Biomechanics Analysis and Reporting Application
User Manual

Contributors:

Ben Senderling, Boston University

Version 1.2

Contents

Intent

The creation of this application hopes to establish a generalized framework by which scientists in the biomechanics community conduct their data processing and analysis. The intent is that this will facilitate data analysis and reporting by making the process easier for research staff and students with varied backgrounds, and by enhancing the repeatability of analytical procedures. The framework provides a base-level application that others can use as-is or build upon with their own sources and methods.

Scope

The base-level of the application includes 1) loading of data, 2) processing and analysis, 3) graphical review and 4) export of the data. Its design was meant to be dynamic and open to varied methods used in the study of human movement. Ultimately, these methods are not all that dissimilar to methods in other fields. It is likely those outside the realm of biomechanics would also find this application useful.

Some aspects of data processing are outside the scope of the application. These may require dedicated hardware or complex software solutions, involve high performance computing or complex modeling and simulation.

Overview

The app has a main application called BAR_App. This will handle all the configuration settings and dynamic use of modules. Modules are considered any mlapp- or m-file that is dynamically used by BAR_App. Some modules are considered central to the application. These include 1) Groupings, 2) Merge and 3) General Review. Modules fit into categories named 1) Load, 2) Process, 3) Analysis and 4) Review. There are also Mini-Modules which can be dynamically called within these Modules. These module names are also important folders used by the app. Another folder called Subroutines is meant to house low level functions that may have widespread use across multiple modules. There are examples of some of these methods included with the source code. These will have 'Example' in the file name.

Load modules are mostly functions that read data from a data file, package it into a BAR App data structure and load it into the app. They are organized using the file extension and equipment type. The file extension must match the file extension of the target file however the equipment type can vary. An example could be load_h5_APDM1Lumbar.m and load_h5_APDM1Raw.m. Both read data from the same h5 file but the first reads only the lumbar data while the later reads only the raw data.

Process modules are more general-purpose analysis methods. These may include segmentation or treatment like filtering. They are not considered the end of a processing step and would produce data that is later used in an Analysis module. Process modules are more general in that they may proceed multiple Analysis Modules, whereas the later is considered an end point.

Analysis modules are similar in function to Process modules but differ in scope and intent. Analysis modules are for complex analyses that are more likely to be other mlapp-files. These modules allow users to perform complex operations with significant user input. They often will take time series or aggregate data and produce single metrics or statistics.

Review modules are meant for data visualization. These modules should be mostly mlapp-files that can dynamically produce general or specific figures of the data. For most users the review_General.mlapp

module is hoped to be sufficient. For most complex analyses there will be a corresponding Figure Mini-Module that produces specific figures describing the analysis results.

Community Guidelines

Individuals or groups seeking to contribute to this work are welcome to do so. Contributions can be made directly to the source code or by working within the framework of the app. Projects can make use of the app and its framework but publish it separately in another repository. This cooperative, but not strictly collaborative, behavior would help promote reproducibility.

If others wish to contribute to the design and development of the app they can visit the Projects page on the GitHub repository. This includes projects to organize efforts on the app's flexibility, adaptability and expansion. Its flexibility is due to the dynamic function calling and database structure. This seems to be developed well so far but might have minor changes as it is used for additional projects. Many potential changes to its adaptability would be made in the modules. These would involve greater user choice in what actions are performed. Most future work is likely to fit into expandability. These would involve new modules and new capabilities. All these efforts are managed as Projects on the GitHub repository. Below are some examples of these. Visit the repository to contribute or add new efforts to these ideas.

Flexibility Improvements

The General Review Module makes use of cells to plot data in multiple dimensions. Of interest is also adding the ability to plot data according to group information. The organization of the data to plot could be changed to fit both of these efforts better.

Adaptability Improvements

Various methods, especially nonlinear methods, should be run with varied parameters to ensure results are consistent. The interface to some modules could be adapted to make this easier.

Expansion of the app

Work at the BU MoveLab features a significant amount of quality control. Currently, this is not a feature of the app. This could potentially be implemented as a module that is run alongside other modules to flag poor quality data. It would likely make use of an additional quality ('qua') data type. It would be important for this to involve automated review methods so visual methods are relied upon less. These methods also need to link results, raw and processed data, and original data files.

Future versions of the app are hoped to include some level of statistical processing. Currently the export and Group Module is developed with statistics in mind but there is no capability within the app of performing them. At first it would include calculations for group descriptives and distribution information. Later it could include T-Tests, ANOVAs or more complex modeling.

Installation

Instructions

The BAR App is provided on GitHub as source code and as a MATLAB App installation file. This allows it to be installed or used in a number of ways. 1) The repository can be cloned to a local directory and used directly. With GitHub Desktop this also allows an easier way for users to contribute to the project. The configuration of the app allows multiple directories to be used for source code. This allows users to keep contributions to the open-source project separate from confidential projects or research efforts. 2) The compressed source code can be downloaded from the online repository. Decompress the files to use them. 3) Releases include a mlappinstall-file that can be used to install the application in MATLAB. After installing it will appear in your MATLAB APPS toolbar, pictured below. 4) The BAR_App.mlapp file can be run the same as in #3 outside of MATLAB. So long as the required version of MATLAB is installed and licensed the mlapp-file can be double clicked to start the app.

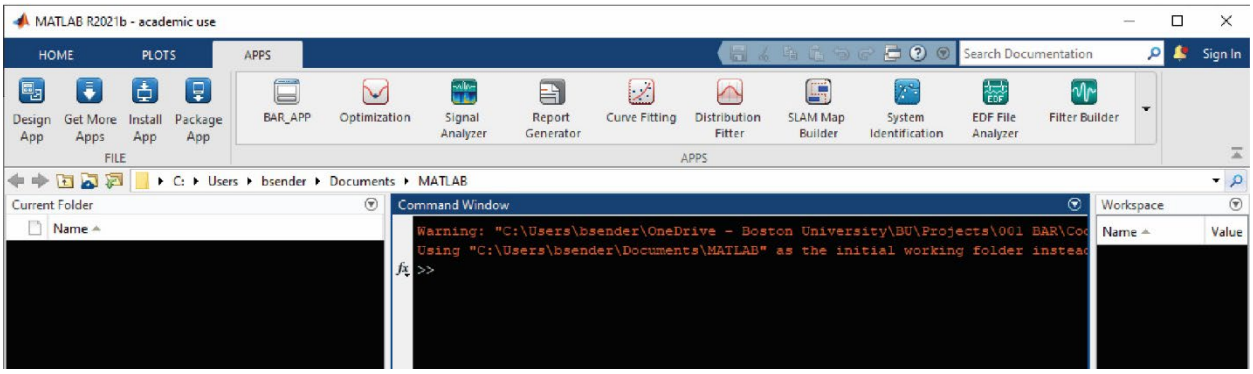


Figure 1

After running the mlappinstall-file the BAR App will show up in the APPS toolbar in MATLAB. The dropdown to the right of the toolbar may need to be selected to locate it. In this view the app has been 'starred' so it shows first.

Recommendations for use

Installing the app as a repository or using the source code directly allows user to edit and use the code as they would any other code. Using the app through the APPS toolbar is best for those who do not need to work on the code but only need to make use of the app's features.

Dependencies

Dependencies for the BAR App includes those below. This does not include dependencies required by other toolboxes used in various modules. If a particular module, such as the Time Lag Module, uses an additional toolbox it will list its dependencies in that section.

- MATLAB 9.11
- Signal Processing Toolbox 8.7
- Statistics and Machine Learning Toolbox 12.2
- System Identification Toolbox 9.15

Quick Guide for First Time Users

This section will help first time users get acquainted with the app. It also takes users through validation steps to ensure the main features of the app are working correctly. Included with the source code are validation files. These are in Validation\txtExample. This folder also includes a createDatasets.m script that can be used to modify or recreate these data files. The instructions below will ask you to perform certain actions using these files. Screenshot are included so you can verify what the app should like once the step is complete. Other sections of this manual may need to be referenced to accomplish these actions.

Step 1 Load the data files

Use the Database Search tab to load the data files for 'Set1'. Your 'Working Database' and 'Search Directory' may not be the same as in Figure 1. These data files have a txt file extension and the Equipment type 'Example'. There is a corresponding load_txt_Example.m function that loads this data into the app.

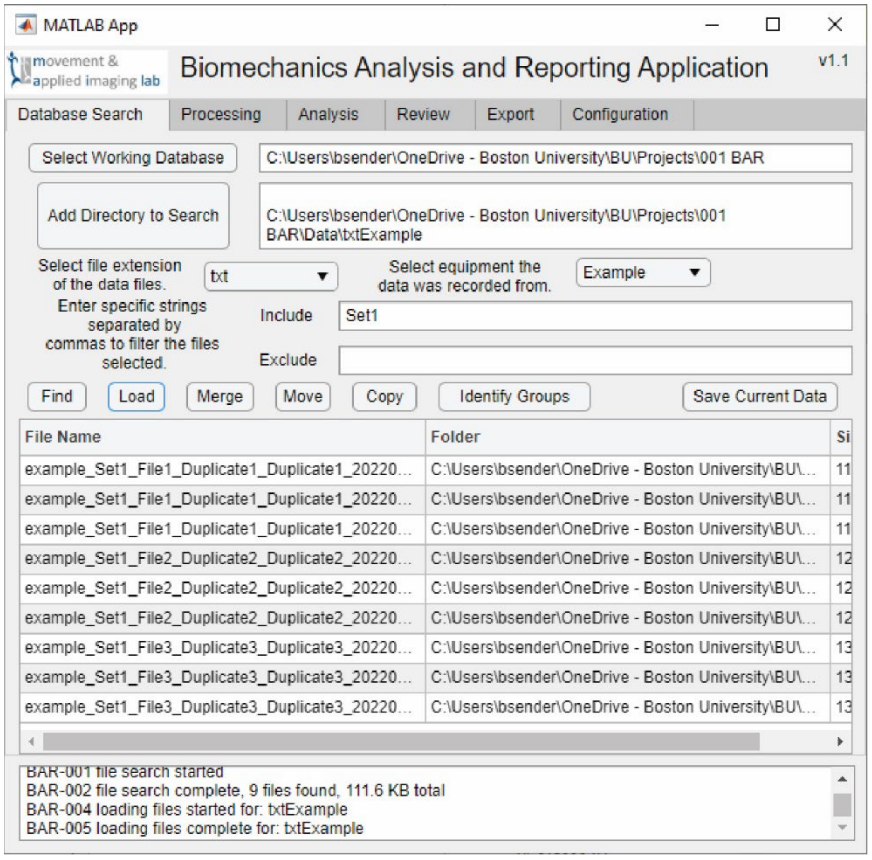


Figure 2

When opening the app it will start on this Database Search tab. This is a picture of the app at the completion of step 1. All of these fields would be blank when opening it for the first time or without a config file.

Step 2 Process group information

Click the 'Identify Groups' button to start the Group Module. Use this to 1) remove duplicates within columns, 2) remove duplicates within rows, 3) remove the remaining 'Duplicate1' column, 4) convert the yyyyMMdd dates to ordinal numbers using the 'File<#>' column, 5) convert the MMddyy dates to ordinal

numbers using the 'File<#>' column. When finished the window should look exactly like Figure 2. Click the 'Save Results and Exit' button to save the data back to the main app.

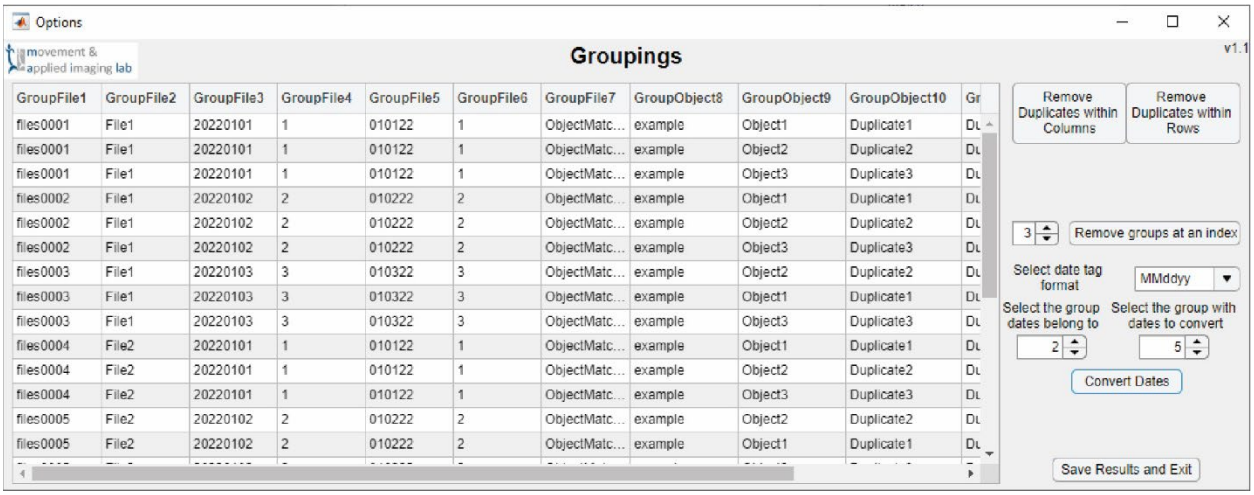


Figure 3

At the end of step 2 the Groupings module should not show any of the file path to the data files or the duplicate file groups. The dates in the yyyyMMdd and Mmddyy format should also have been converted to ordinal numbers that matches their day number.

Step 3 Merge data set 2 into set 1

Once the Group Module has closed look to find the data files for 'Set2' and merge them with 'Set1'. The first data set is already loaded into the app. Data set 2 will need to be found by the app, but not loaded (Figure 3). The Merge Module will load it into the app. Once in the Merge Module 1) combine the data sets file-to-file (Figure 4), and 2) file-to-object. For the second toggle off (Figure 6) and on (Figure 7) the duplicate matches switch. Processing the group information in the previous step is why there are less groups for data set 1 compared to 2. It may seem excessive that the entire path is used for group information, but this has proven the best way to capture information from varied projects. To successfully perform the matching you will need to accurately estimate the matching score. Every tag in data set 1 that matches a tag in data set 2 adds one to the score. Complete any one of these actions and save the data.

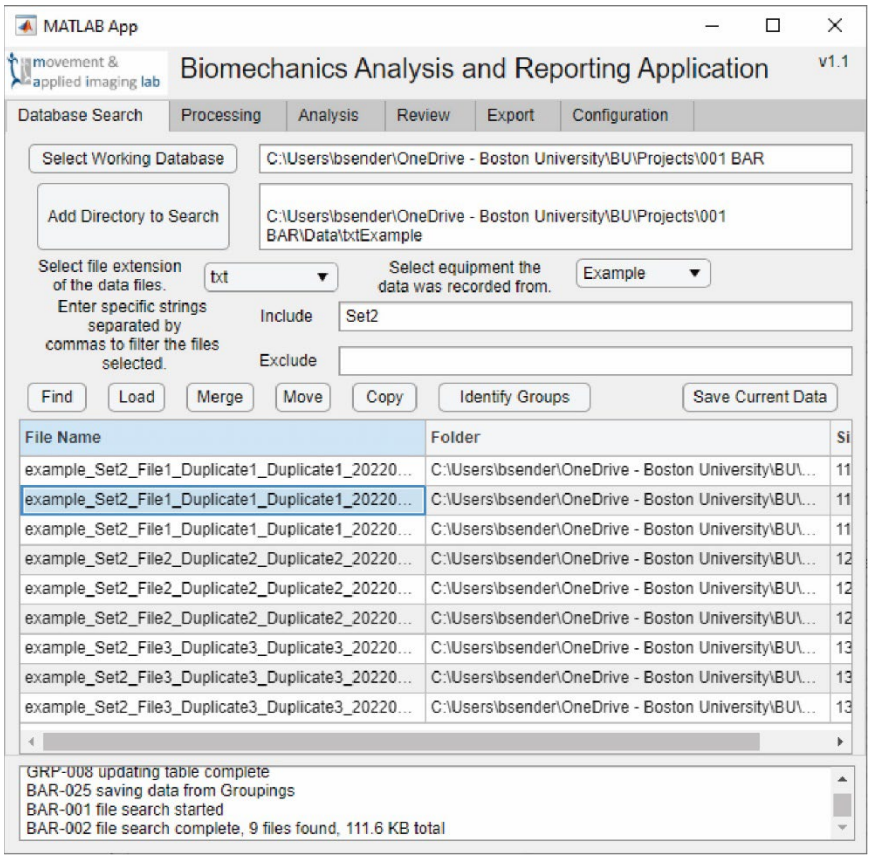


Figure 4

After searching for 'Set2' the app should display all nine of those files. Do not load these files. The Merge Module will do that.

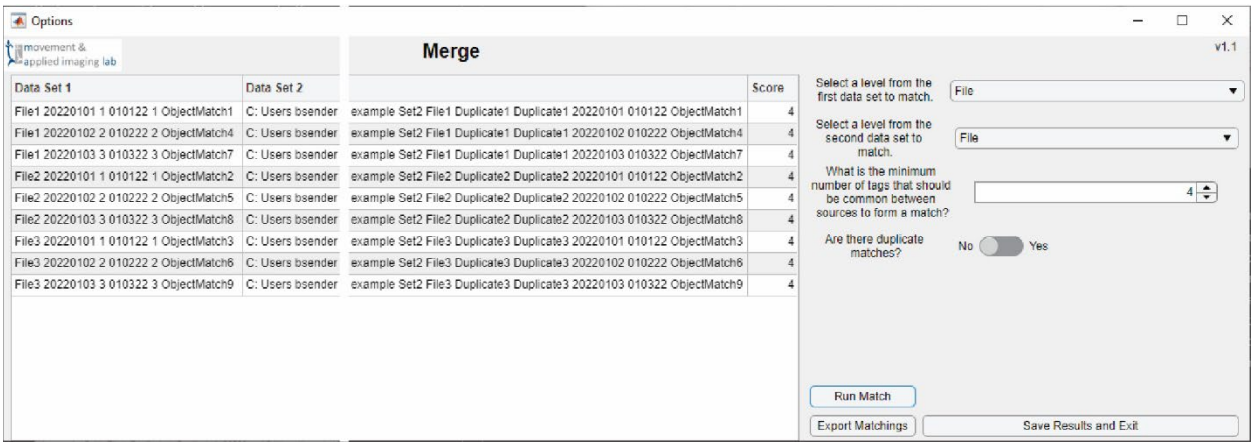


Figure 5

The Merge Module here has matched the File Level of data set 1 with the File Level of data set 2. The score for each match was a 4.

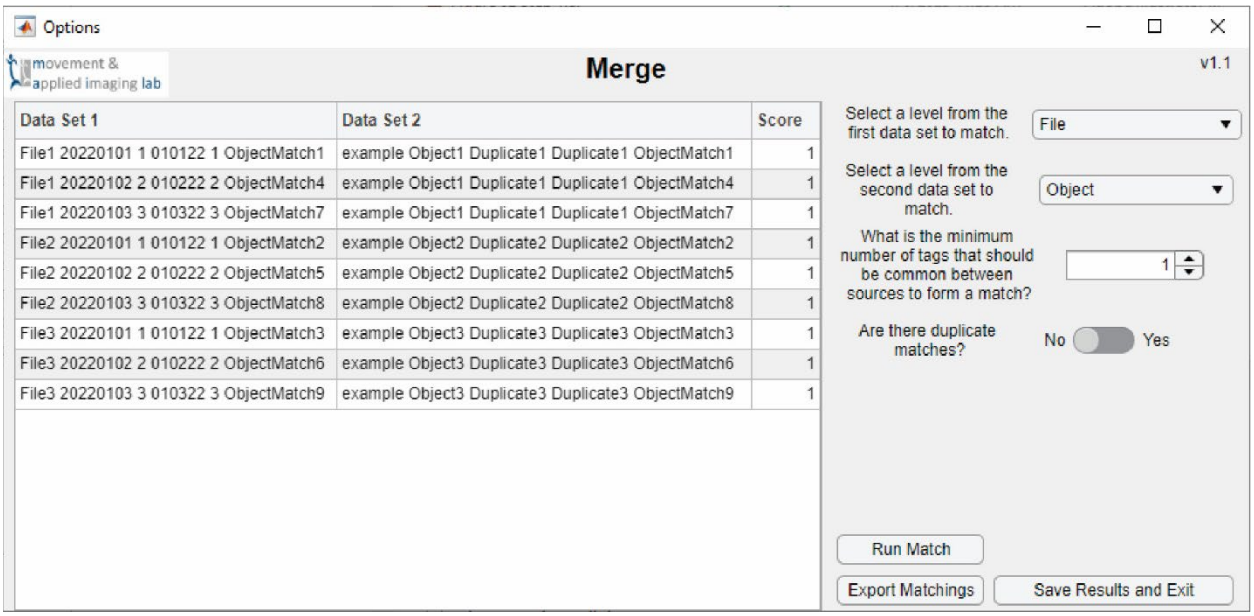


Figure 6

Changing the level selection will reset the merge results. Here the File Level for data set 1 was matched with the Object Level of data set 2 using a score of 1. No duplicates were selected so much of the data is discarded.

