit PRs that alter licensing related files.

Commit Messages

Please format commit messages as follows (based on A Note About Git Commit Messages):

```
Summarize change in 50 characters or less

Provide more detail after the first line. Leave one blank line below the
summary and wrap all lines at 72 characters or less.

If the change fixes an issue, leave another blank line after the final
paragraph and indicate which issue is fixed in the specific format
below.

Fixes #42
```

Also do your best to factor commits appropriately, not too large with unrelated things in the same commit, and not too small with the same small change applied N times in N different commits.

File Headers

StyleCop automatically suggest an appropriate file header. Please use it at the top of all new files.

Copying Files from Other Projects

We sometimes use files from other projects, typically where a binary distribution does not exist or would be inconvenient.

The following rules must be followed for PRs that include files from another project:

- The license of the file is permissive.
- The license of the file is left in-tact.

Porting Files from Other Projects

There are many good algorithms implemented in other languages that would benefit our project. The rules for porting an R file to C#, for example, are the same as would be used for copying the same file, as described above.

The Help command

The easiest and most up to date way of getting help for a command is to use the *help* command or the `--help` option.

For example:

```
AnalysisPrograms.exe --help
```

To see all the available commands, run:

```
AnalysisPrograms.exe --list
```

To get help for a specific command (in this case `audio2csv`), run:

```
AnalysisPrograms.exe audio2csv --help
```

For an in depth guide see Running commands.

Analyze Long Recordings (audio2csv)

- **Command**: audio2csv
- **Config file**: <any compatible config file>

This command applies an analysis to a long recording. It works well with recordings that are up to 1440 minutes (i.e. 24-hours) long.

This command takes a single long-duration audio recording as input, splits it into segments and calculates results and writes the output to several .csv files. This analysis process supports **parallel** processing.

Analyze long recordings can run many types of analyses. The provided config file determines what kind of analysis is run.

To see a list of all analyses that can be used by *audio2csv*, execute:

```
AnalysisPrograms.exe analysesavailable
```

As the command name implies, the output of this analysis is a set of CSV files that can contain:

- recognized acoustic events
- summary indices
- spectral indices

Usage

This section describes the command line arguments required to, for example, calculate acoustic indices derived from a single audio file.

To run the command, type:

```
$ AnalysisPrograms.exe audio2csv [arguments] [options]
```

Here is an example of a command line with abbreviated path names:

```
$ AnalysisPrograms.exe audio2csv "audioPath\fileName.wav" "configPath\fileName.yml"
"outputPath\directoryName"
```

The above three paths are required arguments in that order. The program will fail if they are not found on the command line.

> Reminder: All path names should be double quoted, BUT make sure you use plain double quotes (`".."`) and NOT so called *smart quotes* (`“..”`).
>
> Reminder: The paths to directories should *never* end in a forward-slash or back-slash.

Options

Typically, analyses will also include optional arguments that take default values if they are not found on the command line.

Here is more detail about the command line options:

- `Source`: The path to the audio file to process
- `Output`: The output will be placed the directory provided
- `Config`:
  - **IMPORTANT**: The name of the config file chosen will determine the kind of analysis that is run.
  - The config file contains parameters that are unique for the chosen analysis. If the config file cannot be found, *AP.exe* will try and use a default.

    - Typically, if a required parameter is not found in the config file, a default value will be used or in certain cases, a fatal error may result.

- `--channels`: The default value is nothing (not specified) which means that all channels are used. *AP.exe* can process files with up to 8 channels.
    - [TODO: CHECK THIS] Channels are numbered, 0, 1, 2, ... In the usual case of stereo, left channel = 0 and right channel = 1. Note that, although the software should be able to accept recordings where channel number is greater than two, this use has not been debugged and results are undefined.

- `--mix-down-to-mono`: The default value is `true`. Typically, indices are calculated on the mixed down waveform.
- `--parallel`: If you have access to a multi-core CPU you can set this option to true. Otherwise, the segments will be cut and analysed in sequence.

Use the analysis-identifier option (`-a` or `--analysis-identifier`) followed by the `<analysis type>` to choose the analysis to run. If you do this *AP* will not have to guess the name of your config file and this your config file can be named anyway you like.

Draw Long Duration Spectrograms

- **Command**: DrawLongDurationSpectrograms
- **Config file**: SpectrogramFalseColourConfig.yml
- **Config file2**: IndexPropertiesConfig.yml

This command produces a single false-colour (FC) spectrogram, taking as input the spectral indices produced by the *Acoustic Indices* analysis on a single audio file.

One FC spectrogram is produced per audio file. To learn more about long-duration, false-colour spectrograms, check out our tutorial at https://research.ecosounds.org/research/eadm-towsey/long-duration-audio-recordings-of-the-environment.

Usage

This section describes the command line arguments required to draw a false-colour spectrogram derived from matrices of spectral acoustic indices extracted from single long-duration audio recording.

To run the command, type:

```
$ AnalysisPrograms.exe DrawLongDurationSpectrograms [options];
```

The output of this command is a set of false-colour and grey-scale long-duration index spectrograms.

Here is an example of a command line with abbreviated path names:

```
$ AnalysisPrograms.exe colourSpectrogram –i "path\directoryName1" –o "path\directoryName2" -fcs
"path\fileName3.yml" -ip "path\fileName4.yml"
```

The above four paths are obligatory arguments (the program will return a fatal error if they are not found on the command line).

Options

- -i|--input-data-directory <INPUT_DATA_DIRECTORY> Directory where the input data is located.
- -o|--output-directory <OUTPUT_DIRECTORY> Directory where the output is to go.
- -ip|--index-properties-config <INDEX_PROPERTIES_CONFIG> User specified file containing a list of indices and their properties.
- -fcs|--false-colour-spectrogram-config <FALSE_COLOUR_SPECTROGRAM_CONFIG> Config file specifying directory containing indices.csv files and other parameters.

