Biomechanics Analysis and Reporting

Application

User Manual

Contributors:

Ben Senderling, Boston University

Version 1.0

Contents

[Intent 3](#_Toc122445302)

[Scope 3](#_Toc122445303)

[Overview 4](#_Toc122445304)

[Future Work 4](#_Toc122445305)

[Main Application 4](#_Toc122445306)

[Database Search 5](#_Toc122445307)

[Working Database 5](#_Toc122445308)

[Processing 6](#_Toc122445309)

[Analysis 6](#_Toc122445310)

[Review 6](#_Toc122445311)

[Statistics 6](#_Toc122445312)

[Export 6](#_Toc122445313)

[Configuration 6](#_Toc122445314)

[Analysis Modules 6](#_Toc122445315)

[Custom 6](#_Toc122445316)

[MATLAB Dependencies 6](#_Toc122445317)

[Assumptions 6](#_Toc122445318)

[Results 6](#_Toc122445319)

[Quantitative Sensory Testing 7](#_Toc122445320)

[Assumptions 7](#_Toc122445321)

[Results 7](#_Toc122445322)

[Time Lag 7](#_Toc122445323)

[False Nearest Neighbor 8](#_Toc122445324)

[Recurrence Quantification Analysis 8](#_Toc122445325)

[Loading Data into BAR 8](#_Toc122445326)

[Overview 8](#_Toc122445327)

[Process Modules 8](#_Toc122445328)

[Groupings 8](#_Toc122445329)

[Merge 8](#_Toc122445330)

[Segment 8](#_Toc122445331)

[Treatment 8](#_Toc122445332)

[Review Modules 8](#_Toc122445333)

[General 8](#_Toc122445334)

[Figures 8](#_Toc122445335)

[Quality Review 8](#_Toc122445336)

[Appendix A Data Types 9](#_Toc122445337)

[Appendix B Validation 11](#_Toc122445338)

[Main Application 11](#_Toc122445339)

[Database Search 11](#_Toc122445340)

[Processing 11](#_Toc122445341)

[Groupings 11](#_Toc122445342)

[Merge 11](#_Toc122445343)

[Segment 11](#_Toc122445344)

[Analysis 11](#_Toc122445345)

[Quantitative Sensory Testing 11](#_Toc122445346)

[Time Lag 11](#_Toc122445347)

[Average Mutual Information 11](#_Toc122445348)

[False Nearest Neighbor 11](#_Toc122445349)

[Recurrence Quantification Analysis 11](#_Toc122445350)

[Review 11](#_Toc122445351)

[Export 11](#_Toc122445352)

# 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 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 and quality review and 4) export of the data. Its design was meant to be 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 the app. Some modules are considered central to the application. These include 1) process\_Groupings, 2) process\_Merge and 3) review\_RAW. Modules fit into categories named 1) Analysis, 2) Load, 3) Process and 4) Review. These are also the major folders used by the app. Another folder called Subroutines is meant to house low level functions that may have widespread use across other modules.

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.

Load modules are mostly functions that read data from a file, package it into a BAR App data structure and load it into the app. They are organized using the file and equipment type. The file type must match the file type 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 other 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.

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\_RAW.mlapp module it hoped to be sufficient. For most complex analyses there will be a corresponding Review Module that produces specific figures describing the analysis results.

# Future Work

As of December 12, 2022, the app will only function as hoped within the MATLAB environment. It was hoped that the code could be compiled into an executable that could dynamically use different functions and scripts. This has turned out to be impossible as those scripts need to be compiled at runtime, or a MATLAB server is used. Instead of working on this further the app has been developed as a MATLAB App that can be shared with other MATLAB users.

Future work will produce a quality assurance mechanism within the app. This will allow a user to flag suspect values for further review or exclusion from the results.

A sooner priority is to devise a way to create figures according to group information. This would better allow a user to observe differences between groups. This would be a precursor to offering statistics.

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

# 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

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 review\_RAW module and by supported 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. This will not create subfolders but is useful when quickly changing between Working Directories.

Before files can be loaded the app must know where to look for them. Directories can be entered into the search field2, or added through the button3. 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 on network drives.

Graphical user interface, text, application, email

Description automatically generated

### Working Database

The Working Database is expected to be treated as a project folder with specific subfolders. When the button is used to select one, the app will check that those subfolders exist. If they do not it will create them. Those folders include: Exports, Figures and Export. Exports contains csv files created by exporting data through the Export tab. Figures is used to save figures created at different locations within the app. Export is the target folder for csv files exported through the Export tab. A Working Database selected through the button will also be saved into the config file to be remembered the next time the app is opened.

Users can enter the Working Database through the text field. This will not create the subfolders or be remembered the next time the app is opened.

### Add Directory to Search

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

### File Extensions and Equipment Types

These are two dropdown lists that specific which files should be searched for. The file extension 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 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 and exclusion tags. 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 starts the search process to find the target files. The app will identify files with the file extension specified, that include the words of the Include text field and do not include any of the words in the Exclude 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 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.

|  |  |
| --- | --- |
| - 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 and stored at the object level. |
| - 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 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 stored the sampling frequency of the signals stored under object. If they have a sampling frequency they are assumed to all have the same. |
| - 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. |
| ... | Sublevels for ana 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

Move and Copy

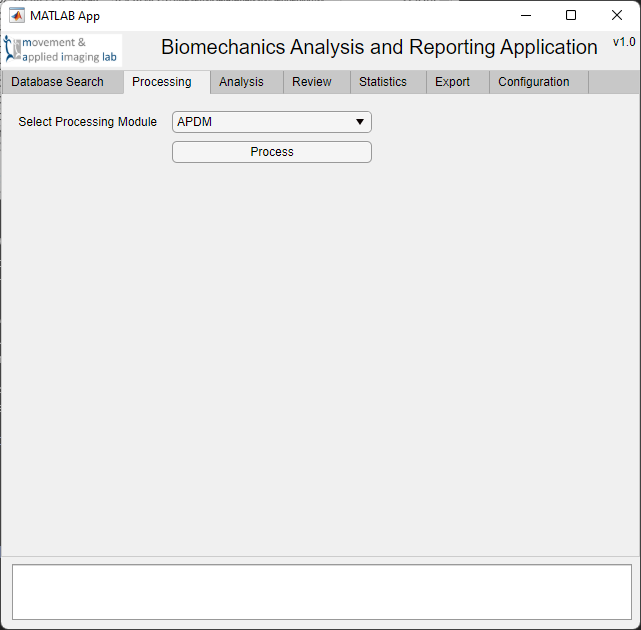
Identify Groups

Save Current Data

Table

Log

## Processing



## Analysis

## Review

## Export

## Configuration

# Process Modules

### Groupings

### Merge

### Segment

### Treatment

# 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.

## Custom

## 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.

### Assumptions

To run this code there may be some products that are required. These would be products needed to run the BAR\_App.mlapp file. Currently only MATLAB 9.11 is required to run the code. The App Designer would be needed to modify the code.

### 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.

### 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. The procedure 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 R2 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

## False Nearest Neighbor

## Recurrence Quantification Analysis

# Review Modules

### General

### Figures

# 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. |

# Appendix B Validation

## Main Application

### Database Search

Loading data

## Processing

### Groupings

### Merge

### Segment

## Analysis

### Quantitative Sensory Testing

### Time Lag

### Average Mutual Information

### False Nearest Neighbor

### Recurrence Quantification Analysis

## Review

## Export