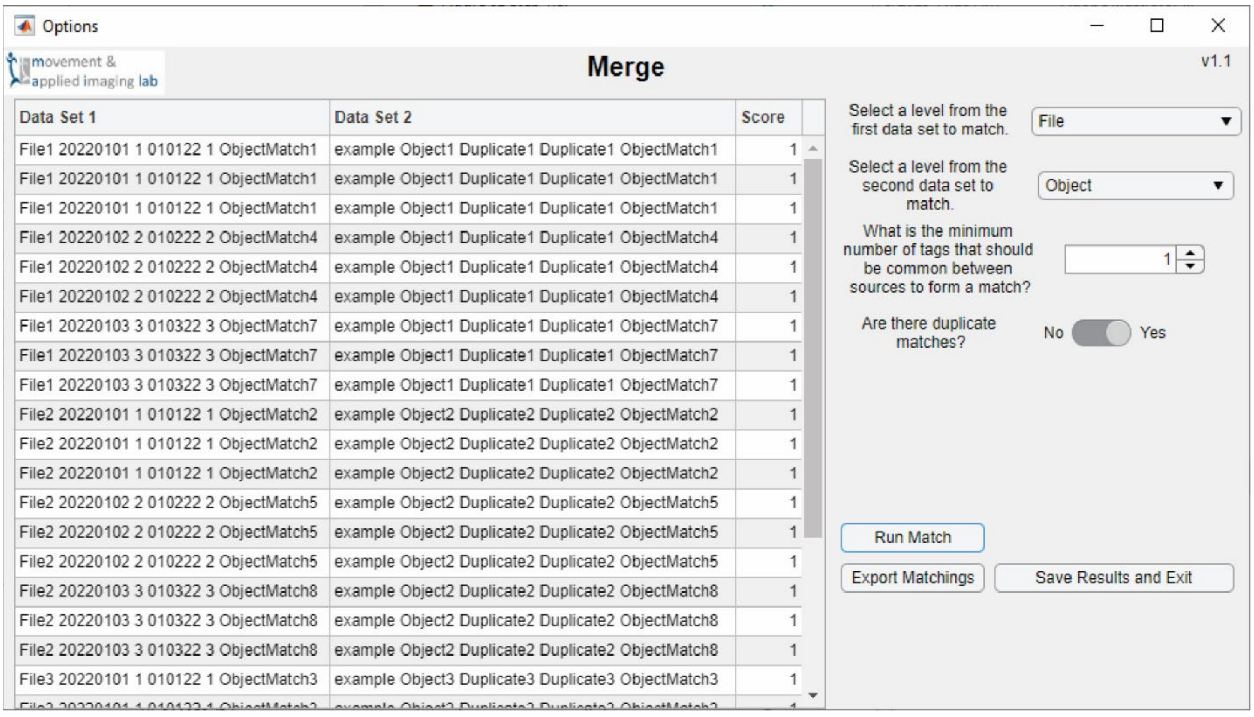


Figure 7

This is similar to Figure 6 except duplicates were allowed. This list is much longer because all of the data was matched.

Step 4 Segment the data

Use the Segment Module to split the data with the Example Mini-Module. This will split the data into thirds. Each third will have a different mean. The new time series will be saved to the 'pro' Data Type with

the number of the segment (1, 2, 3) appended to it. Click on the different levels of the data to see them segmented in the figure (Figure 8). When satisfied click 'Segment all, save and exit' to finish. Segmentation is very specific to individual applications. And it is likely that a custom implementation would look very different. This example code segments all the example data the same, and plots all of them. This contrasts with, as an example, detecting turns from IMU data. This would likely operate only on a select signal or object, and only that item would be displayed.

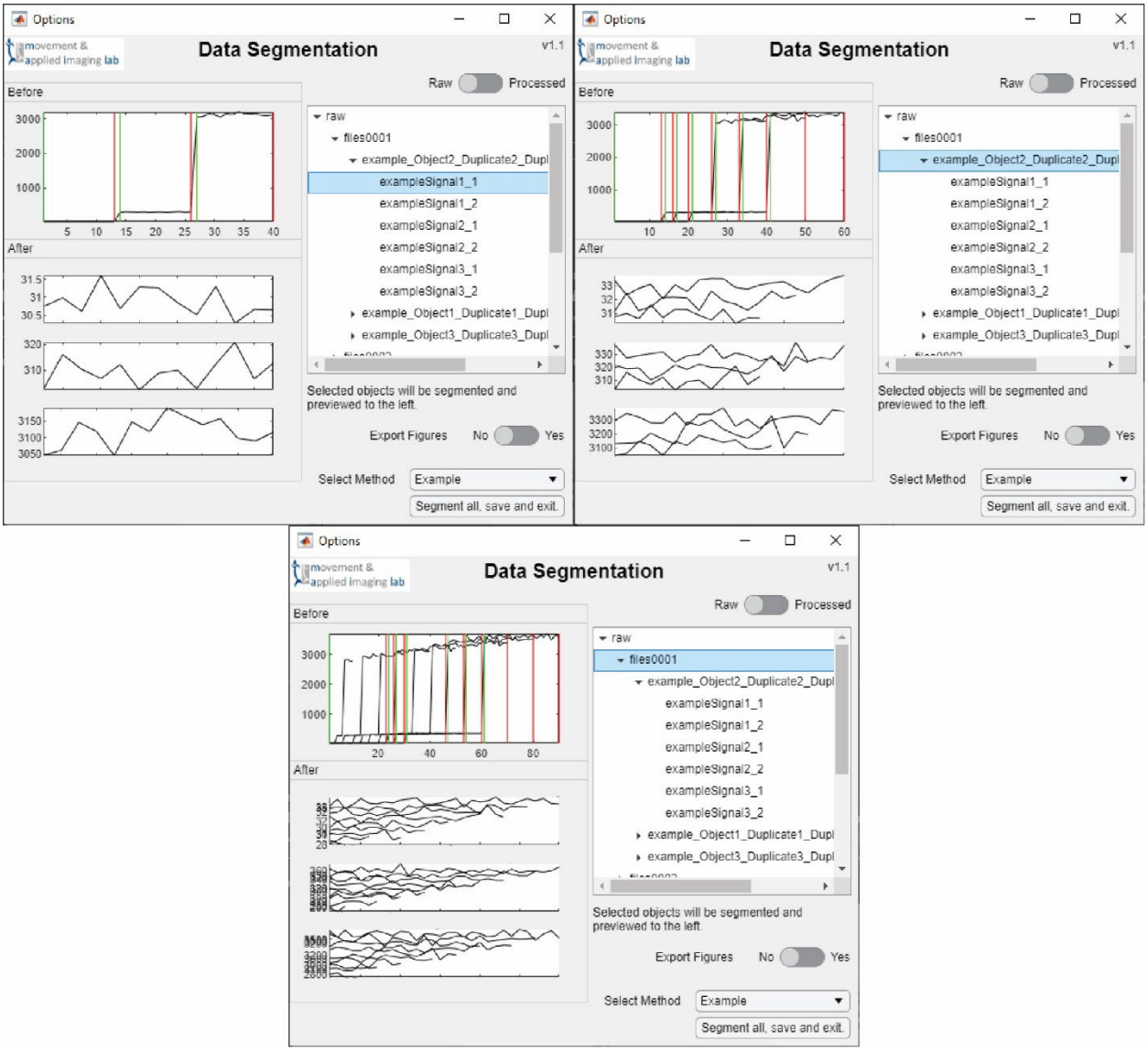


Figure 8

These screen shots show the results of different segmentation selections. In all three the Example Mini-Module split the data into three segments. The top left was performed at the Signal Level, the top right at the Object Level and the bottom at the File Level. Segmentation is very application-specific so custom implementations are likely to be different.

Step 5 Process the data

Next select the Treatment Module from the Processing Tab. Add a Low Pass Filter and a Resample Step to the table (Figure 9). Make sure to experiment adding multiple steps and removing them. Click the 'Process'

button to process all the data. This will create new processed data that can be viewed through the General Review Module (Figure 12). These results will be saved to the 'pro' Data Type with the same name as the raw data.

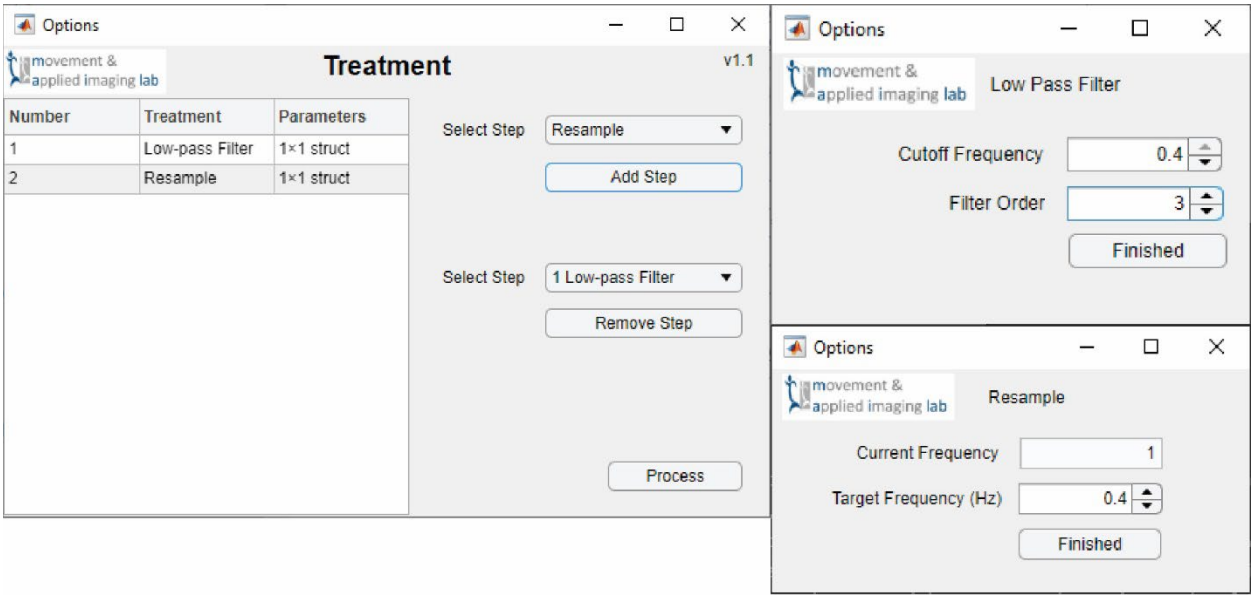
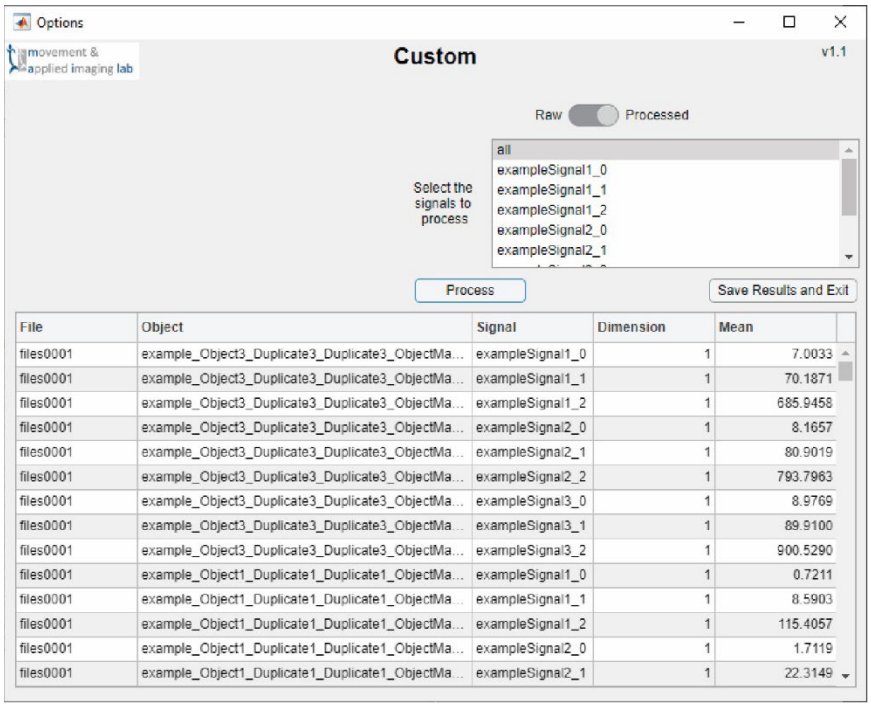


Figure 9

Here the Treatment Module had two steps added to it using the Low-Pass Filter and Resample Mini-Modules. The selections for this example are arbitrary.

Step 6 Analyze the data

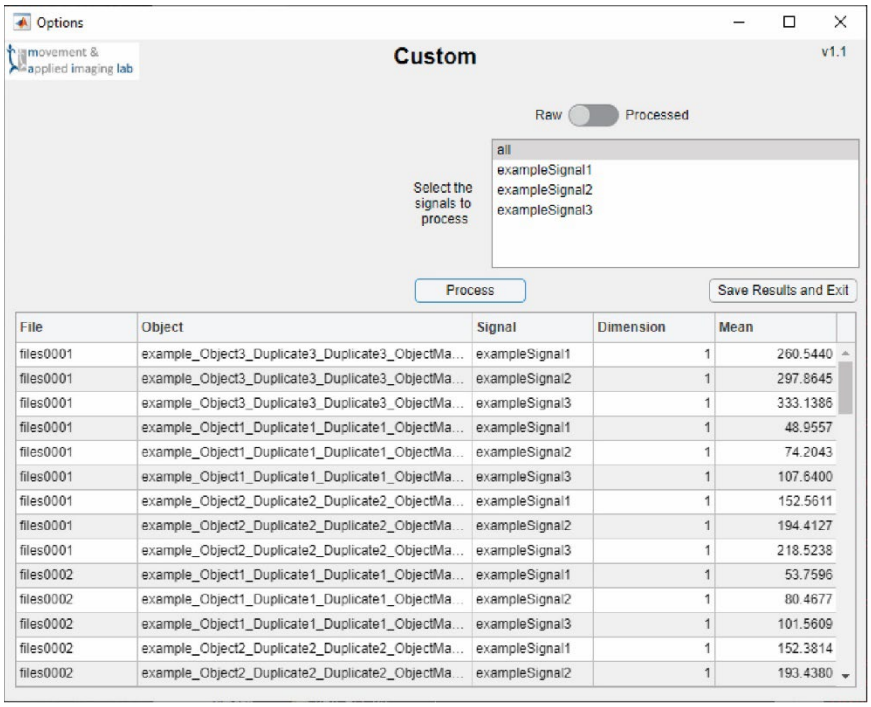
Two examples are provided for this step. One is a script (analysis_Example.m) and the other is an app (analysis_ExampleApp.mlapp). The script runs an analysis, calculating the mean, directly on the raw data. The app provides a user interface that allows users to make choices in the processing. This app is a stripped-down version of other Analysis Modules. It can serve as a starting point for others to develop their own user interface. The image below shows the results of the ExampleApp Module. Figure 11 shows the results of the Example Module.



File	Object	Signal	Dimension	Mean
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal1_0	1	7.0033
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal1_1	1	70.1871
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal1_2	1	685.9458
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal2_0	1	8.1657
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal2_1	1	80.9019
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal2_2	1	793.7963
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal3_0	1	8.9769
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal3_1	1	89.9100
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal3_2	1	900.5290
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal1_0	1	0.7211
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal1_1	1	8.5903
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal1_2	1	115.4057
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal2_0	1	1.7119
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal2_1	1	22.3149

Figure 10

This image of the Example Analysis Module calculated the means of the processed, segmented data. This interface has all of the methods used to write messages to the Log and save results back to the main app. Most modifications would be the addition of switches or dropdown menus to the interface. Those selections would then be used in the Process button callback.



File	Object	Signal	Dimension	Mean
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal1	1	260.5440
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal2	1	297.8645
files0001	example_Object3_Duplicate3_Duplicate3_ObjectMa...	exampleSignal3	1	333.1386
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal1	1	48.9557
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal2	1	74.2043
files0001	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal3	1	107.6400
files0001	example_Object2_Duplicate2_Duplicate2_ObjectMa...	exampleSignal1	1	152.5611
files0001	example_Object2_Duplicate2_Duplicate2_ObjectMa...	exampleSignal2	1	194.4127
files0001	example_Object2_Duplicate2_Duplicate2_ObjectMa...	exampleSignal3	1	218.5238
files0002	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal1	1	53.7596
files0002	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal2	1	80.4677
files0002	example_Object1_Duplicate1_Duplicate1_ObjectMa...	exampleSignal3	1	101.5609
files0002	example_Object2_Duplicate2_Duplicate2_ObjectMa...	exampleSignal1	1	152.3814
files0002	example_Object2_Duplicate2_Duplicate2_ObjectMa...	exampleSignal2	1	193.4380

Figure 11

This is the same as Figure 10 except the raw data was processed. The results will be different as well as the signal names.

Step 7 Reviewing the data

Once that data has been analyzed open the General Review Module through the Review Tab. It will open with the default selections, which is everything. First, do not click anything except the 'Update Plot' button. This will plot all of the raw time series. Next, change the 'Data Type' to 'Processed'. The fields in the dropdowns will update. Select all the Signals named exampleSignal#_# while holding down the crtl key and click 'Update Plot' again. This will display all the filtered and downsampled data. Lastly, change the 'Data Type' to 'Results', change the 'Plot Type' to Histogram, select 'all' in each of the dropdowns, and click 'Update Plot'. Each of these results are shown in Figure 12.