The `--index-properties-config` config file contains a list of indices and their display properties, including information about the brightness and contrast of the RGB channels in the FC spectrograms. These values are extremely important in determining the amount of detail that can be seen in FC spectrograms.

Config file parameters (SpectrogramFalseColourConfig.yml)

The colourSpectrogram config file gives you access to various parameters that control drawing of long-duration false-colour spectrograms. Here are the available parameters with default values.

Here is some additional information about the more important parameters:

- `ColorMap1`: "ACI-ENT-EVN". You can experiment with various combinations of indices but we find these to work well. See the tutorial at https://eprints.qut.edu.au/110634 for more information about the indices to which abbreviations refer.
- `ColorMap2`: "BGN-PMN-R3D"
- `ColourFilter`: This parameter determines the extent to which low index values are emphasized or de-emphasized in their colour representation. Its purpose is either to give emphasis to low intensity features or to de-emphasise them. This

parameter applies a function that lies between y=x^-2 and y=x^2, i.e. between the square-root and the square.
  - When filterCoeff = 1.0, small values are maximally emphasized, i.e. y=sqrt(x).
  - When filterCoeff = 0.0, the matrix remains unchanged, that is, y=x.
  - When filterCoeff =-1.0, small values are maximally de-emphasized, i.e. y=x^2.
  - Generally, usage suggests that a value of -0.25 is suitable. i.e. a slight de-emphasis.

- `FreqScale`: "Linear". This sets the type of y-axis or Hertz scale.
  - Eventual options will be:
    - Linear
    - Mel
    - Linear62Octaves31Nyquist11025
    - Linear125Octaves30Nyquist11025
    - Octaves24Nyquist32000
    - Linear125Octaves28Nyquist32000
  - [TODO: Update] Only "Linear", "Mel", and "Linear125Octaves7Tones28Nyquist32000" are available at present.

- `YAxisTicInterval`: 1000. Horizontal grid lines will be placed every 1kHz interval. This can be set to 5000, if the recording sample rate is 96 kHz, or to 2000 if the recording sample rate is 44.1 kHz.

Concatenate Index Files

- **Command**: ConcatenateIndexFiles
- **Config file**: SpectrogramFalseColourConfig.yml
- **Config file2**: IndexPropertiesConfig.yml

This command also produces false-colour spectrograms. However instead of taking the spectral indices from a single audio file, it takes the total output from multiple runs of the *audio2csv* command, and concatenates them to produce one or more concatenated false-colour index spectrograms.

Typically, this command is used to produce a sequence of one or more 24-hour false-colour spectrograms, where the original recordings can be anything from 30 minutes to 24 hours duration. 24-hour false-colour spectrograms are much easier to interpret because sound-marks, such as the morning chorus, evening chorus, and insect tracks, are easier to identify. False-colour index spectrograms shorter than about 3 hours are difficult to interpret due to the lack of soundscape context.

There is also an option with this command to concatenate the false-colour spectrograms of every audio recording that can be found in a specified directory into one large data-set and image. Due to memory constraints however, one would not usually attempt to concatenate more than about 48 hours of recordings.

It is **strongly** recommended you only run this command on files from a single acoustic sensor deployment - don't mix recordings from different sites or deployments!

Usage

This section describes the command line arguments required to concatenate the output from multiple runs of the audio2csv command on shorter duration audio recordings, that is less than 24-hours. Typically, the output is a sequence of one or more 24-hour FC spectrograms.

To run the command, type:

```
$ AnalysisPrograms.exe concatenateIndexfiles [options]
```

Options

Some of the option are obligatory (the program will return fatal error if they are not found on the command line) and some are optional.

- `-inputdatadirectories`: An array of one or more directories where the original csv files are located. The required files can be in subdirectories to any depth
- `-inputdatadirectory`: A single directory where *all* the original csv files are located. This option exists (in addition to the above) as a hack to get around commas in paths conflicting with PowerArgs' array parsing feature.
- `-outputdirectory`: The directory where the all the output is to go.
  - If it does not exist it will be created. If it exists contents will be overwritten.

- `-directoryfilter`: Used as a pattern matcher to collect the required data CSV files, which are assumed to be in a matching directory or subdirectory. We often place the output in directories with same name as the recording file (including the extension).
  - Typically the recording siteName is used as the filter pattern to select directories. It can also be used for naming the output files

- `-fileStemName`: User defined file stem name for the output files.
- `-startdate`: A date at which concatenation is to begin. If null, then start with earliest available file. Can parse an ISO8601 date.
- `-enddate`: A date at which concatenation ends. If null, then will be set equal to today's date available file. Can parse an

ISO8601 date.

- `-timeSpanOffsetHint`: A TimeSpan offset hint required if file names do not contain time zone offset info. NO DEFAULT IS SET.
- `-indexpropertiesconfig`: User specified file as for the colourSpectrogram command.
- `-falsecolourspectrogramconfig`: Config file for drawing the false colour spectrograms.
- `-concatenateeverythingyoucanlayyourhandson`: Set true only when concatenating more than 24-hours of data into one image

Other notes

- StartDate: A .Net DateTimeOffset object, start dateTime must be in format readable into a DateTimeOffset object, e.g. 2015-10-25T00:00:00+10:00

- EndDate: A .Net DateTimeOffset object, end dateTime must be in format readable into a DateTimeOffset object 2015-10-25T00:00:00+10:00

- TimeSpanOffsetHint (timespan): should be set to +1000 hours for Queensland. = TimeSpan.FromHours(10), e.g. 10:00:00