Biomechanics Toolbox Documentation (v1.0.0)

Program and Help written by Walt Menke

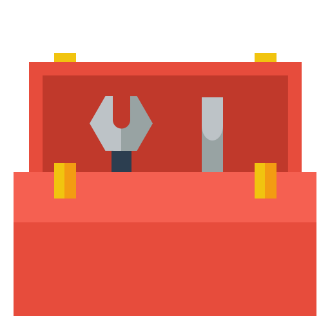


Table of Contents

[License 1](#_Toc162199928)

[*License Information* 1](#_Toc162199929)

[*License Agreement* 1](#_Toc162199930)

[Program Description 2](#_Toc162199931)

[*Purpose* 2](#_Toc162199932)

[*Hardware and Software* 2](#_Toc162199933)

[*Data Format Requirements* 2](#_Toc162199934)

[Package Requirements 3](#_Toc162199935)

[Main Menu 4](#_Toc162199936)

[Saving and Loading Tab Parameters 7](#_Toc162199937)

[Function - Script Gen 9](#_Toc162199938)

[Function - EMG (not available) 11](#_Toc162199939)

[Function - Batch 12](#_Toc162199940)

[Function - Normalize 15](#_Toc162199941)

[Function - Event Pick 16](#_Toc162199942)

[Function - Event Compile 19](#_Toc162199943)

[Function - Quality Check 22](#_Toc162199944)

[Function – Ensemble 24](#_Toc162199945)

[Function – SPM 27](#_Toc162199946)

[Example Files 33](#_Toc162199947)

[Suggested Citations 34](#_Toc162199948)

# License

## *License Information*

**The *Biomechanics Toolbox*** is free but copyright software, distributed under the terms of the GNU General Public License as published by the Free Software Foundation (version 3). Further details on “copyleft” can be found at <http://www.gnu.org/copyleft/>. In particular, ***Biomechanics Toolbox*** is supplied as is. No formal support or maintenance is provided or implied.

Additionally, this program utilizes the spm1d package which is covered under the same GNU General Public License 3.0 with Copyright (C) 2021-2022 Todd Pataky; further license information can be found at <https://spm1d.org/Downloads.html>.

## *License Agreement*

Copyright 2023-2024, Walter Menke

This file is part of the *Biomechanics Toolbox*.

*Biomechanics Toolbox* is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

*Biomechanics Toolbox* is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with *Biomechanics Toolbox*. If not, see <https://www.gnu.org/licenses/>.

# Program Description

## *Purpose*

The field of biomechanics is often plagued with time consuming data processing across multiple applications and scripts on account of the vast amount of data collected in labs (motion capture, kinetics, electromyography, etc.). Frequently, there are no packaged options for gathering data into usable batches, performing quality checks, determining useful discrete time series events, and producing publication ready figures. The purpose of the Biomechanics Toolbox is to facilitate a more efficient workflow for processing biomechanics data and preparing it for presentation in abstracts and manuscripts. Significant efforts have been made to provide numerous and descriptive error messages to the user to prevent frustration and time lost to troubleshooting. The program has been designed to allow users to generate Visual 3D scripts for each subject and condition, filter and process EMG data, batch process V3D output text files, normalize batch outputs for graphing, select only desired variables for discrete event picking, compile events, check data quality, create publication level ensemble graphs, and perform Statistical Parametric Mapping analyses.

## *Hardware and Software*

Package requirements for this program can be found in the “ToolboxRequirements.txt” file included with the application download. The matplotlib modification package “SciencePlots” is not strictly necessary as it requires a local LaTeX installation to be used. Plotting functionality will only be changed minimally without it installed; the effect will only be seen when generating ensemble curves. The program was originally written in Python v3.12.2 (64-bit) on Windows 11 in Visual Studio Code v1.84.2. The default application window size is 750x800 pixels, but some visualization functions create a window of 1100x800 pixels. The OS package in Python has been used to in effort to allow MacOS and Linux functionality, but functionality is not guaranteed outside of Windows 11. Current efforts are underway for testing within Linux.

## *Data Format Requirements*

This program was written with certain file naming conventions and data structures in mind. Specifically, it is highly encouraged to utilize a consistent naming scheme in the format of ‘S1C1T1,’ where S corresponds to the subject number, C corresponds to the condition (gait, cycling, stair climb, etc.), and T corresponds to the trial number. Once the Batch function has been used, for example, this program will recognize how many components (X, Y, Z) have been exported, but it is recommended to export all three components for this program to reduce the potential for backtracking. Given the ability for the user to select which specific variables to perform event picking on, there is no downside to having all three components present in the data. Be sure that single component variables (Joint Work, ROM, etc.) are *not* included in the files you would like to process with this program. Those variables should be in a separate file. Variable names should be named like “HipAngle” and “KneePower” because some functionality is tied to finding upper- and lowercase letters to insert whitespace.

It is strongly advised that you do *not* export two components for one variable and three components for another in the same output file. This is likely to result in abnormal program behavior. Additionally, script files using the V3D ‘Export\_Data\_to\_ASCII’ pipeline command should ideally export a file for each condition and side individually. For example, a file ‘S1C3\_Right\_Vars.txt” would be a good name to save all C3 data for S1. Combining multiple sides of conditions per output file is *not* advised as this may make your experience with later functions in the program more difficult, if not impossible. The Quality Check and Event Picking functions will become more cumbersome as a result. More specific formats and structures are explained along with relevant function sections.