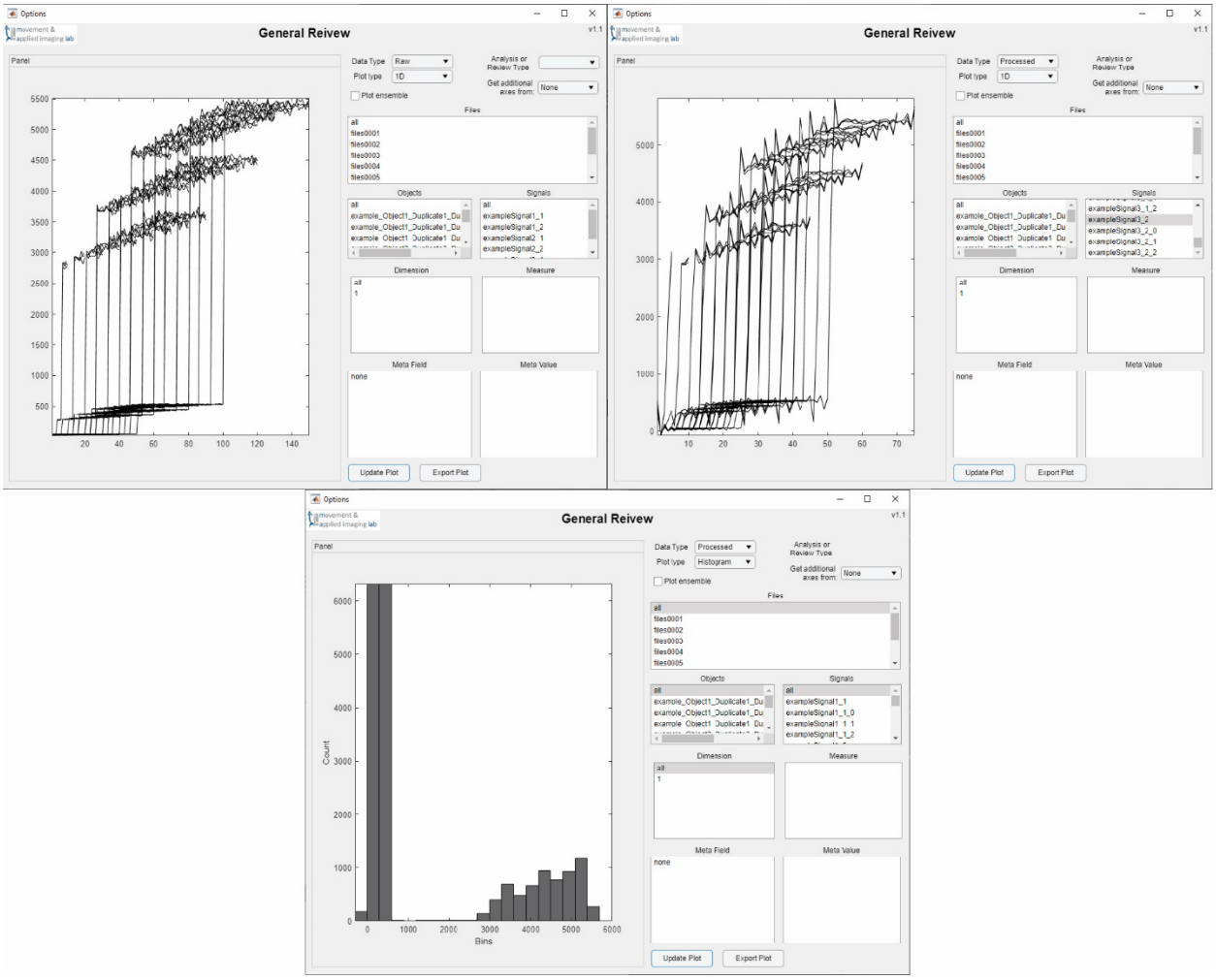


Figure 12

The Review Module should display these results while completing Step 7. In the top left all of the raw and unprocessed data is plotted against their index number. In the top right all the treated data from the Processing Module is plotted. You can see the sharp edge effects from the low-pass filter and the data length is half of the raw data's length. The bottom shows a histogram of all the results created using the Example Module script.

Step 8 Export the data

The last step is to use the Export tab to send the data to an xlsx file outside of the app. For this data we want to export the 'res' data using a 'General' formal. There are two results in this data for the segmentation and the analysis.

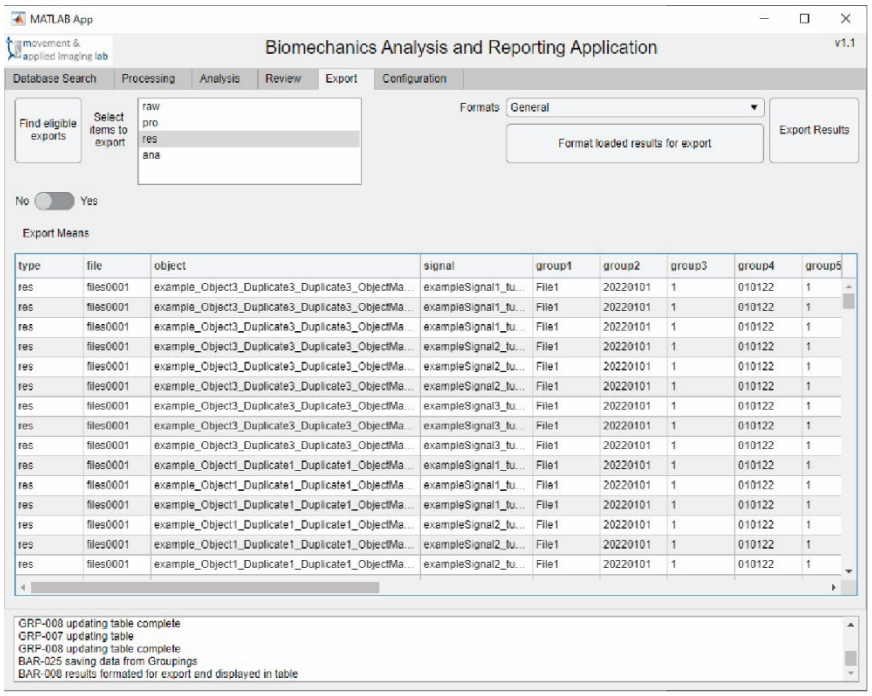


Figure 13

In this image of the Export Tab the results of analyzing the segmented data is shown. The table shows the level of each result and some of the group information, including the dates converted to ordinal number. The numerical results are outside the current window to the right.

Main Application

This section will describe the BAR_App.mlapp file that is central to the BAR App. It controls the 1) locating and 2) loading of data, 3) processing, 4) analysis, 5) review and 6) export. These functions are mostly accomplished by the different tabs within the app.

Database Search

Working Database

Database Search is where all of the file handling will be performed. This starts by setting the Working Database.[1] The button will allow a user to select a folder. The app will then check that certain folders exist within it. These include folders named Export, Figures and Results. Export is used as the default target folder when exporting data from the main application. Figures is used by the General Review and Segmentation modules as a default folder to save figures to. Results is used to save intermediate data and results. Directories can also be typed into the text field.[2] This will not create subfolders but is useful when changing between Working Directories. A Working Database selected through the button[1] will be saved into the config file to be remembered the next time the app is opened.

Before files can be loaded the app must know where to look for them. Directories can be entered into the search field[3], or added through the button[4]. The app performs a recursive search algorithm that looks through all listed folders and subfolders for the target files. This is generally quick but will be significantly slower with network drives.

Add Directory to Search

This field contains the directories used by the app to search for the target data files. Directories added to the text field with the button[4] will be remembered the next time the app is opened. They can also be entered directly into the text field.[3] Multiple directories must be listed on separate lines.

File Extensions and Equipment Types

These are two dropdown lists that specify which files should be searched for. The file extension[5] comes after the period near the end of the file name. On some computers the file extensions may be hidden by default. The equipment type[6] is the system or equipment that produced the file. This indicates how the data file is organized. When the file extension is selected in the app the equipment type options will automatically update.

The available options are identified when the app is started. It is expected that in the same directory as the BAR_App.mlapp file there will be a folder called ‘Load’. This should contain all the loading methods named in the format load_<file extension>_<equipment type>.m. These files need to organize the data into the required BAR App data structure.

Below the two drop downs is a location to include inclusion[7] and exclusion[8] tags. The app will search for these tags within target file names and directories. Words typed into the include text field need to be separated by commas. The app will only identify target files that include all of these tags. Words typed into the exclude text field also need to be separated by commas. Any files including any one of these tags will not be identified by the app. These fields are specifically useful when searching for files from specific experiments or procedures or avoiding files from other projects or analysis efforts.

Find

This button[9] starts the search process to find the target files. The app will identify files with the file extension[5] specified, that include the tags from the include[7] text field and do not include any of the tags in the exclude[8] text field. It is a recursive process where subfolders will also be searched. Searching many subfolders will be slower. It will also be slower searching files located on a network drive.

Load

This button[10] will attempt to load data from the files identified by the app. A progress update will be printed in the log and any errors will also be displayed.

BAR App Data Structure

Here are the details for the BAR App data structure. Data must be organized into this form for a consistent experience with the app. All the various modules of the app rely on this structure. It is mainly constructed using MATLAB's structure data type and dot notation. Load methods will organize data only up to the data.raw.obj level.

- data	This is the parent level of the structure.
- raw	Raw contains raw data. This is data imported directly from a data file.
- file	This is an app-determined name for the file. The true file name is not used mainly because 1) illegal characters may be used and 2) file names can be longer than allowed by MATLAB for variable names. The file names are converted to group information.
- meta	This is used to store meta data from the file. Meta data is often included at the top of the files and includes information about who and what recorded the data and includes other information characterizing the data.
- group	This stores group information about the file. The information is pulled from the file's path by looking for '\' delimiters and is pulled from the file name by looking for '_' delimiters.
- obj	This is called the object level. The idea is this would be named after a sensor. Within that sensor there may be various signals and dimensions.
- data	This contains the signals. It is largely skipped by the app as it is used to store other information about the data.
- sigs	These are the signals that contain the data. They should be numeric arrays with the signals arrayed in columns. Multiple columns represent multiple dimensions of the signal. Multiple dimensions are not required to be located in the same signal. Instead it could be that each dimension is a separate signal. This could be 'FP1_X' or 'FP1_Y' as the signal instead of 'FP1' containing two columns. How this is organized largely depends on how the processing or analysis code is written.
- freq	This stores the sampling frequency of the signals under object. All the signals under an object are assumed to have the same sampling frequency.

- groups	This contains group information specific to the object. The information is pulled from the object names by looking for '_' in the name.
- pro	Processed contains processed data. These have a similar length to raw data but have undergone some processing method. Process methods may also create results but those would be stored in ana or res. Sublevels of pro are expected to be the same as raw.
- ana	Analysis contains processed results that are not true results but may be used for figures or other analysis methods. They are stored separately from res so they are not exported. They are not stored in pro so they are not mistakenly selected for process or analysis methods.
- <Analysis Name>	Under Analysis is the name of the analysis. This could be data.ana.TimeLag, or data.ana.Segment. This is done to separate analysis results so variable names and custom figures can target the results for those analyses without naming clashes.
...	Sublevels for ana.<Analysis Name> are expected to be the same as raw and pro.
- meas	Unlike raw and pro, the signals under ana do not contain the data. Under signals are the measures which are the quantified results of an analysis. Measures contains the data in ana and res.
- res	These are result metrics. They should be short in length, maybe one value, but may be numerous. They will chiefly be exported by the app.
...	Sublevels for res are expected to be the same as ana.

Merge

The merge button[11] runs the Merge Module and allows data from a new search to be combined into an existing dataset loaded into the app. This is useful when data from different files need to be combined. This does require a common, but not strictly similar, naming convention between the two files. More on this will be explained in the Process section of the manual.

Move and Copy

These two methods[12, 13] act on files identified by the app. Their data does not need to be loaded into the app. This is useful when many files across multiple directories need to be combined into a single directory. To enable this blank load methods can be created so the app will make the options available in the Database Search. An example of this is load_agd_Actigraph.m.

Identify Groups

This button[14] runs the Groups Module. It has two purposes. First is to identify new or process existing group information so exports do not contain duplicate or extraneous information. The second is to process group information so the merge process is more efficient. More on how to run the Groups module will be explained in the Process section.

Save Current Data

This button[15] saves the current data within the app to a mat-file. This is most useful when used in between processing steps.

Table

The table[16] in the Database Search tab displays the files that were identified by the app. It includes their file name, path and hard drive size.

Log

This log[17] is where all messages from the BAR App and its modules end up. Messages are coded with a three letter abbreviation for the app they originate from and a 3 digit number identifying the message. Many of these announce if a process is starting or has completed. Most methods also include an update on how many files they have processed. Some processes also include a catch that will print MATLAB errors to the log.

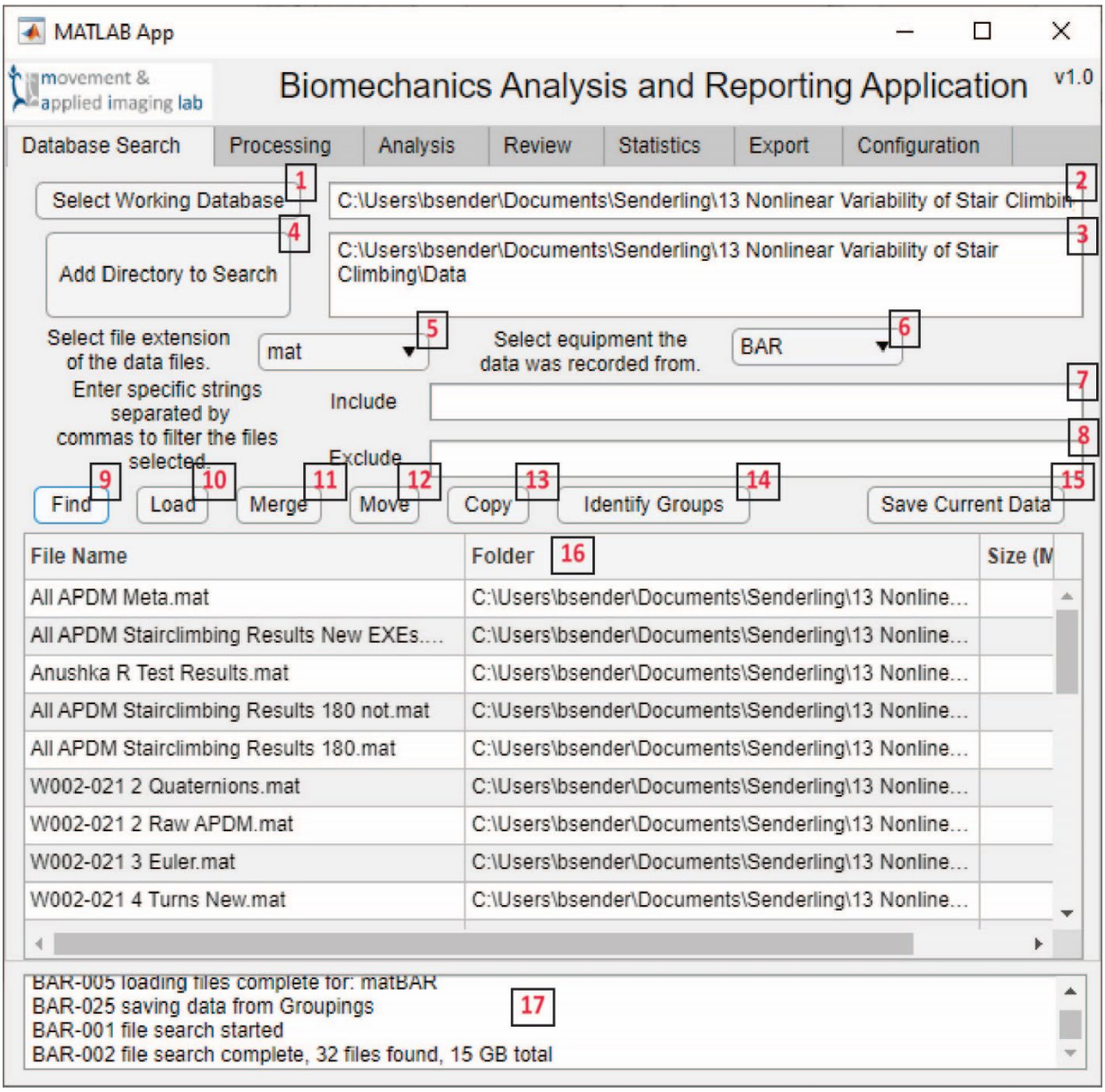


Figure 14 Database Search tab of the BAR App

1. Sets a project folder and creates subdirectories needed for the app.
2. An open text field to change the Working Database.
3. An open text field to change which directories are searched for data files.
4. Adds directories to the search that will be remembered after closing the app.
5. File extensions of the data files to search for.
6. The equipment type and format of the target data files.
7. Tags that must be found in target files.
8. Tags that cannot be in target files.
9. Start the file search.
10. Loads the currently listed files.
11. Merges the currently listed files into an already loaded data set.
12. Moves the listed files.
13. Copies the listed files.
14. Starts the Group Module.
15. Saves the current data.
16. Lists the current files found by the app.
17. The log the BAR App and all modules write to.

Processing

The Processing Tab[1] is used to run Processing Modules. These may be other mlapp-files or m-files. When the app starts it will search the Process folder for files that start with 'process_'. The word following this will be the name listed in the dropdown.[2] So a file with the name process_Segment.mlapp would be listed in the dropdown as Segment. Some modules, like the Segment Module, may have Mini-Modules. An example would be process_Segment_TurnsAPDM.m. This script would not be listed in the BAR App but would be recognized as a method of the Segment Module. Which directories the app looks in for these files is determined in the Configuration tab. Functionally there is little difference between the Processing and Analysis tabs and methods. Data processed with a Processing Module are expected to produce other pro, or raw, data. Data processed with an Analysis Module are expected to produce ana or res data. Once an item is selected from the dropdown select the 'Process' button[3] to run it.

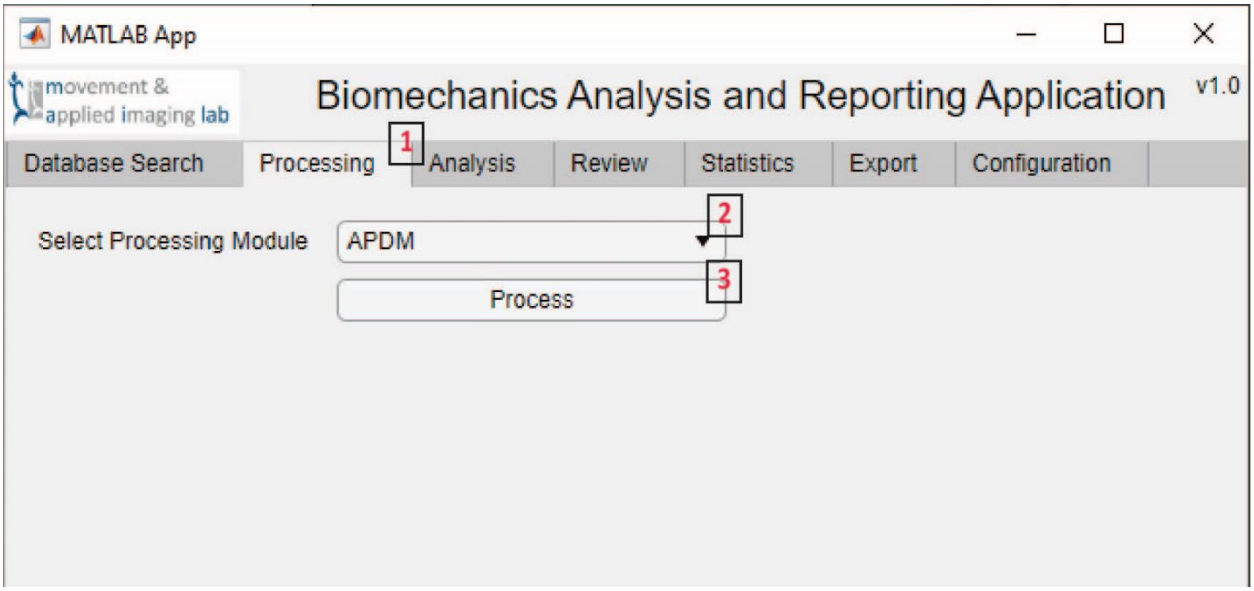


Figure 15 Processing tab of the BAR App

1. Processing tab within the BAR App.
2. Lists the dynamically found process modules.
3. Starts the selected processing module.

Analysis

The Analysis tab[1] is used to run Analysis Modules. These may be other mlapp-files or m-files. When the app starts it will search the Analysis folder for files that start with 'analysis_'. The word following this will be the name listed in the dropdown.[2] So a file with the name analysis_Example.mlapp would be listed in the dropdown as Example. Unlike the Process Modules these methods have not included further dynamic function calling. This is due to the custom and specific nature of some analyses and their reliance on other projects. Analysis Modules are not very different from Process Modules except they are expected to produce ana or res results that would be used for figures (ana) or exported for statistics (res). Once an item is selected from the dropdown select the 'Run Analysis' button[3] to run it. Once the analysis has been performed in the analysis module it will save the data back into the BAR App. It will also automatically prompt the user to save the data to the Results folder of the Working Database. This window can be closed; data will still be saved within the app but not to storage.

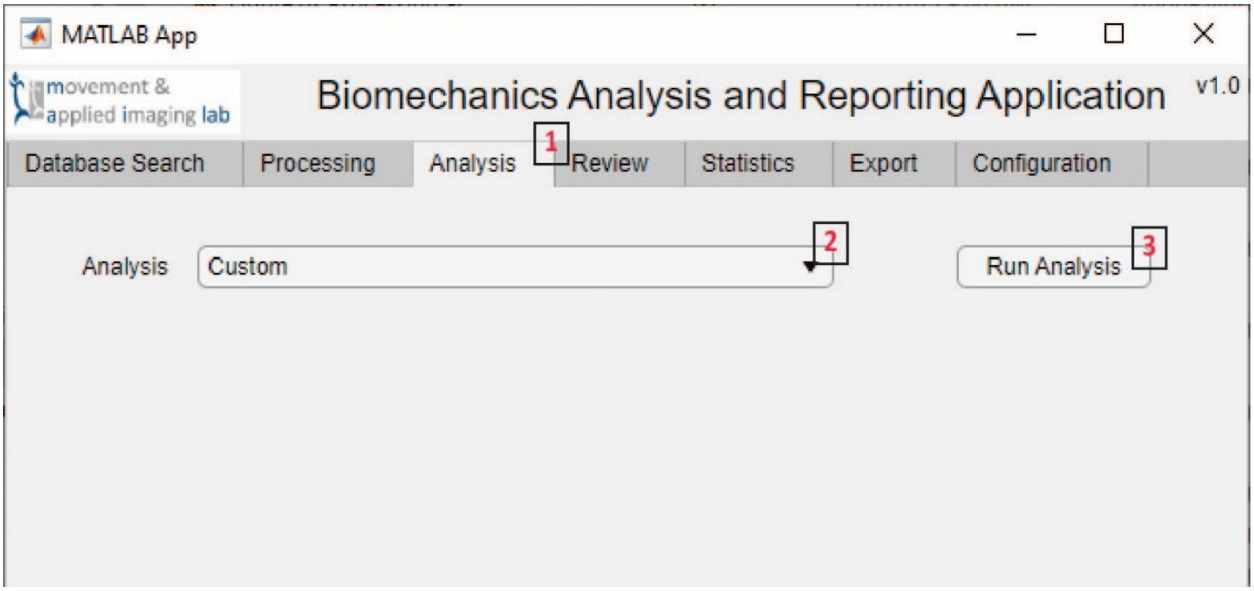


Figure 16 Analysis tab of the BAR App

1. Analysis tab within the BAR App.
2. Lists the dynamically found analysis modules.
3. Starts the selected analysis module.

Review

The Review tab[1] dynamically runs Review Modules similar to the Process and Analysis tabs. At the moment there is one Review Module that will also dynamically identify figure scripts. This one module may eventually be incorporated directly into the BAR App. Similar to the Process and Analysis tabs the Review modules are dynamically found. The file names must be review_<Name>.mlapp, or they may be m-files. The names will end up listed in the dropdown[2]. Make the selected and then click the Review Analysis button to run it[3]. The main goal of this tab is to review data and produce figures. More on the single Review General module and the figures will be explained in the Review Module section.

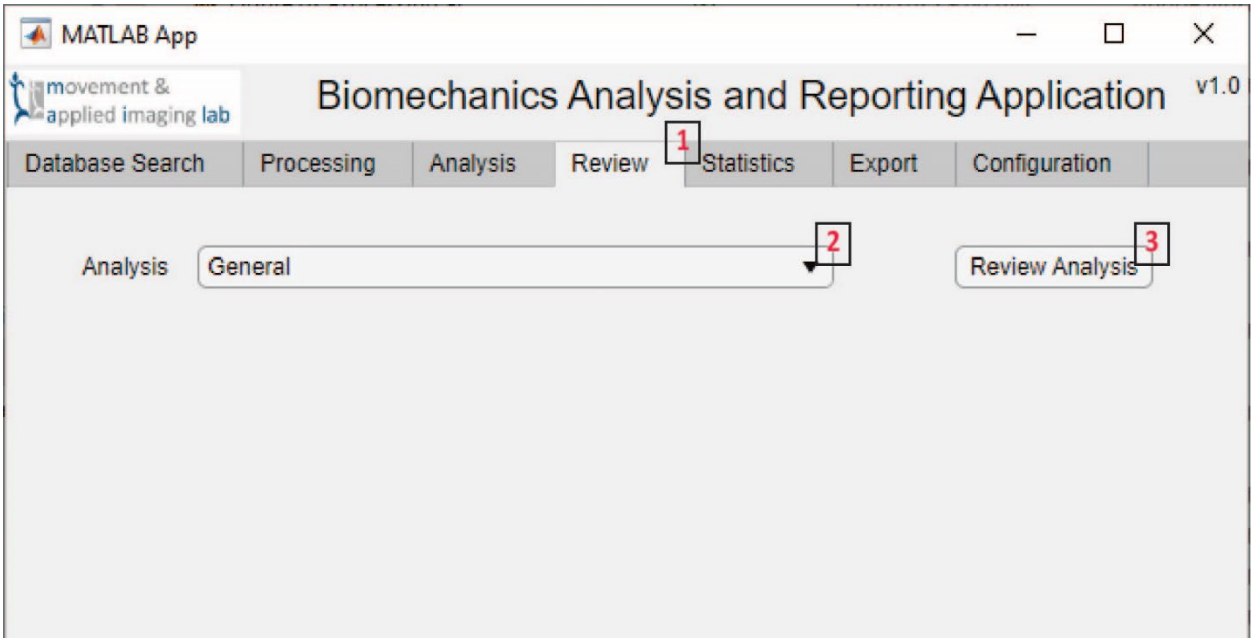


Figure 17 Review tab of the BAR App

1. Review tab within the BAR App.
2. Lists the dynamically found review modules.
3. Starts the selected review module.

Export

The Export Tab[1] is used to send data from the app to an xlsx-file in a long format that is ready for statistics in another software. First the user must click the 'Find eligible exports' button.[2] This will find what data types are present in the app (raw, pro, ana, res). Select one of these[3] and select a format option from the from the dropdown[4]. The General format should work for most datasets. Then press the 'Format loaded results for export' button.[5] There is also an option for means to be calculated or not.[6] Selecting 'No' will cause all values in the app to be exported. This may produce a very long table if the raw or pro data is selected. Selecting 'No' will cause all individual values to be exported. This is useful when the app is used to combine results data from very many files produced by another software. A good example are ascii exports from C-Motion Visual3D that contain the peak joint angles or average step times from a biomechanics evaluation. Formatting the data may take a moment for a large amount of data but is generally quick. Once it has been formatted the results are displayed in the table so they can be reviewed[7] before exporting[8].

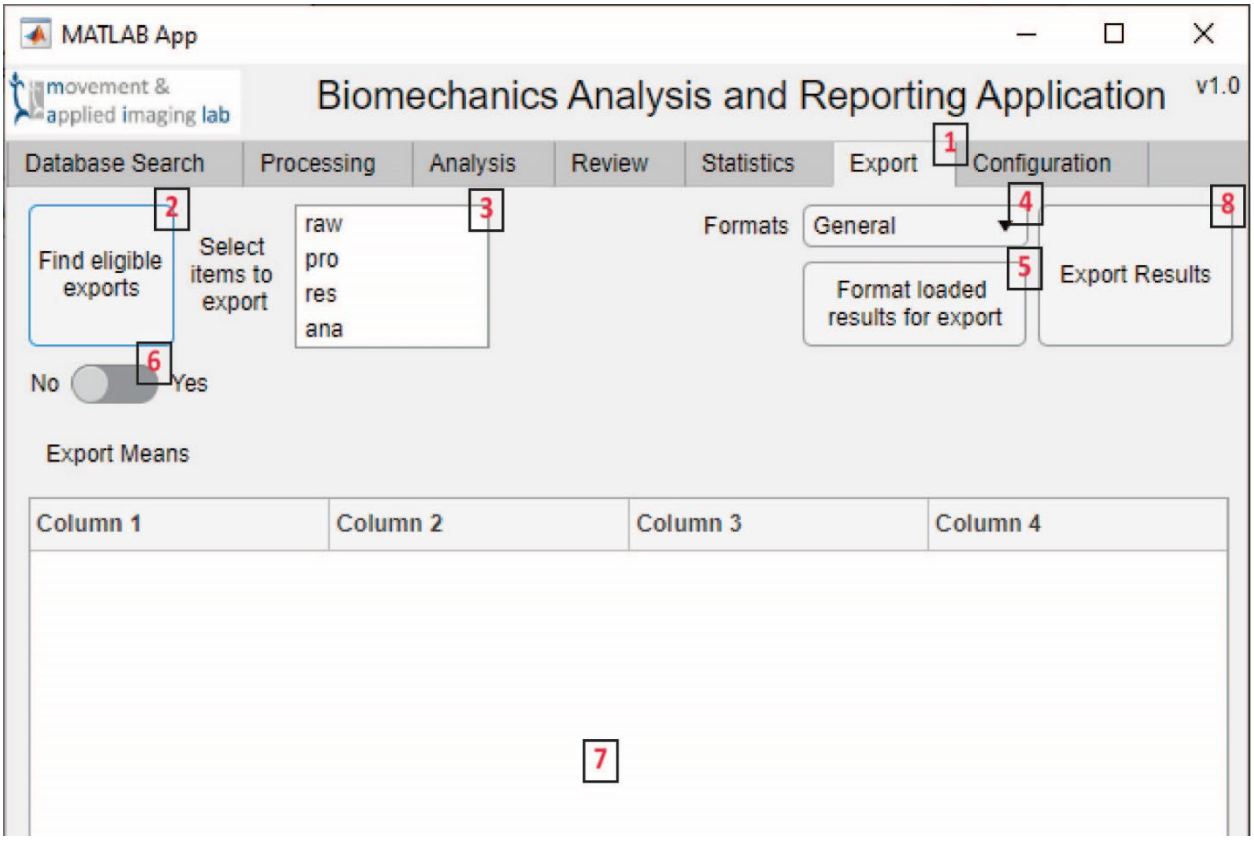


Figure 18 Export tab of the BAR App

1. Export tab within the BAR App.
2. Identifies the data types in the current data.
3. Lists the found data types.
4. Specifies the format of the export.
5. Export the results to a xlsx-file.
6. Formats the data for export according to the selected format.
7. Tells the app if means of repeated values should be calculated.
8. Displays the formatted data if it is in table format.

Configuration

The Configuration tab[1] was created to help run the app across different computers and between projects. The tab contains four text fields. These contain the directories the app will search to find Load[2], Process[3], Analysis[4] and Review[5] methods. Directories can be typed or pasted into the fields and must be listed on separate lines. After being changed the 'Update Configuration' button[6] needs to be clicked to update the app. This will update the various options across the tabs. In the example below you can see each field has a 'Beta' folder and duplicate folders with a different username. This is an example of how the same methods can be used across computers by the same user while keeping confidential code separate from the public repository. There is a config.xml file that contains these directories and other information used by the app. This file should be located in the same folder as the BAR_App.mlapp file. If it is not present the app will create a new blank configuration.

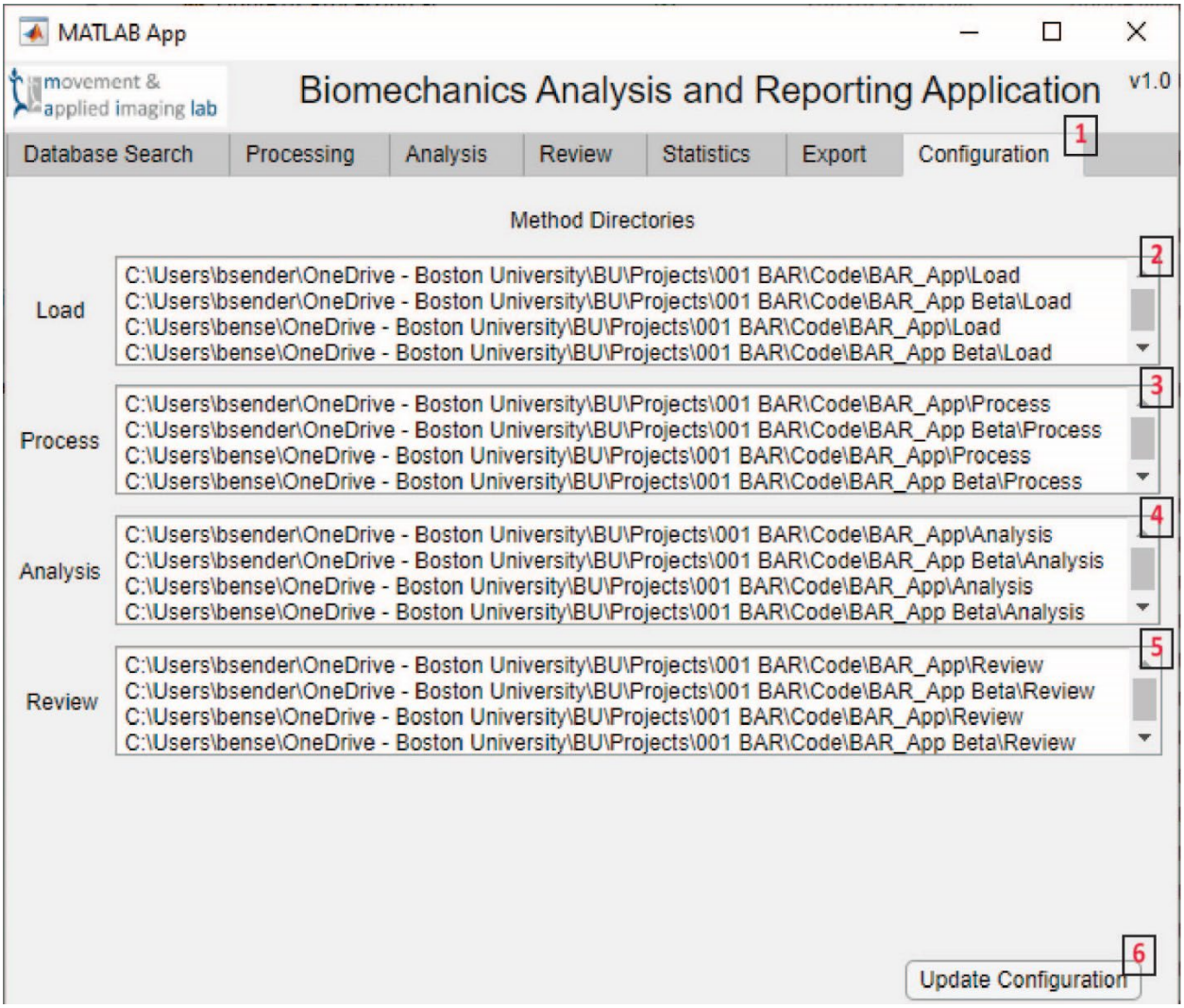


Figure 19 Configuration tab of the BAR App

1. Configuration tab within the BAR App.
2. Lists the directories containing load methods.
3. Lists the directories containing Process methods.
4. Lists the directories containing analysis methods.
5. Lists the directories containing review methods.

Process Modules

Groupings

This module is run from the Database Search tab of the main user interface. It is used to help organize group information used to identify data and perform statistics. It could be subject numbers, visit dates, analysis methods, project names, experiment names, etc. Groups are found primarily during the data loading process. The module then allows those results to be processed.

When data is loaded from a single file the file name is parsed and separated into separate strings. The object names are also parsed. For either case the app will find '\' and '_' characters and treat them as delimiters to find group information. There is no discrimination during these steps. The groups are found as is. As you can see in the image below the drive letter and user account are also pulled as group information. If a delimiter is repeated, such as '__', this would produce a group filled with empty values.

In the Groupings Module there are generally three methods currently available. First, repeated values in columns[1] or rows[2] can be removed. An example of repeated information in the columns would be the drive letter or the username. Since this information is repeated throughout all the data it does not offer any discrimination among the results and can be removed to shorten the export. Repeated values in rows would be any string that is found at multiple locations within a file path and file name. An example would be a data file called 'S001_20230103.txt' located in a directory called 'C:\Users\bsender\Documents\Example Project\S001\20230103'. Because both strings 'S001' and '20230103' occur twice the extra values can be removed. These exact strings do not have to be present in every row for them to be removed. Removing them helps conserve processing time in the Merge Module and helps conserve space in the export. The second method removes groups from any column of data in the table.[3] The column number can be entered, and the button pressed to remove that column. This was mainly added after the addition of the third method, which allows strings containing dates to be converted to ordinal numbers.[4] The format of the dates needs to be selected as being 'yyyyMMdd' (year-month-day) or 'MMddyyyy' (month-day-year). The column contains the dates and the column they belong to needs to be selected. The dates will be converted from strings to dates using the selected format. Then they will be ordered within each unique value they belong to. Those ordinal numbers will be added as an additional group after the date they were converted from. Currently it is assumed the dates will always be at the file-level. Because the dates are converted into ordinal numbers they may cause conflicts in the Merge Module. An ordinal date number would get matched with a trial number, or other integer. For this reason the second method was added. It is best to remove extra groups before running the Merge Module and to convert any dates before data export. Once the processing is complete make sure to click the 'Save Results and Exit' button to close the app.[5]

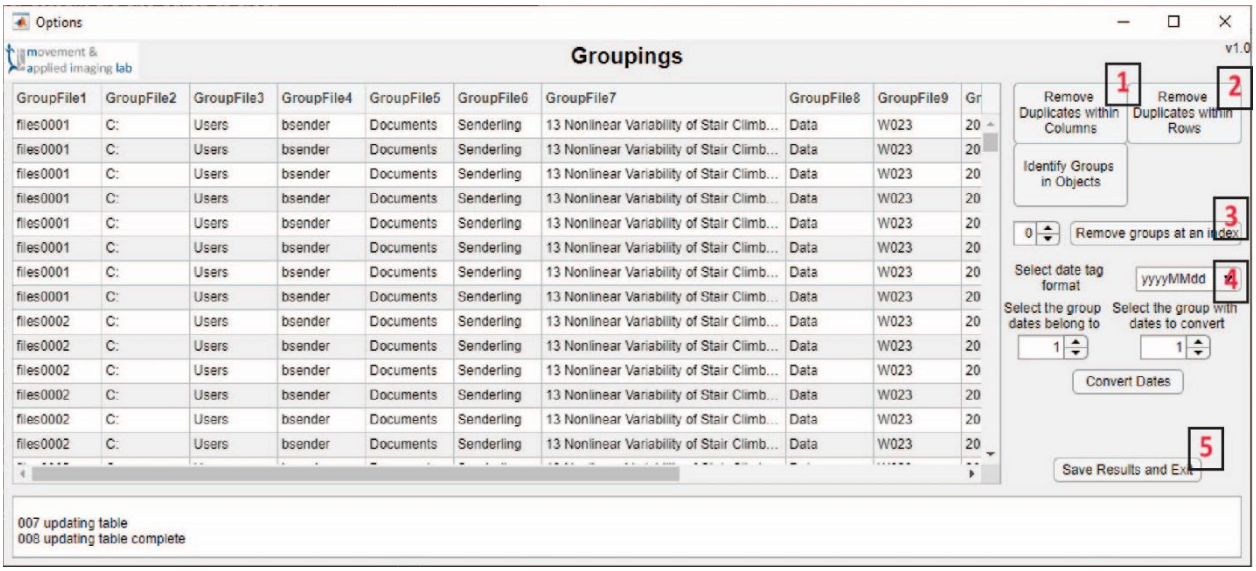


Figure 20 Groupings Module

1. Removes duplicate values within columns.
2. Removes duplicate values within rows.
3. Removes a group at a selected column.
4. Converts dates from a string to an ordinal number.
5. Saves the results and closes the module.