# Package Requirements

Below is a list of packages and their respective versions that will work in this program with at least Python 3.12.2. Use the following line of code to easily install all required packages for the *Biomechanics Toolbox* using a file provided with the source code.

pip install -r ToolboxRequirements.txt

*Package == Version*

contourpy==1.2.0

cycler==0.12.1

fonttools==4.45.1

kiwisolver==1.4.5

matplotlib==3.8.2

numpy==1.26.2

packaging==23.2

pandas==2.1.3

Pillow==10.1.0

pyparsing==3.1.1

python-dateutil==2.8.2

pytz==2023.3.post1

SciencePlots==2.1.1

scipy==1.11.4

six==1.16.0

spm1d==0.4.22

ttkbootstrap==1.10.1

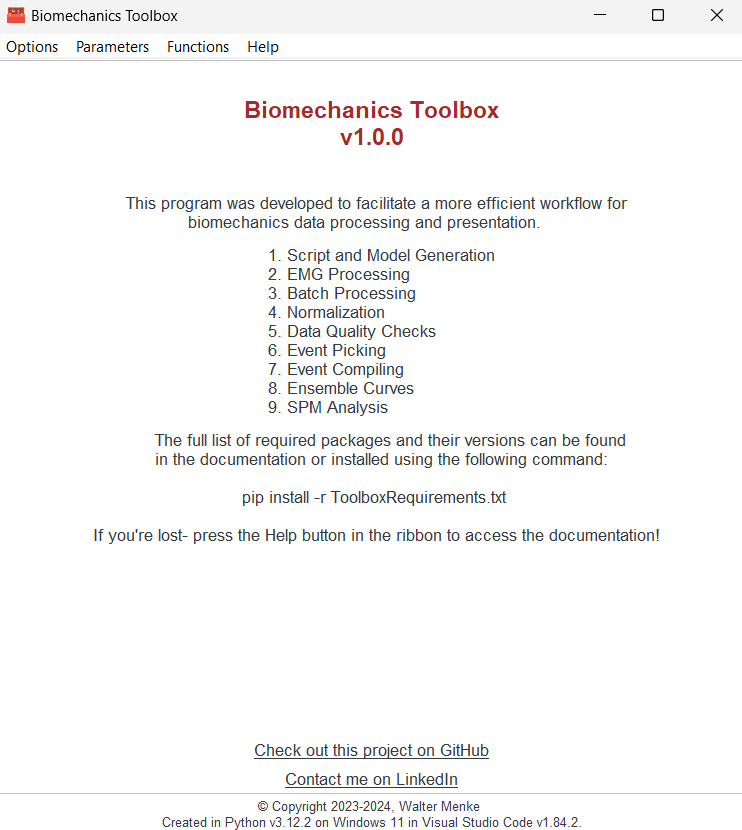
tzdata==2023.3

# Main Menu

The Main Menu of the program is shown below in Figure 1, and it consists of a small overview of the program’s purpose, functionality, and some required packages. The toolbar consists of ‘Options,’ ‘Parameters,’ ‘Functions,’ and ‘Help’ entries whose sub-entries are displayed in Figures 2, 3, 4, and 5, respectively.

Throughout the program, in general, input buttons will be ***green***, output buttons will be ***blue***, and function execute buttons will be ***orange***.

Figure 1:Main Menu image of the Biomechanics Toolbox with no tabs open.



The toolbar within the main window provides multiple dropdown menus: Options, Parameters, Functions, and Help (Figure 2,3, 4, and 5 respectively). The Options menu has six sub-entries that are explained here.

Reset Tab Entries: Resets all entry boxes, checkboxes, dropdowns, etc. for the current tab. With multiple tabs open, this will shift the selected tab to the end of the tab list.

Close Current Tab: Closes the currently selected tab.

Close All Tabs: Closes all open tabs and returns the user to the main menu.

Restart: Restarts the program.

Exit: Safely exits the program.

Figure 2: The Options toolbar entry expanded to show the sub-entries.

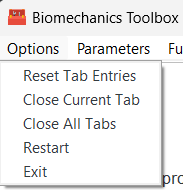
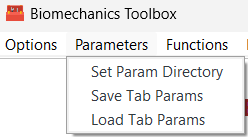


Figure 3: The Parameters toolbar entry expanded to show the sub-entries.



Set Param Directory: This option allows you to set a standard directory for your parameter text files. Parameters will be saved there by default and will be set as the default directory for when parameters are loaded in.

Save Tab Params: This option saves the currently selected tab’s entries and parameters to a txt file in a location specified by the user. This includes the tab name, strings, checkboxes, lists, etc.

Load Tab Params: This option loads saved tab parameters into the tab as specified in the Save Tab Params output file.

The Functions toolbar entry expanded to show the various functions available in the program are seen in Figure 3. The options are described in greater detail in the next section of the documentation.

Figure 4: The Functions toolbar entry expanded to show the sub-entries.

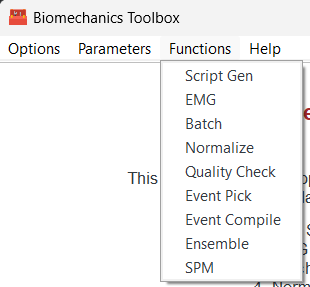
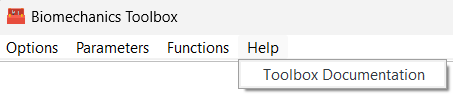


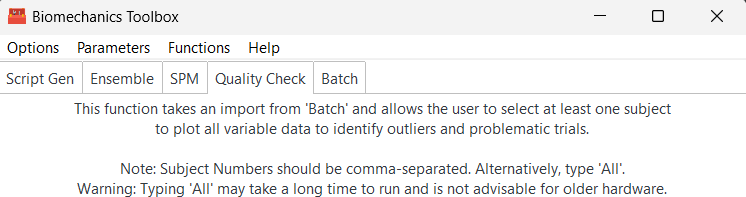
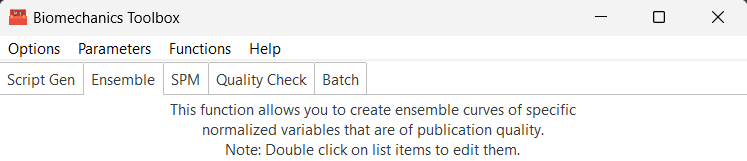
Figure 4 shows the Help toolbar entry expanded which has a single entry to open this documentation.

Figure 5: The Help toolbar entry expanded to show the Toolbox Documentation sub-entry.