Merge

The Merge Module is used to combine datasets. These may include data from different software that are used together in calculations or data that will be used as covariates for statistics. In the later, the data may not be used in the app but it is easier to combine programmatically instead of manually. The merge can take some time for large data sets or for numerous group information.

The module is currently unidirectional. Data from data set 2 will be moved to data set 1. Data set 1 has been previously loaded into the app while data set 2 has been identified in the file search process. Two options are available for the merge. The File level from data set 1[1] can be matched with the File level for data set 2[2]. Or the File level for data set 1 can be matched with the Object level for data set 2. During the matching process the group information from one data set will be compared to the other. The total number of matching group information is the match score. The user should select the lowest number of common groups that should be required for a match.[3] In some cases multiple files from data set 2 may be combined into data set 1. In this case the duplicate match switch should be set to 'Yes'.[4] However, in many cases there will not be duplicates and 'No' will be selected. This will speed up the matching as previous matches are removed and not considered in future calculations. Click 'Run Match' to start the matching.[5]

In the image below both data sets are being matched at the File level. The first data set has been processed in the Groupings Module and so only has three groups per file. This is a subject number, a date (unconverted) and a trial number. Data set 2 was newly loaded during the module startup and so has not undergone group processing. We can see each of the groups from data set 1 appears in data set 2 once. So the minimal score would be 3.

It is possible that numbers will exist in the group information with different formats. Ex: 1, 01, 001, etc. For these cases there is a check to see if groups can be converted to numbers and produce a match. In this way a file named 'W002_20210324_1' would be matched with 'W002_20210324_001' and have a score of three. This is also why it is better to convert group information to dates after a merge is complete.

This process has been found to take longer and will also take significantly longer when using a network drive. Some good practices include using data stored locally on the computer and processing the group information beforehand. To date the algorithm has been found to work well and is reliable, and does not need to be run more than once. However, it is still being developed. To check the process users can export the matching results to save them for later review.[6] Also make sure to click 'Save Results and Exit'[7] when the matching is complete and satisfactory.

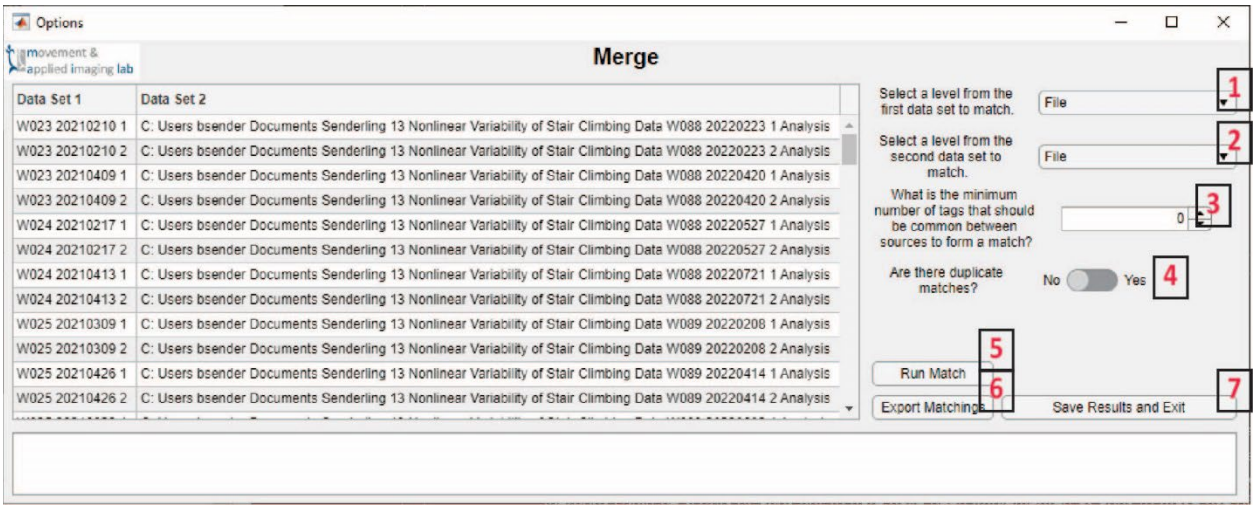


Figure 21 Merge Module

1. Selects the data level in the first data set to serve as the basis for the match.
2. Selects the data level in the second data set to be matched.
3. Sets the minimum score to be considered a match.
4. If no is selected matches will be removed from the second data set to speed up future matches.
5. Starts the matching algorithm
6. Exports a csv-file with the current table results.
7. Saves the results in the main app and closes the module.

Segment

The Segment Module is part of the Process Module but has its own Mini-Modules. These Mini-Modules are found dynamically every time the Segment Module is started. They are expected to have the file name 'process_Segment_<Method Name>.m'. These Mini-Modules are best implemented as scripts with limited user interface. The module's purpose is to split time series into separate segments. These segments are pro results that can be processed or analyzed further.

Segmentation can be performed on Raw or Processed data. Each time the switch[1] is changed the node tree will update with the selection options. These options are the file, object and signal levels of the BAR App data structure. Each time an item is selected in the node tree[2] the module will automatically attempt to segment it with the selected method[3]. If the segmentation is successful, the panels on the left will display the results. The Before[4] panel shows the unsegmented data. Green lines indicate the start of a segment while red lines indicate the end. The After[5] panel shows the segmented data with as many subplots as segments. This figure can be exported to the Figure folder in the Workspace if the 'Yes' option is selected to export the figure.[6] If the method is satisfactory then the button 'Segmental all, save and exit' can be pressed.[7] This will also create figures if 'Yes' is selected.

Segment Mini-Modules are very specific to individual experiment designs. Currently, only one example is included with the app while a couple others are in development. This has the name 'process_Segment_Example.m'.

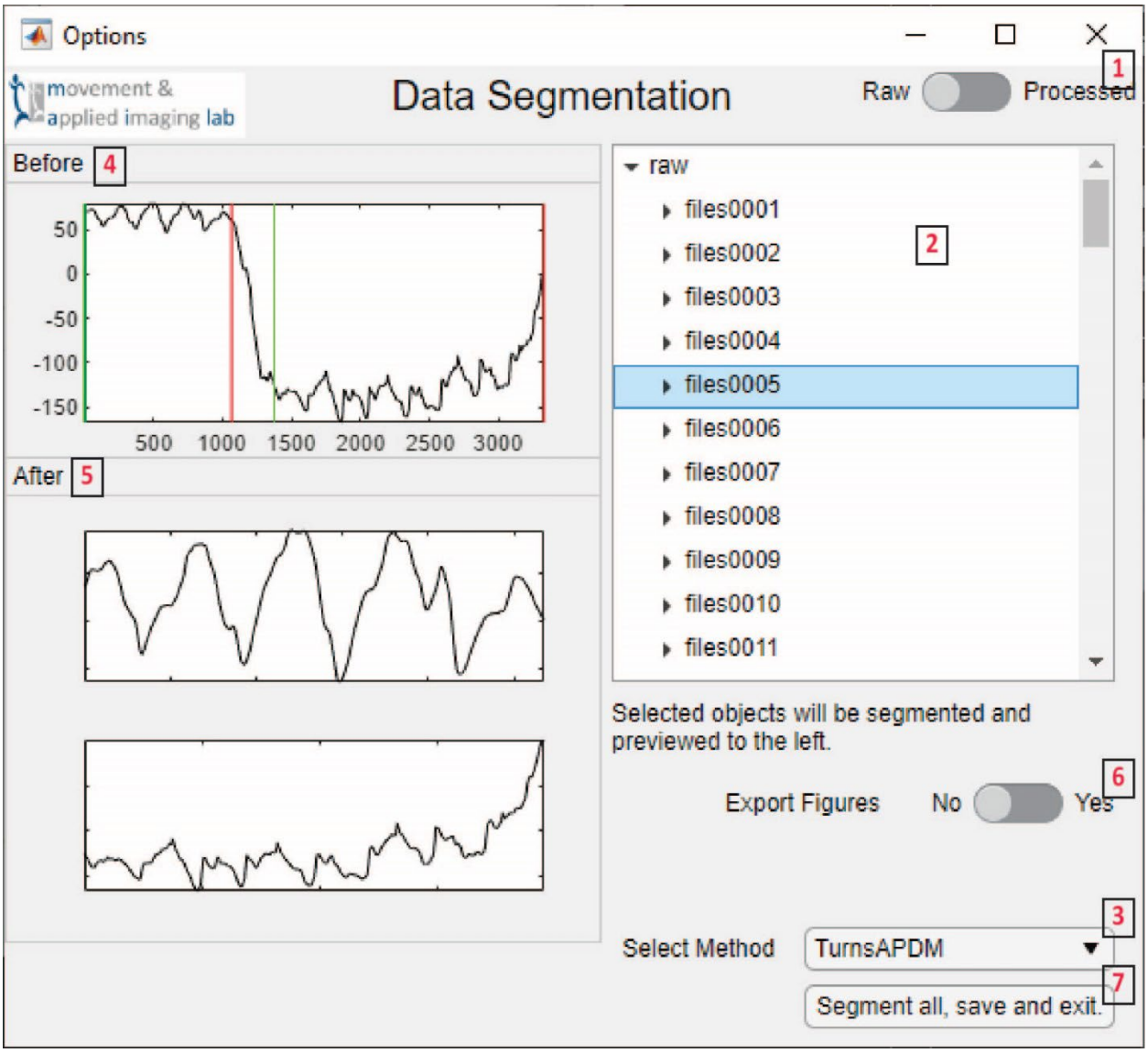


Figure 22 Segment Module

1. Changes the node tree between raw and processed data.
2. Displays a node tree of the current data type.
3. Lists the dynamically found segmentation methods.
4. Shows the unsegmented results with green and red lines for the start and end of each segment.
5. Shows the segmented data in subplots.
6. Will export the figures as jpg- and fig-files.
7. Saves the data in the main app and closes the module.

Treatment

The Treatment Module allows the user to perform data treatment steps on the data ahead of additional processing or analysis. Like the Segment Module this module has Mini-Modules that are dynamically found by the app. They are expected to have names like 'process_Treatment_<Method Name>.mlapp'. These methods could be scripts or apps but to date have been implemented as apps to make use of the user interface.

The Treatment Module directly serves as an interface for users to add and remove treatment steps. Treatment options can be selected from the dropdown menus[1,2] and then added[3] or removed[4]. The 'Process' button[5] will perform all of the steps and close the module. The table on the left side of the app displays the current steps.[6] These are identified by the unique step number and the treatment name. Currently the treatment parameters are not displayed in the table.

The Mini-Modules are very small apps and mainly involve sending values from the user interface back to the Treatment Module. Because of this an example is not provided.

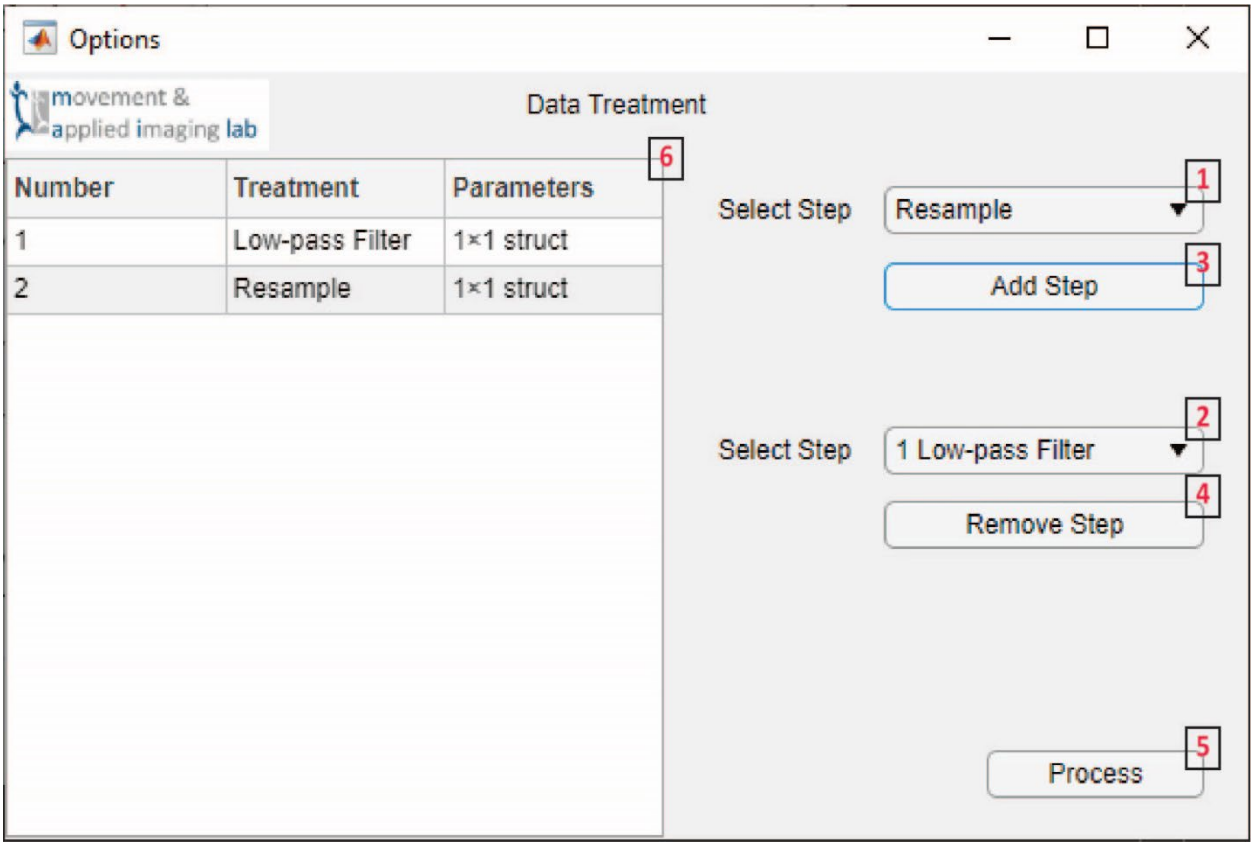


Figure 23 Treatment Module

1. Lists the treatment options to add.
2. Lists the added treatment options for removal.
3. Brings up the mini-module for the selected method.
4. Removes the selected step.
5. Processes all the data with the current steps.
6. Displays the current steps.

Low Pass Filter

This mini-module allows a user to select the parameters for a bidirectional low-pass Butterworth filter. The available cutoff frequencies are between 0 and ½ of the minimum sampling frequency minus 0.1, within the data.[1] Currently there is not an option to select parameters more specifically when varied sampling frequencies are present. The filter order can also be set.[2] This method does use FiltFiltM from the MATLAB File Exchange as it has proven significantly faster than the native methods. Once the parameters are selected click 'Finish' to add the step to the Treatment Module.[3]

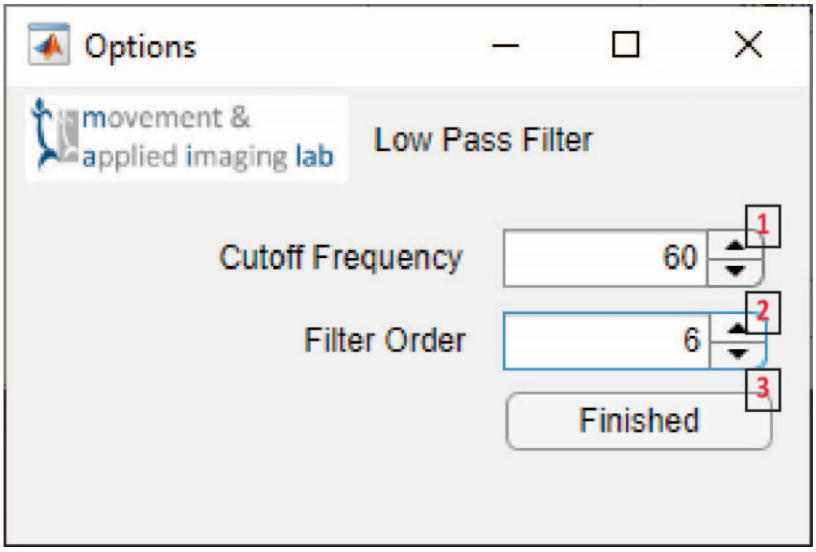


Figure 24 Low Pass Filter Mini-Module

1. Sets the cut-off frequency with an upper limit of half the sampling frequency – 0.01.
2. The filter order of the Butterworth filter.
3. Adds the step with the selected parameters.

Resample

Resample is a Mini-Module that allows a user to select a target frequency[2] to resample the data to. It also displays the current sampling frequency.[1] This method uses the MATLAB function resample, which cannot operate on non-integer sampling frequencies. In these cases the sampling frequency will be rounded to the nearest integer. Once the parameters are selected click 'Finish' to add the step to the Treatment Module.[3]

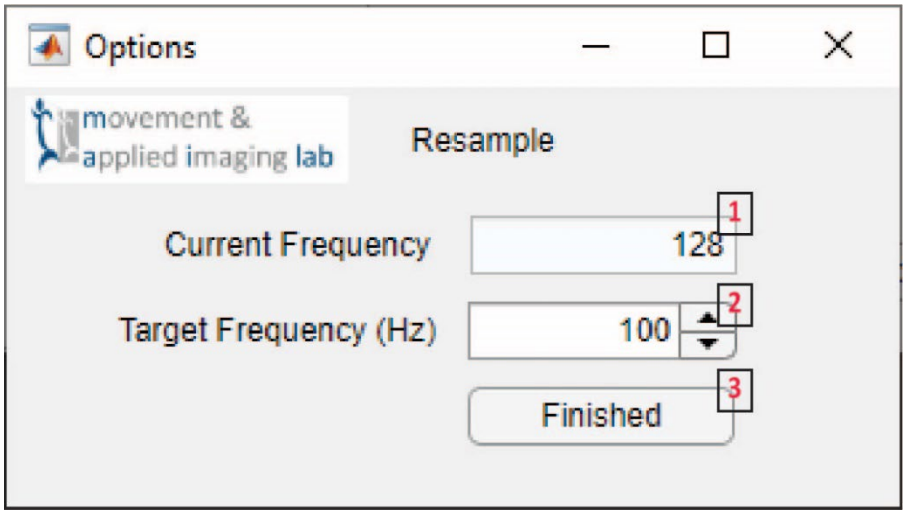


Figure 25 Resample Mini-Module

1. The sampling frequency of the data.
2. The selectable target sampling frequency to resample to.
3. Adds the step with the selected parameters.

Analysis Modules

The follow section describes the different modules available in the current release. Each section starts with 1) a brief description of the module, 2) assumptions regarding the origin of the data, and 3) a description of the results and what their intended use was. Analysis Modules arguably perform the most important functions of the app and produce the results that would be used for figures, statistics and publications. This User Manual is mainly concerned with the operation of the modules and not necessarily their complex scientific or practical use. References are included where relevant to point users to the most useful publications explain those methods. There are both Example m- and mlapp-files that can be followed to help develop your own Analysis Module.

MATLAB Dependencies

This analysis will determine what MATLAB products are needed from a selection of m- and mlapp-files. This will mostly be useful to programmers who which to communicate to others what MATLAB products are needed to run their code. There is no user interface for this analysis module. These data can be exported from the app using the 'Tables' export format.

Assumptions

There are no assumptions or dependencies of this Analysis Module.

Results

The script will produce a list of all the MATLAB products needed to run all the analyzed files. This will look like this.

Name	Version	ProductNumber	Certain
MATLAB	9.11	1	TRUE
Signal Processing Toolbox	8.7	8	TRUE
Statistics and Machine Learning Toolbox	12.2	19	TRUE

Quantitative Sensory Testing

The Quantitative Sensory Testing (QST) Module is a single script that processes specific experimental data recorded at the BU MoveLab. The method itself does not require any user input and allows the results to be collected into a single file. There is no user interface for this analysis module.

Assumptions

Technically this script could run on any data loaded into the BAR App but it is expecting xlsx-Medoc data types. This data was recorded from Medoc software and exported as an xlsx file before being loaded into the app. The experimental procedure it was developed for was a pressure-pain sensitivity test. In this test an algometer is used to apply pressure to an anatomical landmark. As that pressure is increased at a set rate the subject presses a trigger to indicate the first sensation of slight pain. The xlsx-file will contain information for the pressure, sampling times and the button press. It will also contain information on the sequence and trial numbers, and meta data related to the test.

Results

The script will produce the following results.

Variable	Intent	Description
sequenceN	Experiment descriptive	This is the number of the Program Sequence in Medoc that the data comes from. This allows a user to identify a result and trace which trial it came from.
trialN	Experiment descriptive	This is the number of the Program Trial in Medoc that the data comes from. Each Program has a Sequence and within each Sequence are Trials. This allows a user to identify a result and trace which trial it came from.
valuePeak	Results metric	This is the absolute peak of the recorded pressure.
valueEvent	Results metric	This is the instantaneous pressure at the time of the button press.
rSquaredAdjusted	Quality metric	This is an Adjusted R^2 that describes how well the pressure data fits a linear line. The application of pressure it meant to increase at a set linear rate. Deviations from a linear rate will result in a lower value and indicate lower quality methods.
slope	Quality metric	This is the slope of a linear line fit to the data. The pressure should be applied at a set rate. This slope is that rate.

Time Lag

The Time Lag Module is used to calculate time lags from time series data. These are often used for state-space reconstruction and additional analyses. It does require input from the user. Currently, only the ability to calculate time lags[1] and to use one method of time lag calculation is available[2]. In the future additional methods would be available and the option to calculate Average Mutual Information (AMI) would be available. A maximum lag does need to be selected.[3] A good rule of thumb is to use the primary frequency of the data. The maximum time lag is entered as a frame count. Lower values will save processing time but the AMI_Stergiou method used is fairly quick, so higher values are ok. Occasionally, the maximum lag will not be enough to find a time lag and there are options to extend the processing if that is the case.[4] The user can also select if the raw or processed data should be used[5], and they can select which signals to process[6]. Click 'Process' to start the analysis.[7] Once the method is complete the results will be displayed in the table.[8] It can be difficult to tell if these values are good or not but a good rule it to treat any time lags under 10 as suspicious. Often, they will be around 1/4th of the primary frequency of the time series. Click 'Save Results and Exit' to send the results make to the main application.[9] This module does make use of the NONAN Library available on GitHub.

Assumptions

This analysis can be performed on any time series data. It does not need to be a particular type or of a particular nature.

Dependencies

- MATLAB 9.11
- Statistics and Machine Learning Toolbox 12.2

Results

This method will produce the following results.

Variable	Intent	Description
tau1st (analysis)	Figure creation	All the lags at which there is a minimum in the AMI verses lag graph.
tau1_rth (analysis)	Figure creation	The lag at which the AMI drops to 1/5th of its initial value.
ami (analysis)	Figure creation	An array with the lags and their AMI.
tau1st (results)	Results metric	The very first minimum in the AMI verses lag graph.
tau1_5th (results)	Results metric	The lag at which the AMI drops to 1/5th of its initial value.

References

Abarbanel, H. D. I. I., Brown, R., Sidorowich, J. J., & Tsimring, L. S. (1993). The analysis of observed chaotic datain physical systems. Reviews of Modern Physics, 65(4), 1331–1392. https://doi.org/10.1103/RevModPhys.65.1331

Thomas, R. D., Moses, N. C., Semple, E. A., & Strang, A. J. (2014). An efficient algorithm for the computation of average mutual information: Validation and implementation in Matlab. Journal of Mathematical Psychology, 61(September 2015), 45–59. https://doi.org/10.1016/j.jmp.2014.09.001

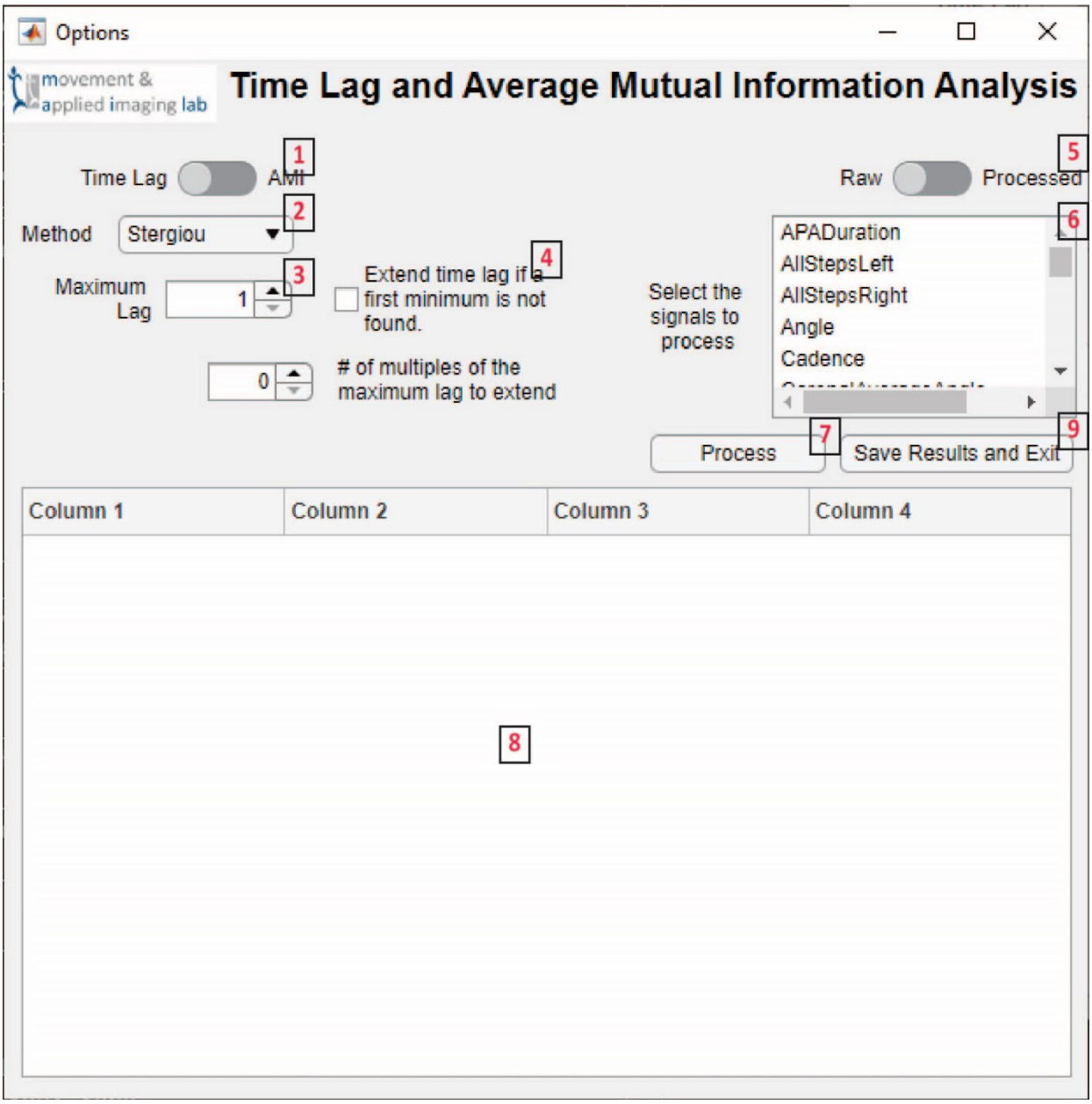


Figure 26 Time Lag Analysis Module

1. Switches between the calculation of time lags and Average Mutual Information. Currently only time lags are available.
2. Lists the available algorithms.
3. Specifies the maximum lag for the algorithm to calculate to.
4. Provides options for extending the algorithm in case a first minimum is not found.
5. Changes the data type used by the algorithm
6. Allows the user to select which signals should be analyzed.
7. Starts the analysis.
8. Displays the analysis results when complete.
9. Saves the results in the main app and closes the module.

False Nearest Neighbor

The False Nearest Neighbor (FNN) Module is used to calculate embedding dimensions from time series data. These are often used for state-space reconstruction and in other methods used to analyze dynamical systems. It can be run on any time series data but does require input from the user. Two of these, the R[1] and A[2] tolerances are set to default values in the app that are recommended in Kennel et al., 1992. The maximum dimension is also set to a convenient default but can be made higher or lower as needed.[3] Making it lower will save processing time, but this method is fairly quick. Likewise, the module can be told to process all dimension or not to.[4] If 'No' is selected the method used to calculate the embedding dimension will exit once a good value is found. A time lag is required to run the app and the user can select whether to use the first minimum or the 1/5th value produced by the Time Lag Module.[5] Lastly, the user can specify if raw or processed should be used[6], and can select which signals to analyze[7]. Click 'Process' to start the analysis.[8] Once the analysis is complete the table will be updated with the results.[9] Embedding dimensions are integers and typically in the single digits for human movement data. Outliers suggest the presence of noise or artifacts. Click 'Save Results and Exit' to send the results make to the main application.[10] This module does make use of the NONAN Library available on GitHub.

Assumptions

This analysis requires a time lag be calculated first through the Time Lag Module. It does need to be run on time series data but it does not have to be of a particular type of nature.

Dependencies

- MATLAB 9.11

Results

Variable	Intent	Description
dE (analysis)	Figure creation	This is a single dimensional array with the % FNN for each dimension.
dim (results)	Results metric	This is the selected embedding dimension determined by the algorithm.

References

Kennel, M. B., Brown, R., & Abarbanel, H. D. I. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. Physical Review A, 45(6), 3403–3411. https://doi.org/10.1103/PhysRevA.45.3403

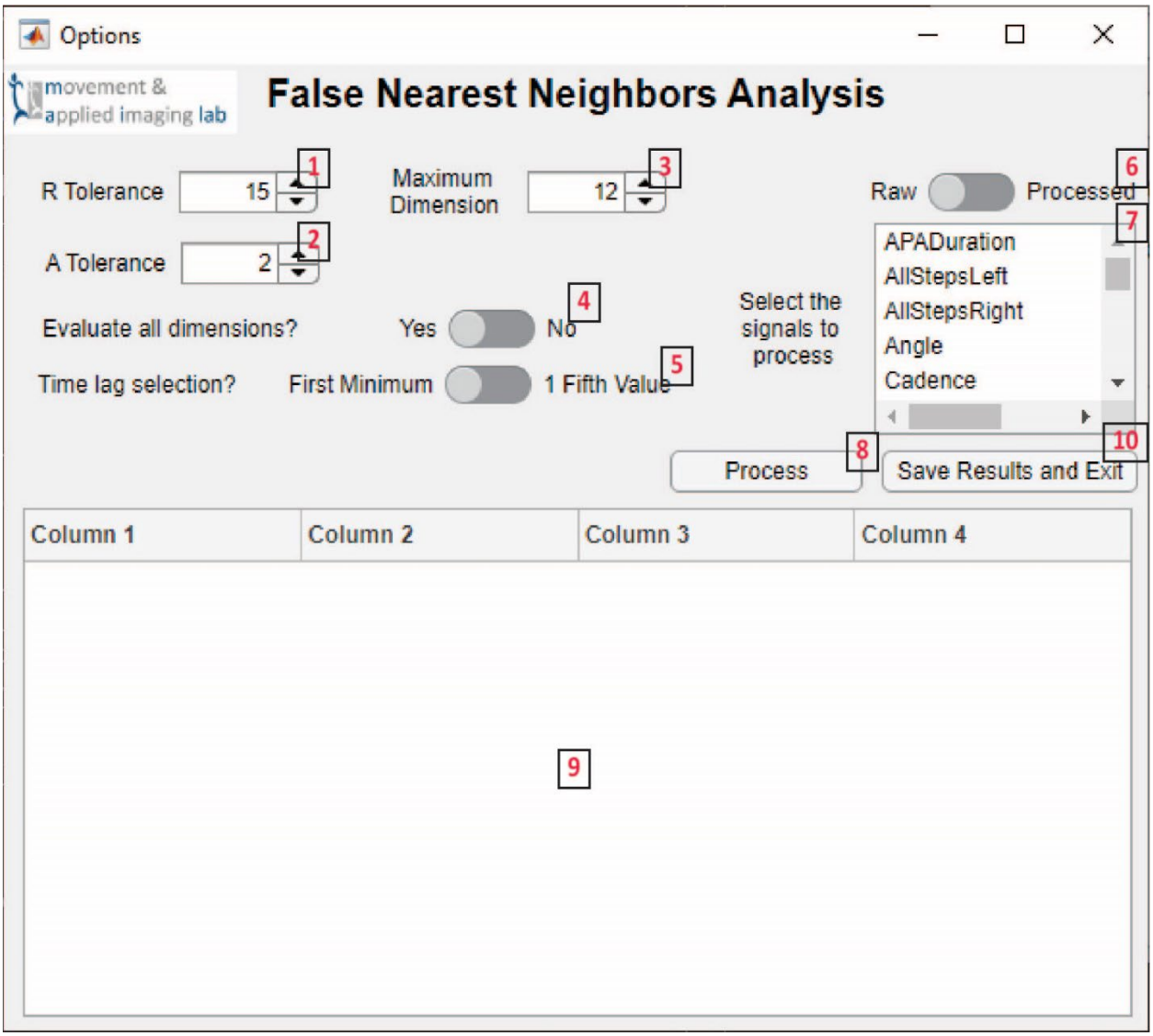


Figure 27 False Nearest Neighbors Analysis Module

1. Sets the R tolerance of the algorithm. The default is recommended in the citation.
2. Sets the A tolerance of the algorithm. The default is recommended in the citation.
3. Sets the maximum dimension for the algorithm to calculate to.
4. If no is selected the algorithm will stop once an embedding dimension is found.
5. Specifies what time lag to use from the Time Lag Module.
6. Specifies if raw or processed data should be used.
7. Allows the user to select signals for the analysis.
8. Starts the analysis.
9. Displays the results once complete.
10. Saves the data in the main app and closes the module.

Recurrence Quantification Analysis

This module performs Recurrence Quantification Analysis on time series data. It produces are fairly large number of results and has a significant amount of secondary input parameters. Not all of these have been tested. The user does need to specify those secondary parameters before running the analysis. The method of RQA can be selected.[1] Currently only the RQA method has been tested within the app but there is also cross (cRQA), joint (jRQA) and multi-dimensional (mdRQA). The user can decide to Z-score the time series before analysis.[2] The time series can be normalized according to its Euclidean distance, maximum distance, minimum distance or not at all.[3] A target recurrence rate can be set or a radius set directly.[4] The first is commonly set to 2.5% and this is the default. It does require the analysis to be performed iteratively to reach the target recurrence rate and so takes longer to process. The later requires the user to select a value according the nature of the data. Because the method it only run once at the selected radius it will run faster. A choice of embedding dimension is required.[5] The user can select either the value unique to the time series, the mode, mean or median across all data, or a separate set value. A time lag is required to run the app and the user can select whether to use the first minimum or the 1/5th value produced by the Time Lag Module.[6] The user can specify if raw or processed should be used[7], and can select which signals to analyze[8]. Click 'Process' to start the analysis.[10] Once the analysis is complete the table will be updated with the results.[11] Lastly, because this method can take a long time to run there is an option to periodically save the results.[9] This will automatically save the BAR App data structure to a mat-file in the Results folder of the Working Directory. The table will also periodically update with the current results. Click 'Save Results and Exit' to send the data back to the main app.[12] This module does make use of the NONAN Library available on GitHub.

Assumptions

This analysis requires both a time lag be calculated through the Time Lag Module and an embedding dimension calculated through the False Nearest Neighbor Module. It does need to be run on time series data but it does not have to be of a particular type of nature.

Dependencies

- MATLAB 9.11
- Statistics and Machine Learning Toolbox 12.2
- Image Processing Toolbox 11.4

Results

Variable	Intent	Description
recurrencePlot (analysis)	Figure creation	This is the full recurrence plot.
RQAresults (results)	Figure creation	This is a Boolean used to indicate if the analysis was successful.
DATA (analysis)	Figure creation	This contains the input time series and is used to create the axis on the recurrence plot.
DIM (results)	Methodological review	The dimension of the output data, maily used for mdRQA.
EMB (results)	Methodological review	The embedding dimension used for state-space reconstruction in the analysis.
DEL (results)	Methodological review	The time delay used for state-space reconstruction in the analysis.

RADIUS (results)	Results metric	The target radius used for the recurrence plot, or the resulting radius found if the recurrence rate was set.
NORM (results)	Methodological review	The type of normalization used.
ZSCORE (results)	Methodological review	A Boolean that is true of the data was z-scored.
Size (results)	Methodological review	The size of the recurrence plot.
%REC (results)	Results metric	The resulting percent recurrence rate if the radius was set or the target recurrence rate if it was specified.
%DET (results)	Results metric	Percent of points on a diagonal line.
MeanL (results)	Results metric	Average length of adjacent recurrent points.
MaxL (results)	Results metric	Maximum length of adjacent recurrent points.
EntrL (results)	Results metric	Shannon entropy of the distribution of the diagonal lines.
%LAM (results)	Results metric	Percentage of vertically adjacent recurrent points.
MeanV (results)	Results metric	Average length of vertically adjacent recurrent points.
MaxV (results)	Results metric	Maximum length of vertically adjacent recurrent points.
EntrV (results)	Results metric	Shannon entropy of the distribution of vertical lines.
EntrW (results)	Results metric	Weighted entropy distribution of vertically weighted sums.

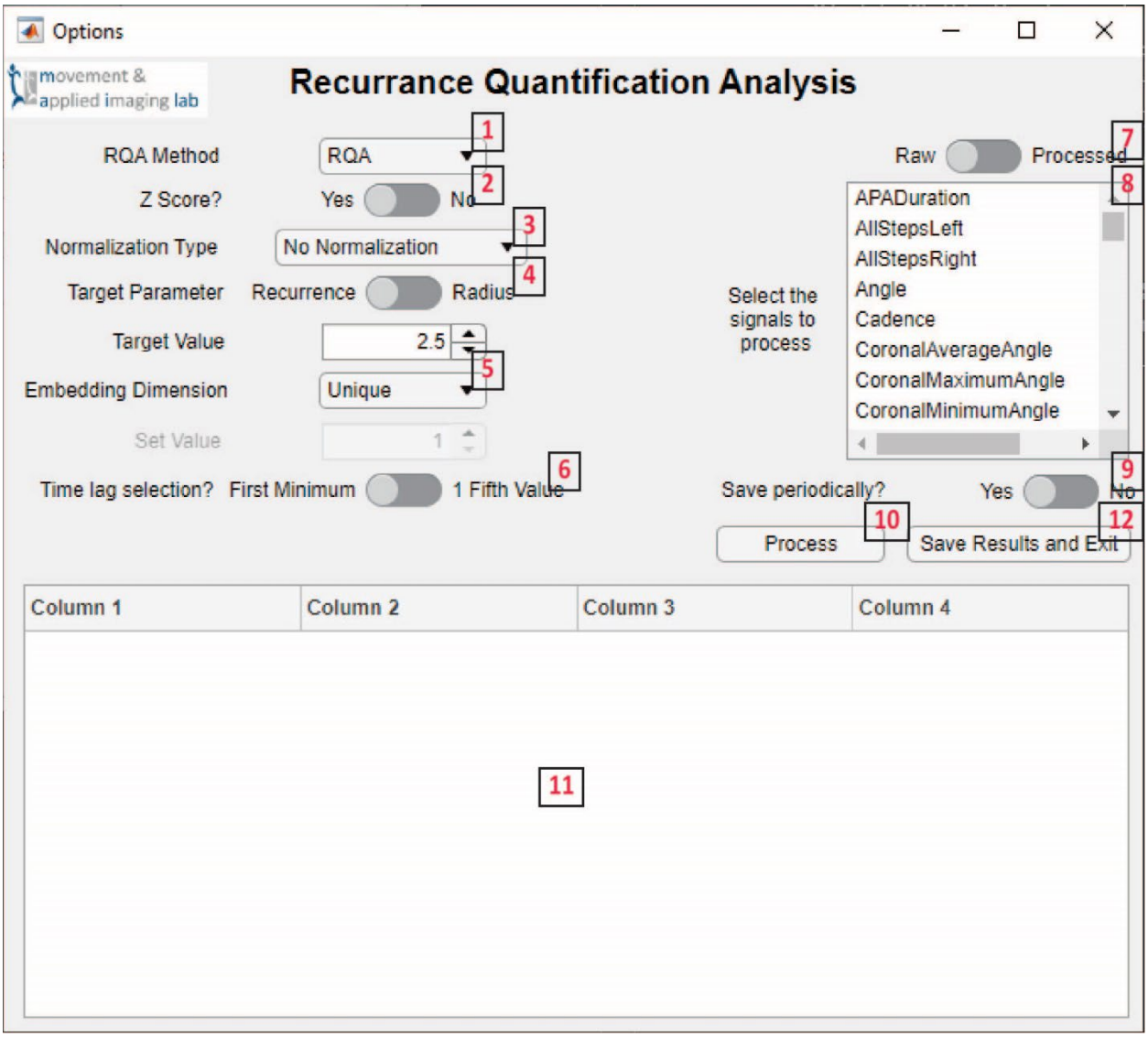


Figure 28 Recurrence Quantification Analysis Module

1. Lists the possible RQA methods.
2. Specifies if the data should be z-scored.
3. Lists the normalization methods possible.
4. Allows the user to set a recurrence radius or a target recurrence rate.
5. Specifies how to use the embedding dimensions from the FNN Module.
6. Specifies which time lag to use Time Lag Module.
7. Specifies if raw or processed data should be used.
8. Allows the user to select signals for the analysis.
9. Tells the app to save the data every 10 files.
10. Starts the analysis.
11. Saves the data in the main app and closes the module.