The program supports the ability to have multiple tabs (each containing a function) open at one time and these tabs are clickable once opened (Figure 6).

Figure 6: Sequential images showing the selectable multi-tab functionality of the toolbox by selecting Ensemble or Quality Check.



# Saving and Loading Tab Parameters

Each tab has functionality to allow for saving of the selected tab’s entries, checkboxes, dropdowns, etc. This was designed to be used when the program is opened, and the user would like to resume a previous function with similar entries and selections. An example using the Batch tab is presented below in Figure 7-10. For improved functionality through sustained use of the program, you can also set a default Param Directory. This will be the default directory that is opened when loading or saving any parameters. It is, therefore, a clever idea to keep your parameter files in one location.

Figure 7:Saving the parameters of the Even Pick tab.

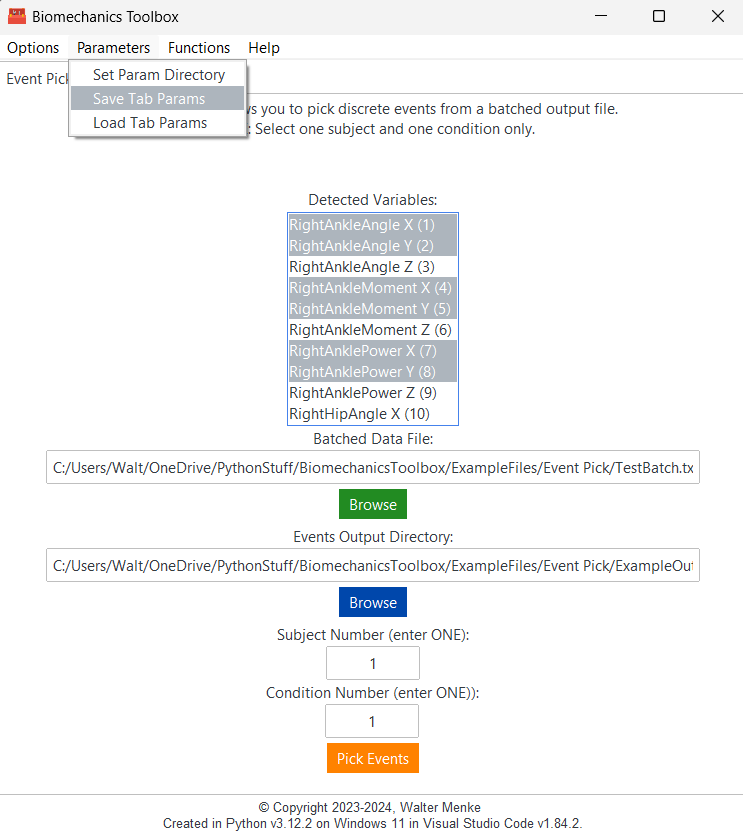


Figure 8: Save confirmation upon writing the parameter text file.

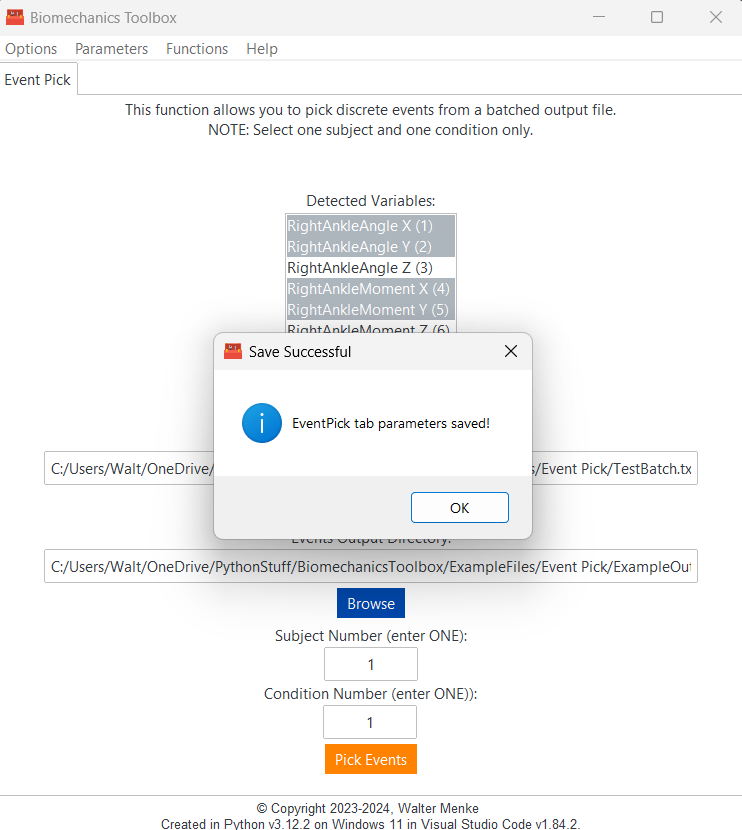


Figure 9: A snippet of the saved parameter file that can be loaded in.