Review Modules

Review Modules are used to create figures of the data. They are largely organized into one General Module with Mini-Modules being used to create special figures. In future versions it may be incorporated directly into the BAR App.

General Review

This module has a number of dropdowns that allow the user to select the data type[1], figure type[2], where to pull additional axes from[3], the file[4], object[5], signal[6], dimension[7], results measure[8], and data with specific meta data[9, 10]. If Analysis or Review is selected as the Data Type[1] the specific analysis can be selected from the Analysis or Review Type dropdown[11]. Changing these two selections will update the various dropdowns with the available selections. The Plot Type dropdown[2] includes some default figure types but also allows the allowing use of custom plots. Some of these are described in the Time Lag, FNN and RQA figure sections. Click the 'Update Plot' button[12] to create the figure[13] after making your selections. The 'Export Plot' button will export the current figure as a jpg and fig-file.[14] Selecting an additional level to use as additional axes[3] will produce different results depending on the figure. For 2D and 3D plots it will add an addition one or two axis only so a line can be plotted. For 1D and Histogram it will plot additional lines and distributions. Some data can be plotted as an ensemble.[15] When unchecked individual data points and lines will be plotted. If checked the average and standard deviation will be plotted as an average line and grey fill.

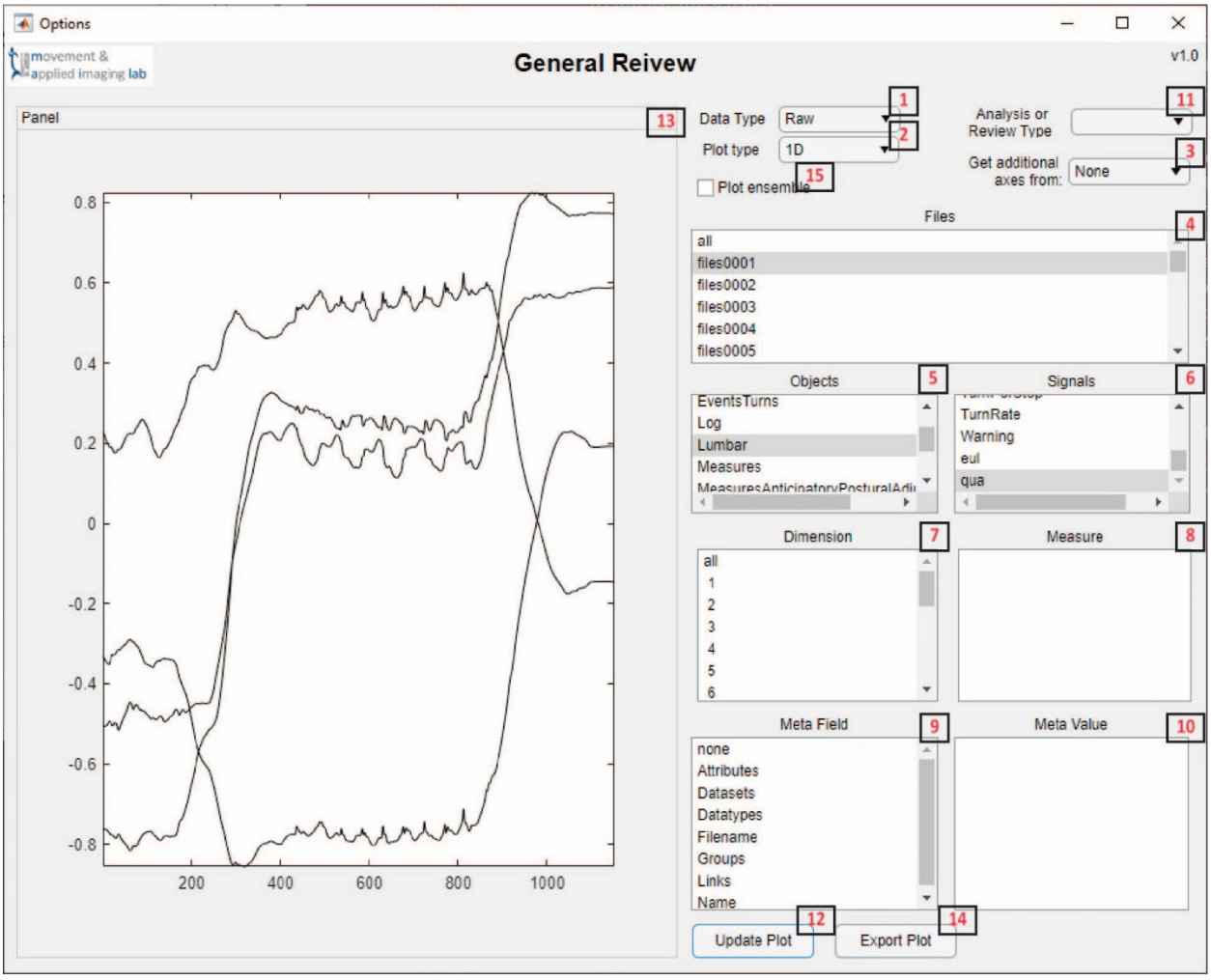


Figure 29 General Review Module

1. Changes the data type data will be pulled from. Changing this will update the other dropdowns.
2. Specifies what type of figure to create.
3. Specifies what to use as additional axes.
4. Lists the files of the current data type.
5. Lists the unique objects found throughout all the files of the current data type.
6. Lists the unique signals found throughout all the files of the current data type.
7. Lists all the dimensions found within any signal.
8. Lists the unique measures found for all signals for the analysis and results data types.
9. Lists the fields found within meta data for all files.
10. Lists the unique values found for all files for the currently selected Meta Field.
11. Changes what analysis or review type to pull the data from. Changing this will update the other dropdowns.
12. Updates the figure using the current selections.
13. Displays the figures.
14. Exports the figure as a jpg- and fig-file.
15. Specifies if an ensemble, or average plot should be made.

Figures

These mini-modules create figures that are copied into the General Review Module. Like other methods they are dynamically found and assumed to have the name figure_<Name>.m. The name will appear as a plot type option in the module. Note that this is separate from the dropdown to select the analysis or review type.

1D

This is a one-dimensional (1D) plot with the data as the y axis and the index number as the x axis.

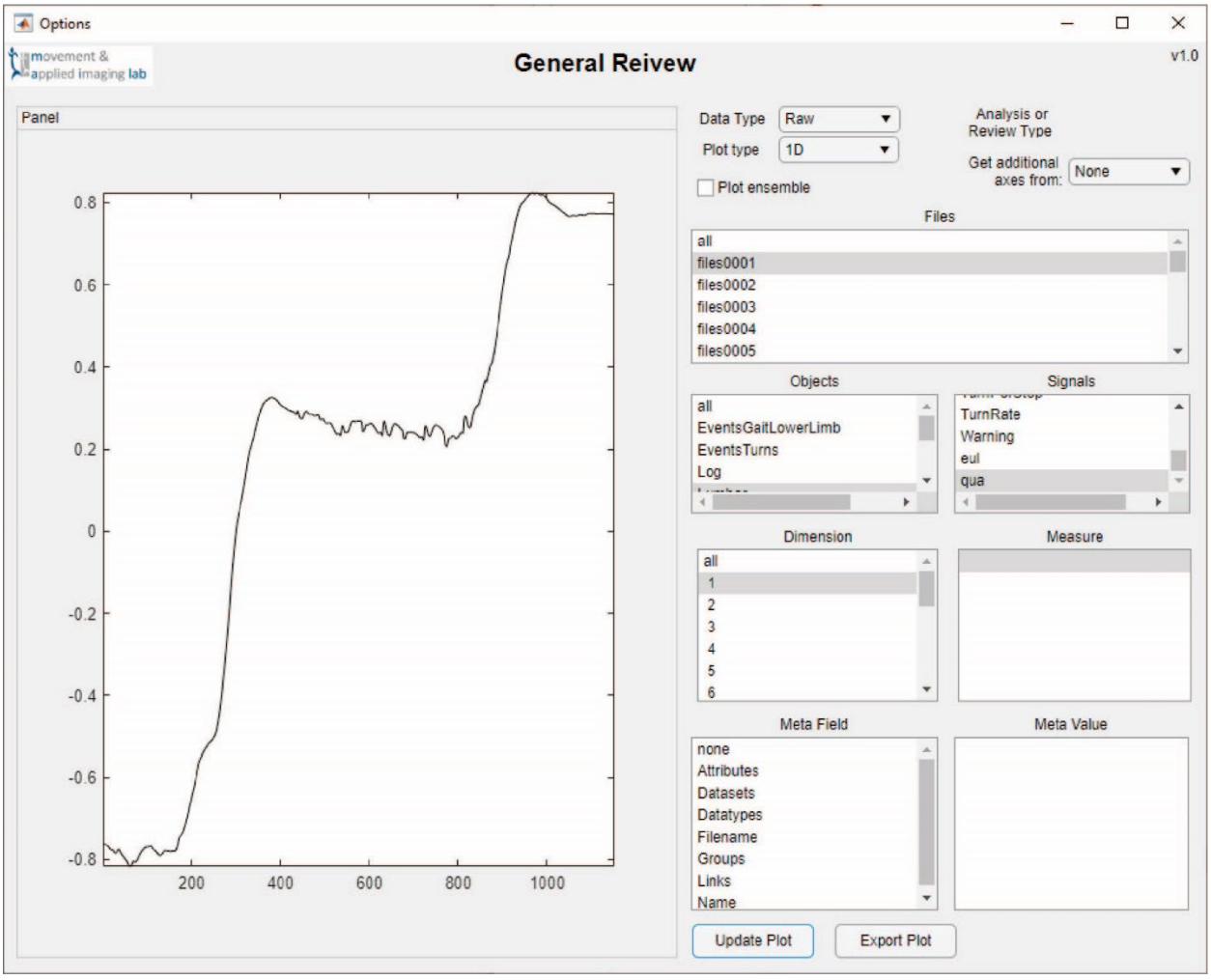


Figure 30 1D Figure mini-module

This is an example 1D figure created from one file, object, signal and dimension.

2D

This is a two-dimensional plot. The first signal or measure selected will be used as the x axis while the second selected is used as the y axis. The dropdown to use additional axis must be selected and two items must be selected.

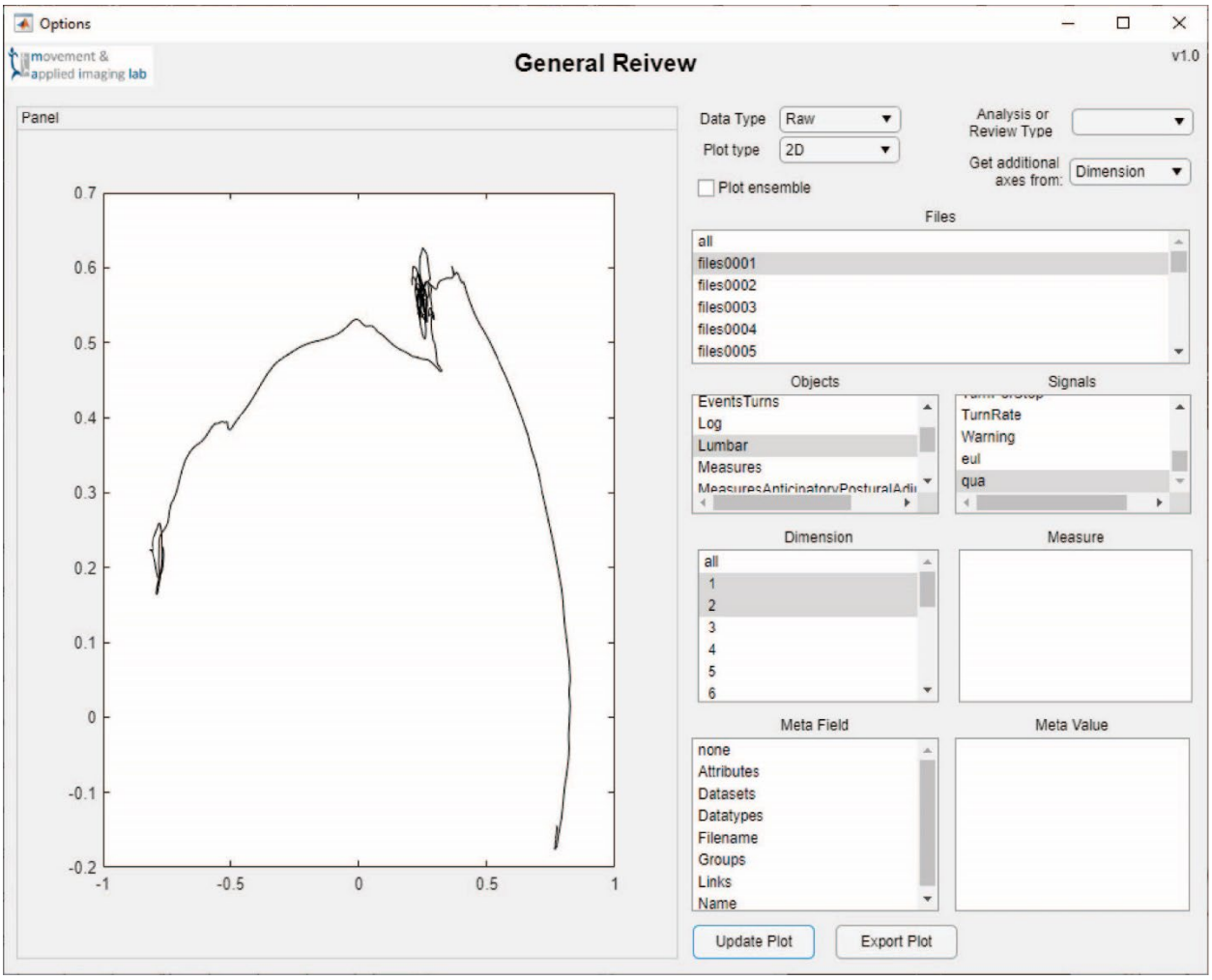


Figure 31 2D Figure mini-module

This is an example 2D figure created from one file, object and signal. There are two dimensions selected to use as the x and y axes.

3D

This is a three-dimensional plot. The first signal or measure selected will be used as the x axis, the second selected as the y axis and third as the z axis. The dropdown to use additional axis must be selected and two items must be selected.

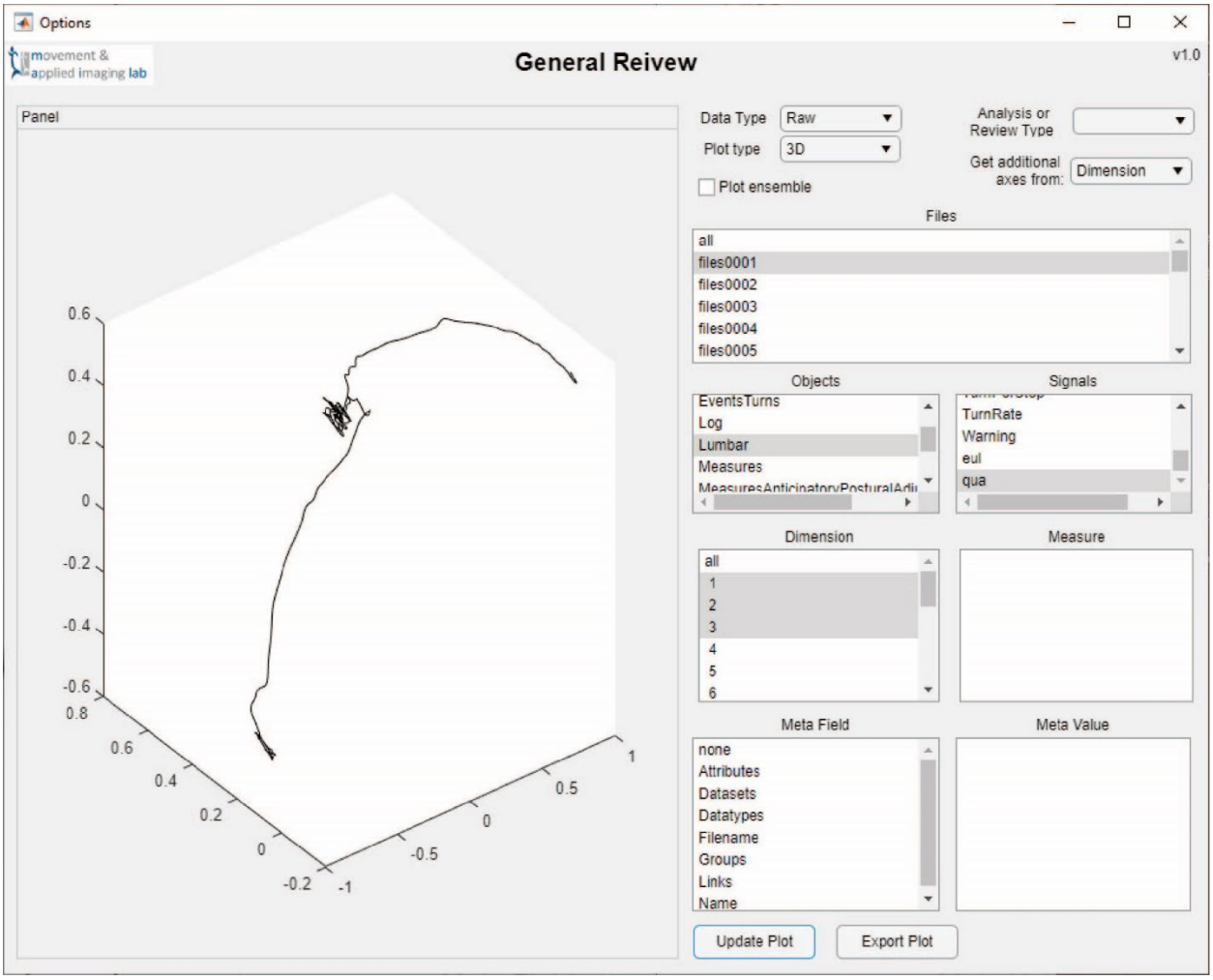


Figure 32 3D Figure mini-module

This is an example 3D figure created from one file, object and signal. There are three dimensions selected to use as the x, y and z axes.