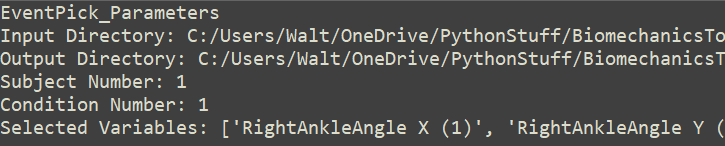
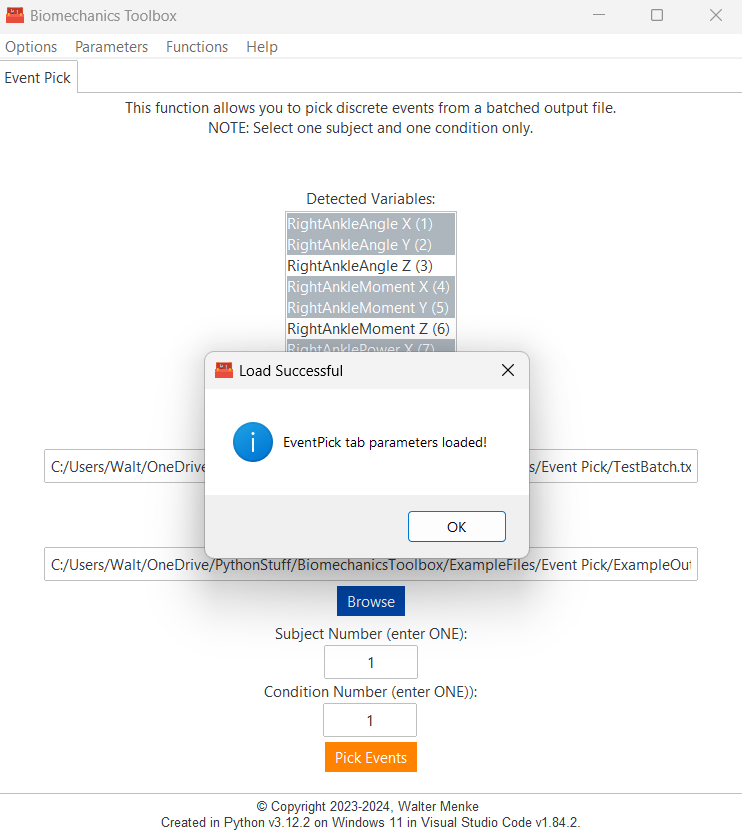


Figure 10: Loading in the saved Event Pick parameters from the blank tab.

A screenshot of a computer

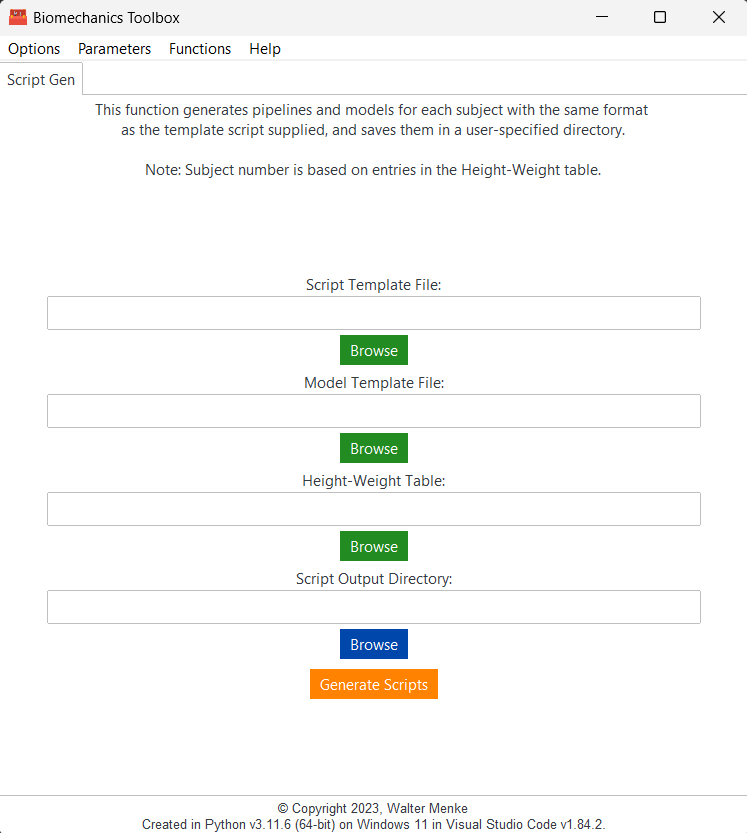
Description automatically generated

The following sections describe each of the available toolbox functions, their appearance, inputs required, and outputs produced for the functions. Descriptions of the variable type expected in each tab are also presented.

# Function - Script Gen

This function provides the ability to input a template V3D pipeline, V3D model file, and a Height-Weight table to produce scripts and models for all subjects with specific heights and weights applied. The intention of this function is to take an input script that processes *all* data for one subject and all conditions by naming them as S1C1, S1C2, etc within the script. This function will then iterate over the script and replace the subject number and model file based on the additional inputs below.

Figure 11: A view of the Script tab without entries.



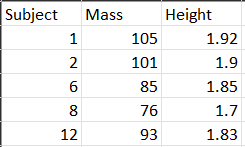
*Inputs*

Script Template File (string): Specify the full path to the Visual 3D pipeline file (.v3s) you would like to duplicate for multiple subject numbers. *Be sure that ‘S1’ is referenced throughout the template script as the program searches specifically for ‘S1’ to replace with your subject numbers.* This script should contain pipeline commands for ALL conditions of each subject so that you only have one script per subject.

Model Template File (string): Specify the full path to the Visual 3D model file (.mdh) you would like to duplicate for multiple subject numbers. As specified above, ‘S1’ should referenced within the template model file. The toolbox simply replaces the heights and weights between subjects by searching for /METRIC\_NAME=Mass (or Height) followed by /METRIC\_VALUE=(some float value).

Height-Weight Table (string): Specify the full path to an Excel file (.xlsx) that contains a layout as specified below (Figure 12). Mass should be entered in kilograms and Height should be entered in meters.

Figure 12: Example Height-Weight Table layout.



Script Output Directory (string): Specify the full path to the directory you would like the Scripts and Models to be placed. The program will generate a ‘Models’ and ‘Scripts’ folder in the directory if they do not already exist to store the respective files (Figure 13). You will need an entry for Mass and Height for each subject that is in the Subject column. Subject numbers without corresponding entries will result in an error.

*Outputs*

The function outputs can be seen below (Figure 14). The number of files generated depends on the length of the subject column in the Height-Weight table.

Figure 13: Models and Scripts folders generated by the toolbox.

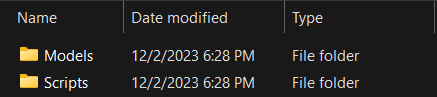
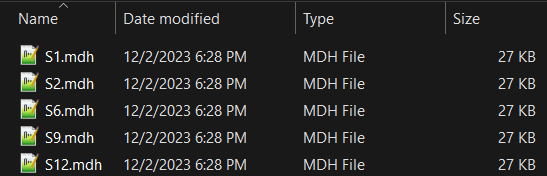
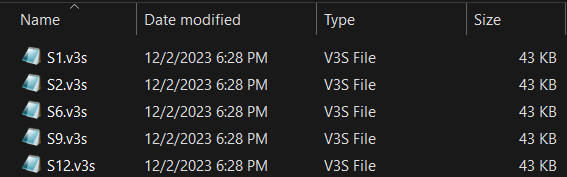


Figure 14: Contents of the Scripts and Models folders according to the Figure 7 Height-Weight table.



# Function - EMG (not available)

This function will allow for electromyography (EMG) data to be filtered, normalized, and optionally graphed if desired. Sensor numbers for muscles can also be changed within scripts from this function.

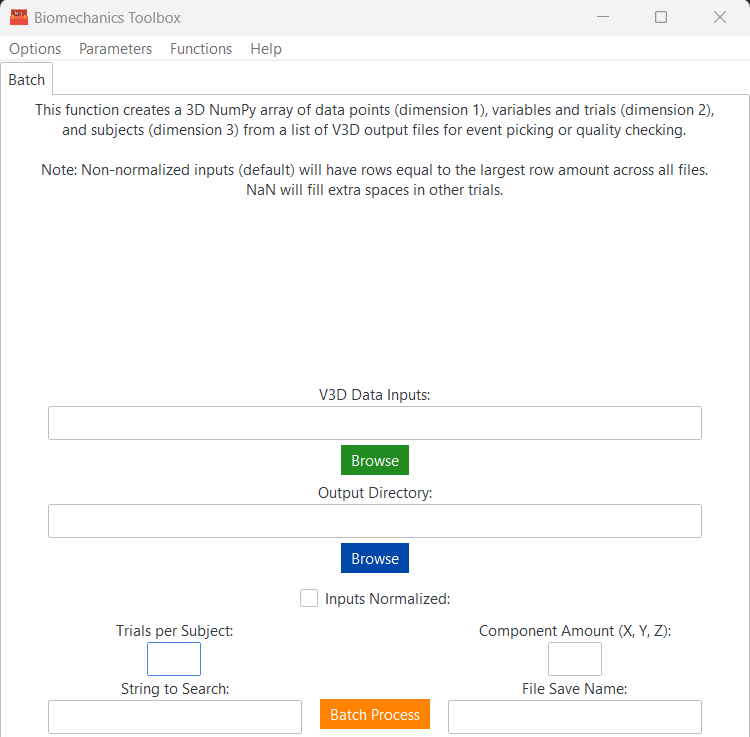
*Inputs, Outputs*

*Example*

# Function - Batch

This function creates a text file output that contains the compiled data for all subjects for a particular condition that is specified by the user.

Figure 15: A view of the Batch tab without entries.



*Inputs*

V3D Data Inputs (string): Specify the full path to the directory you have saved all V3D text output files of interest. All output files of interest for all subjects should be in this directory (Figure 17).

Output Directory (string): Specify the full path to the directory you would like your batched data to be saved to.

Inputs Normalized (True/False): A checkbox denoting if the input data are normalized to 101 data points.

Trials Per Subject (integer): A single integer that denotes the number of trials that are contained within each file in the V3D Data Inputs directory.

Component Amount (comma-separated list): A Boolean list describing the presence or absence of X, Y, and/or Z components. For example, to specify that X and Z components are present, you should enter ‘1,0,1.’ It is required to always export all components for this program.

String to Search (string): Specify a string that is a partial pattern match to select files from the folder you would like to compile. For example, if your folder contains files named as shown in Figure 17, the entry option here could be ‘\_walk.txt’ to select all related files. This string should be specific enough to exclude other files present if they are not currently of interest.

File Save Name (string): Specify the filename you would like the output to be saved to. Do not add a file extension, as the default extension is .TXT.

Figure 16: A view of the Batch tab with all entry boxes filled.

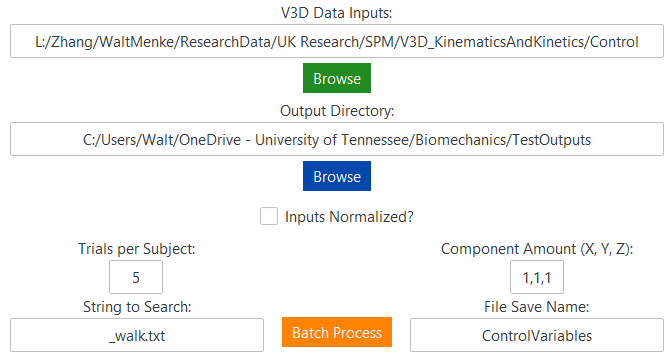
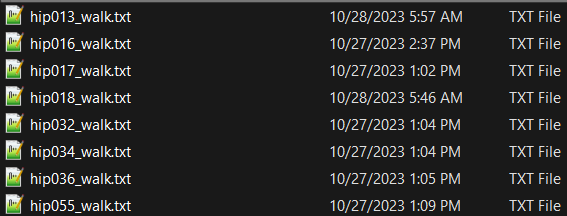


Figure 17: An example setup of a folder to search for the Batch function.



*Outputs*

The Batch function returns a text file with three lines of metadata followed by a flattened 3-dimensional array containing the observations (rows), trials and variables (columns), and subjects (slices). The first few lines of an output from this function can be seen in Figure 18.

Figure 18: An example output from the Batch function.

A screenshot of a computer

Description automatically generated

The first line (101 135 8) is metadata about the Rows X Columns X Slices of the original array and is used to reshape the array in later functions. The 101 rows correspond to the actual data points in your files. This example data has been normalized, and therefore has a row count of 101. For data that are not normalized, the row count will correspond to the longest row present in the data set and all intermediate values will be filled with NaN. The 135 columns correspond to the (Trials \* Variable Amount) for your dataset. The 8 slices indicate that there are 8 subjects contained within the dataset. Keep in mind that in other functions such as Quality Check and Event Pick, the subject number you input is mapped to this slice number and not your “actual” subject number. For example, if the 8 slices contain subjects 3,6,8,9,10,11,12, and 13 from your data- then doing a Quality Check on subject “1” in the later functions maps to your true subject 3.