Histogram

This is a histogram of the selected data. It will work on individual files, objects and signals, but is best used on large amounts of data. If additional dimensions are selected, they will be plotted as separate distributions.

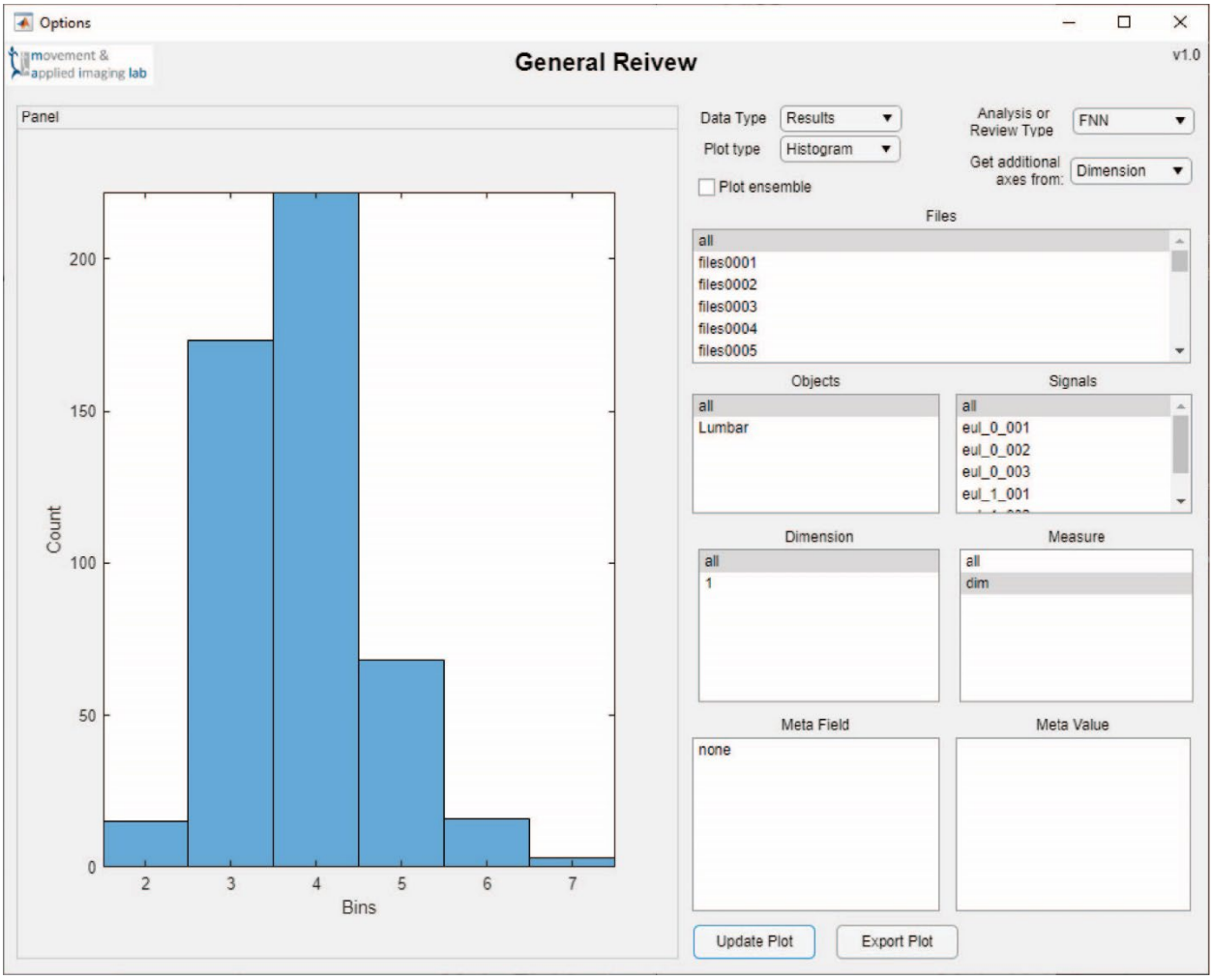


Figure 33 Histogram Figure mini-module

This is an example Histogram figure created from all the embedding dimensions from the FNN Analysis Module.

Time Lag

This figure is specific to Time Lag Module results. It will plot the AMI for each selected signal against its lag. The 1st minimum will be circled in red and the 1/5th value circled in blue. In the image below the ensemble option is checked. If unchecked the additional minimums would be plotted as single black dots.

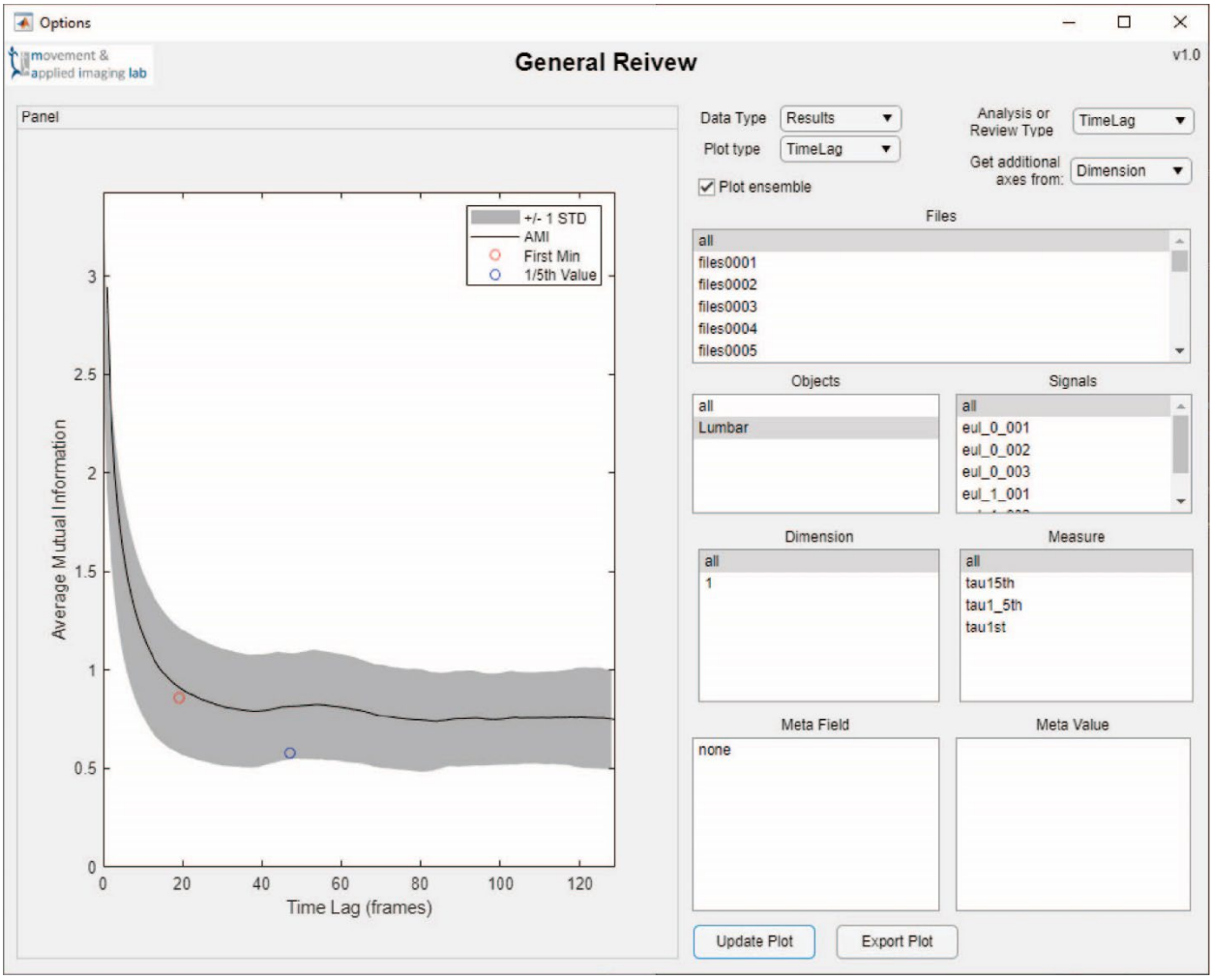


Figure 34 Time Lag Figure mini-module

This is an example Time Lag figure created from all the results from a Time Lag Analysis Module. Here ensemble is checked so the average and standard deviation were plotted. If it was not then all the individual results would be plotted. This plot has the Average Mutual Information plotted against the time lag. The red circle highlights the average first minimum. The blue circle highlights the average 1/5th value. The figure created will be specific to the current data selection.

False Nearest Neighbors

This figure is specific to the False Nearest Neighbor Module. It creates two plots using the selected data. The top plot shows the % FNN against the embedding dimension and the selected value circled in red. This plot will use the selected items. The lower plot shows the distribution of embedding dimensions for only the selected items.

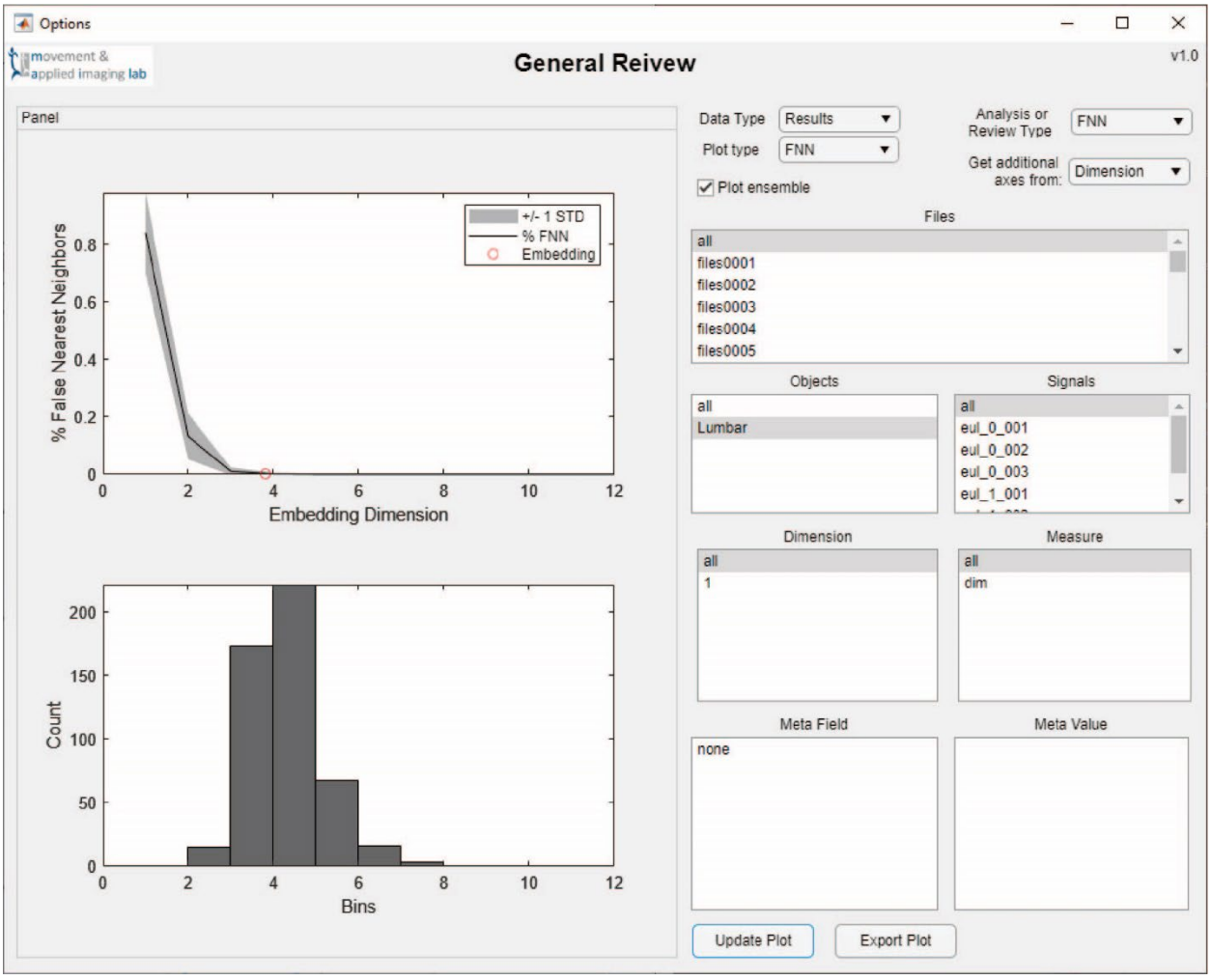


Figure 35 FNN Figure mini-module

This is an example FNN figure created from all the results from a FNN Analysis Module. The top plot shows the % FNN against the embedding dimension. The red circle highlights the average embedding dimensions. Since ensemble is checked the averages and standard deviation were plotted. Otherwise, the individual values would be plotted. The lower plot shows a histogram of the embedding dimensions. This is the same as Figure 20. Both plots in this figure are specific to the selected data.

Recurrence Quantification Analysis

This figure is specific to the Recurrence Quantification Analysis Module. It produces a number of figures. The first two at the top are the binary and weighted recurrence plots. For their x and y axes the time series are show. These plots are only shown for the first valid result selected. Below these the module will display histograms of each selected measure. These include only the selected data.

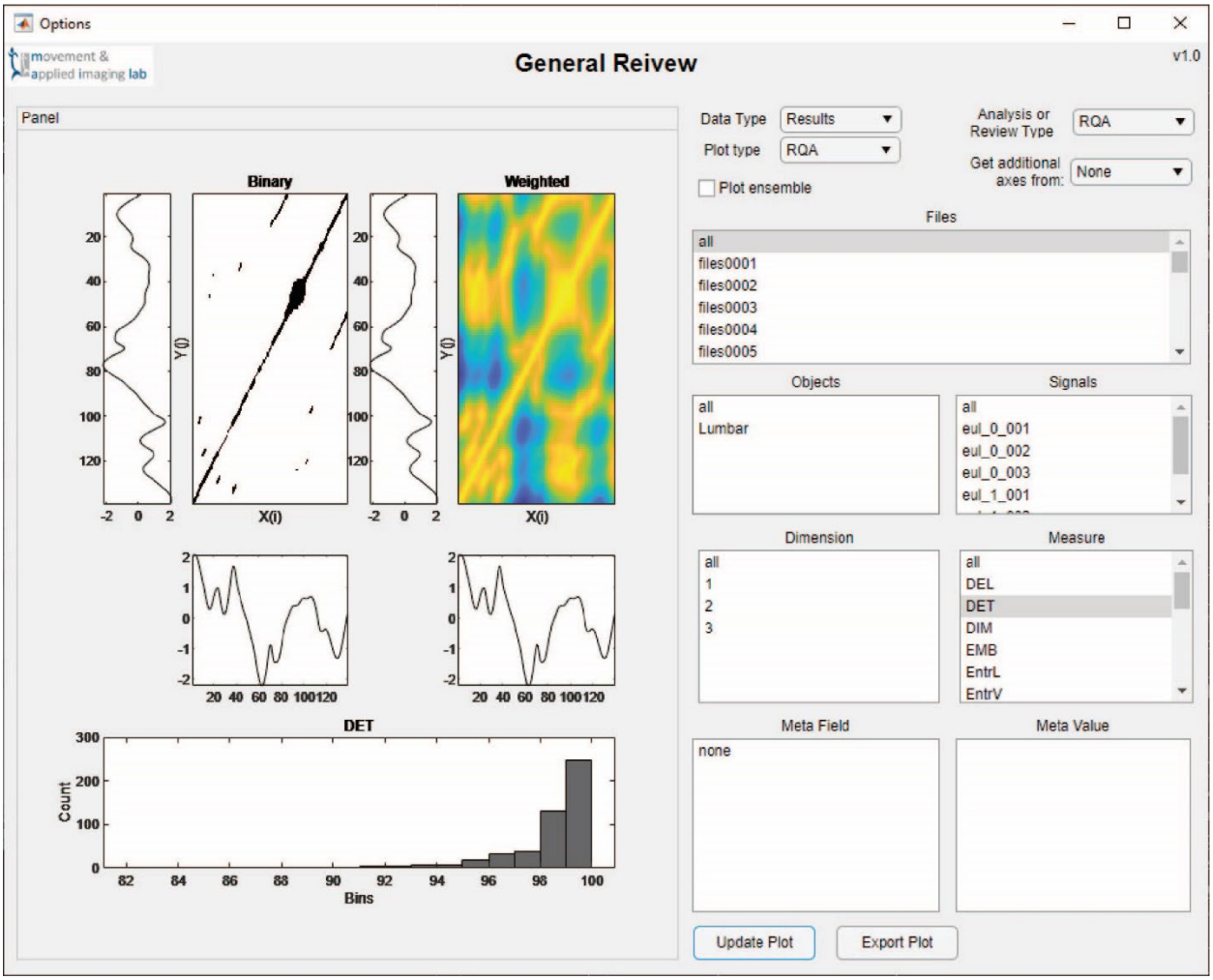


Figure 36 RQA Figure mini-module

This is an example RQA figure created from all the results from a RQA Analysis Module. The top six plots are really two figures. Each has the recurrence plot shown with the time series displayed as the x and y axes. The left figure shows the binary results while the right is the weighted results. Since all files, objects and signals are selected, these plots are only for the first signal with valid results. Below these two figures are displayed histograms of the selected measures. Here one measure is selected so one histogram is shown.

Appendix A Data Types

File Type	Equipment Type	Description
agd	Actigraph	Actigraph uses agd-files in much of its processing in ActiLife. However, they are not the csv exports with the results. These files are not read by the BAR App but can be copied and moved.
csv (unavailable)	Actigraph (unavailable)	These are the csv exports from Actigraph ActiLife. It does include the spreadsheet exports called: DailyDetailed, DailyTotals, HourlyDetailed, HourlyTotals, SedentaryAnalysis, SleepScores and WearTimeValidation. It does not include the Variables spreadsheet. (unavailable)
csv	BAR	These are csv exports from the BAR App that can be loaded back into the app.
csv	Delsys	These are comma separated values exported from Delsys using the Delsys File Utility. When exporting data it is required the option to include headers is checked. Delsys has a number of sensor and file types so this code may not work for all csv-Delsys files.
h5	APDM0Meta	This data type is the meta data from an h5 APDM file. These can be the raw data recordings or processed data from APDM. Only the meta data is loaded to make use of its contents. This increases the speed of the code compared to loading the entire h5 file.
h5	APDM1Raw	This includes both the raw data and the meta data from an h5 APDM file. This is only the raw acceleration, gyroscope and magnetometer and not the quaternions.
h5	APDM1RawLumbar	This includes the meta data and only the raw data from a Lumbar sensor from an h5 APDM file. It includes the acceleration, gyroscope and magnetometer but not the quaternions. Loading only this sensor will save computation time and file size.
h5	APDM2Results	This includes all the data from an h5 APDM results file. This does not include processed data and was produced after processing a raw h5 file with an executable. Meta data is included.
h5	APDM3Qua	This includes the quaternions from all sensors in an h5 APDM file. It also includes the meta data. Raw data is not included.
h5	APDM3QuaLumbar	This includes only the quaternions for the Lumbar sensor from an h5 APDM file. Meta data is included.
m	MATLAB	These are the scripts used in MATLAB.
mat	BAR	These files contain a single BAR App data structure.
mat	QTM	MATLAB data file exports from Qualisys QTM.
mlapp	MATLAB	This is the file type of the MATLAB Applications used in App Designer.
qtm	QTM	These are the motion capture recordings from Qualisys QTM. They are not loaded into the app but allow the app to move and copy them.
txt	V3D	These are text file ascii exports from C-Motion Visual3D.

xlsx	Medoc	These are Excel files exported from Medoc software. There are very few options or alternate versions but these may vary with the testing equipment and Program configuration.