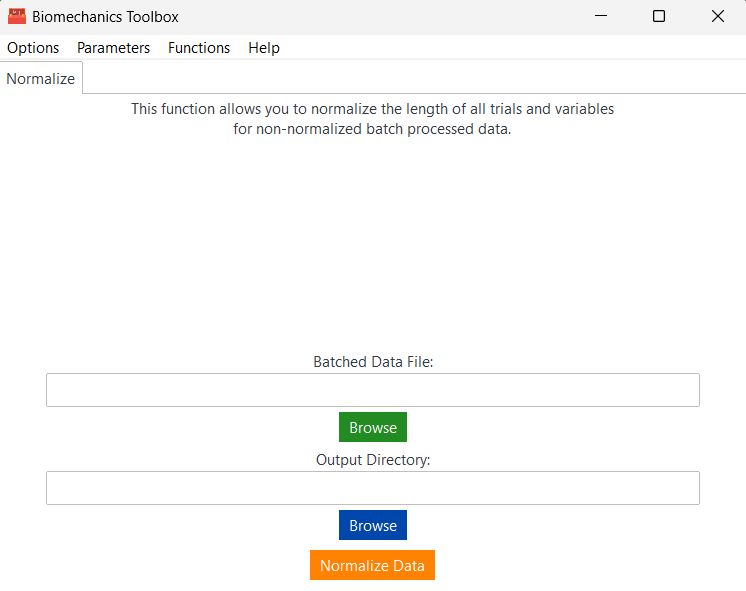
The second row in the Batch output file is a list of strings that are the variables present in all the files. The program checks to ensure each input file has the same number of columns. In Figure 18, ‘RightAnkleAngle’ is seen repeated three times. The purpose of this is to have a string for each component present in row three, which has (1,1,1), indicating that X, Y, and Z are present. If this entry was (1,0,1), then ‘RightAnkleAngle’ would only be repeated twice. In this example, the data were normalized before processing, but for non-normalized data the rows count will correspond to the largest number of rows for a trial across all subjects and ‘nan’ will fill all other empty spaces.

It is not recommended to batch more than one condition together. In Figure 18, we are batching all walk trials for the subjects. It is not advised to combine walking and running trials within one batch file for all subjects in this program.

# Function - Normalize

This function allows for the normalization of a file produced by the Batch function. In cases when normalized data are not exported from V3D, this should be used to allow for ensemble curve generation.

Figure 19: A view of the Normalize tab without entries.

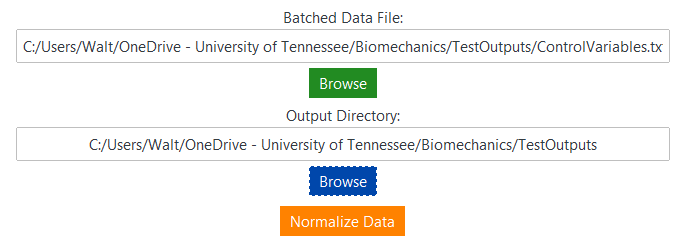


*Inputs*

Batched Data File (string): Specify the full path to a text file that was produced by the Batch function previously. Raw V3D outputs will not work here, the file must be produced by the Batch function.

Output Directory (string): Specify the full path to save the normalized text file. The filename will be the same as the input file with ‘\_Normalized” appended to the end.

Figure 20: A view of the Normalize tab with both entry boxes filled.



*Outputs*

A file with the same format as the output from the Batch function will be produced, except the row data will now be 101. Columns and slice amount will not be changed.

# Function - Event Pick

This function is designed to allow the user to assess, remove, and replace local maxima and minima from variables of interest. It allows the user to import a file produced by the Batch function (required) and select a single subject and single condition to process. This is the most complicated function available in the toolbox, and thus, likely to be the most prone to error. This function should not be used with normalized data.

One of the main benefits of this function is that it allows the user to load in a file with all their variables and define a subset of interest to pick events from (Figure 22). The advantage of this approach is that one file can be exported with all variables and all components instead of multiple output text files from Visual 3D. The parameter file will also remember your selections. You must select at least one variable to pick events from before proceeding to the next window (Figure 22). Use the left and right arrow keys to cycle between plots as an alternative to the buttons.

Figure 21: A view of the Event Picking tab without entries.

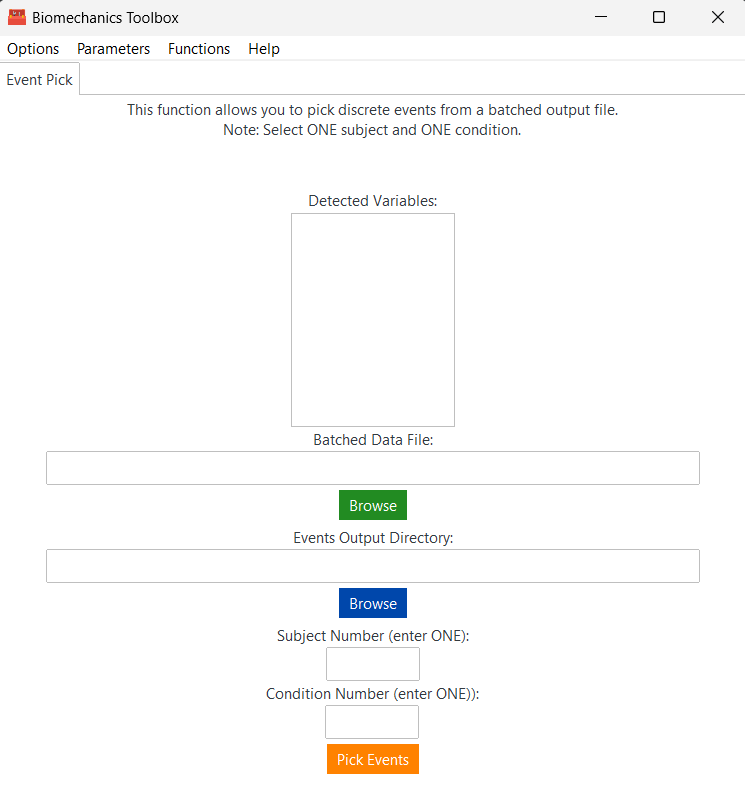


Figure 22: The Event Picking window with example entries present.

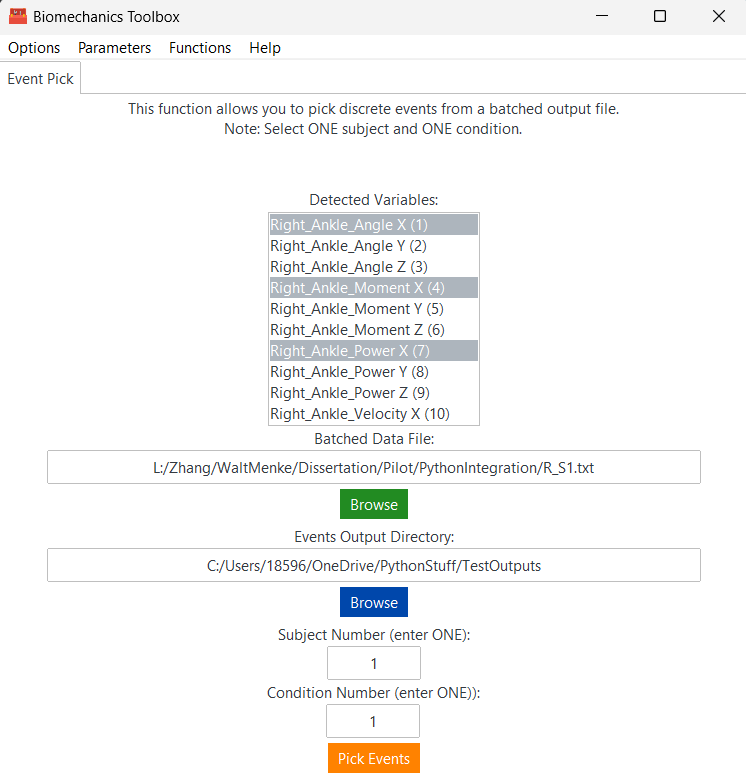
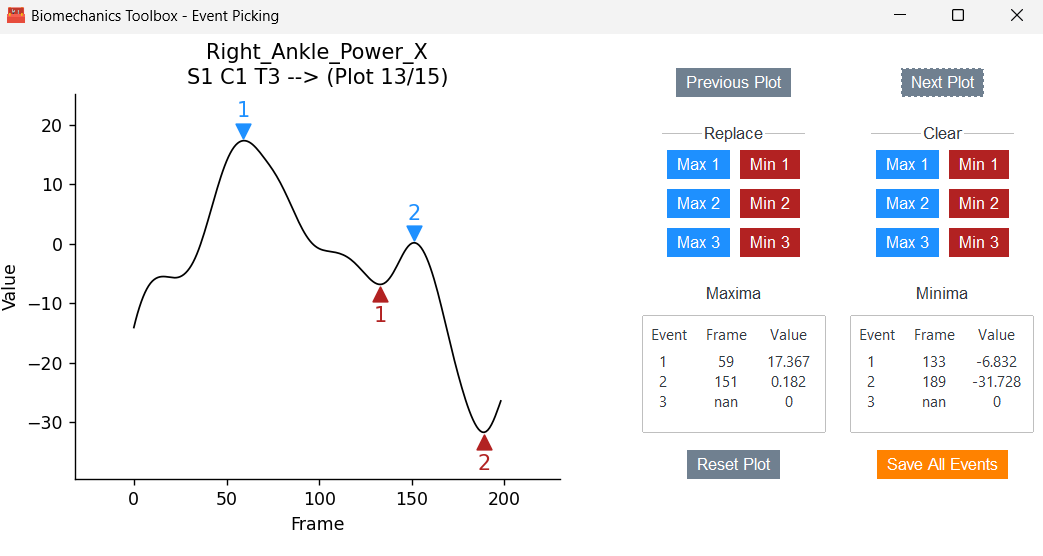


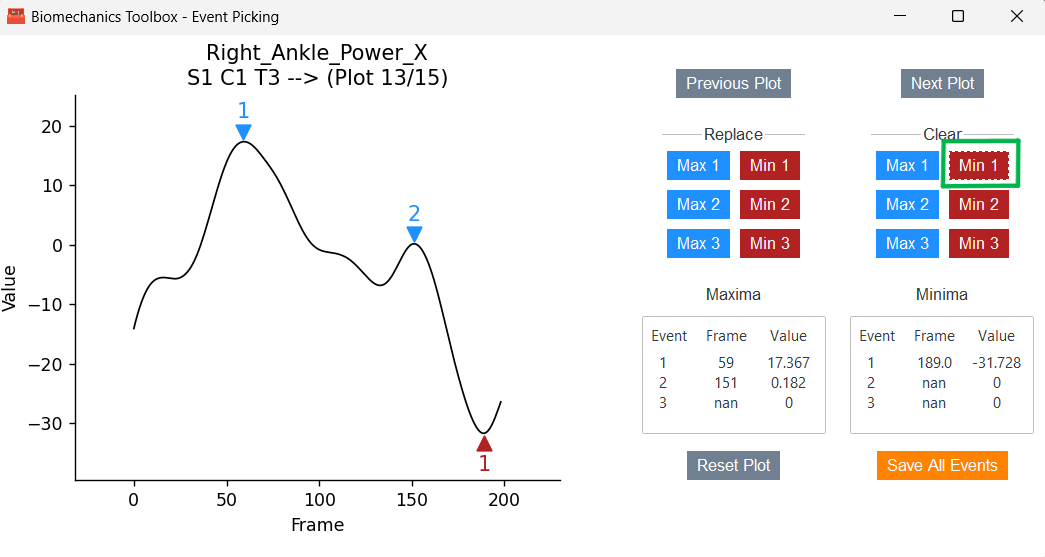
Figure 23: A view of the Event Picking window after clicking ‘Pick Events.’



*Clear Functionality*

One of the main functions of the button clusters on the right side of the window is to clear specific Max or Min values that are undesired. If the user is interested in naming maximum Ankle Power Max 1 at a certain point, the time series may generate a local maxima that is undesirable. In this case, the event can be cleared. Figure 24 shows when Min 1 is cleared, compared to Figure 23. The amount of minima changes, and Min 1 now takes the place of what used to be Min 2.

Figure 24: The Event Picking window after clicking ‘Clear’ on ‘Min 1,’ highlighted in green.



*Replace/Add Functionality*

The second, more powerful functionality is the Replace button cluster that allows the user to replace or add an event by clicking their mouse on the graph. By clicking Replace Min 1 (Figure 25), the user can then click anywhere inside the graph and the maxima will be placed within a 5-frame range closest to the clicked point and select the maximum (or minimum) value within that range. The user may also select Replace to add a second event, if only one max or min was found by default (Figure 26).

Figure 25: The Event Picking window after clicking 'Replace' on 'Min 1' and then clicking on the desired location.

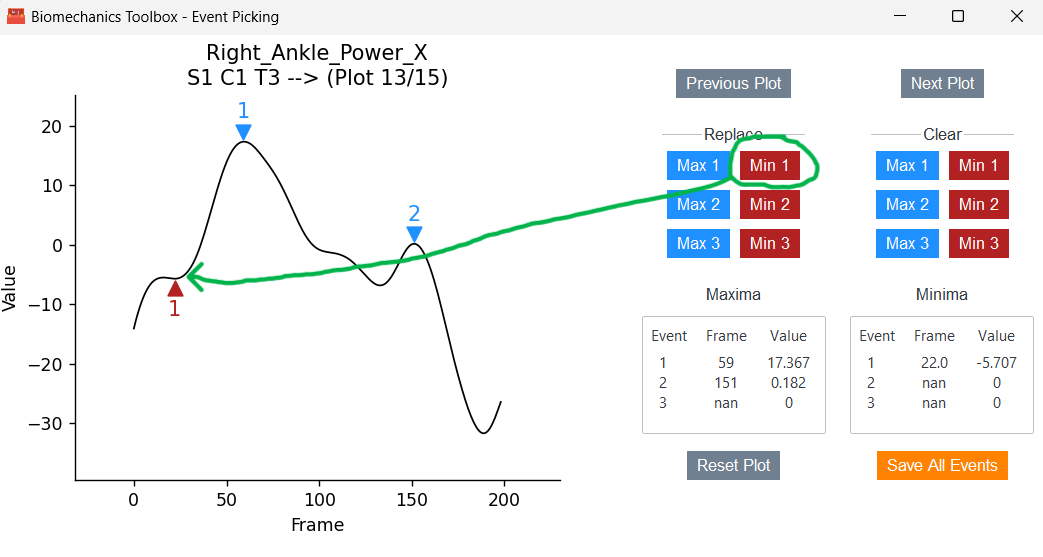
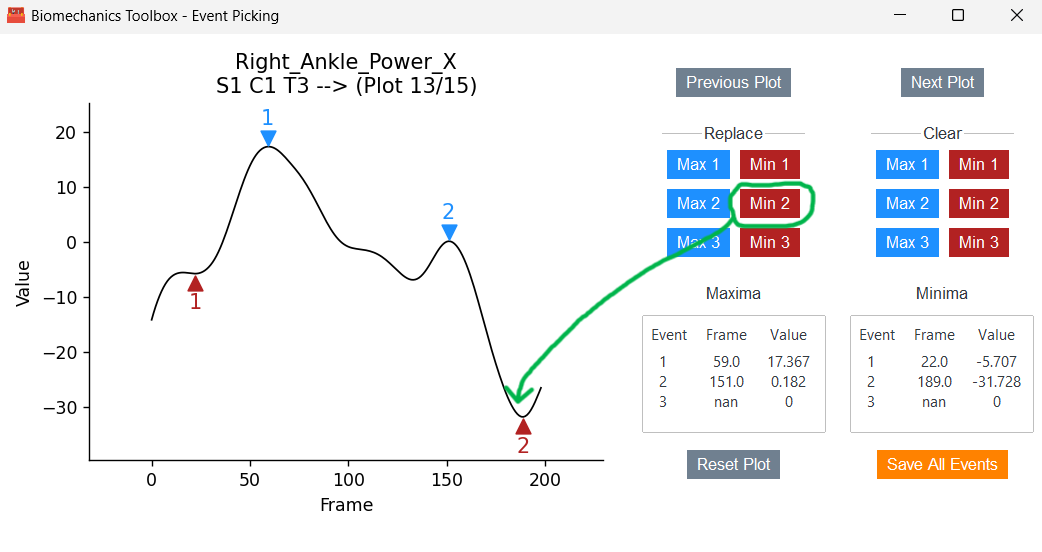


Figure 26: The Event Picking Window after selecting 'Replace' on 'Min 2' and clicking the desired location.



*Inputs*

Batched Data File (string): Specify the full path to a text file that was produced by the Batch function previously. Raw V3D outputs will not work here, the file must be produced by the Batch function.

Events Output Directory (string): Specify the full path to a directory for the exported Maxima and Minima files for the requested subject and condition.

Subject Number (integer): A single integer, not zero based, that determines the subject number you would like to select from the Batch input file. An input file with 5 subjects will throw an error if 6 is selected for this entry.

Condition Number (integer): A single integer, not zero based, that sets the condition number. This is defined by the user and will be used to generate filenames.

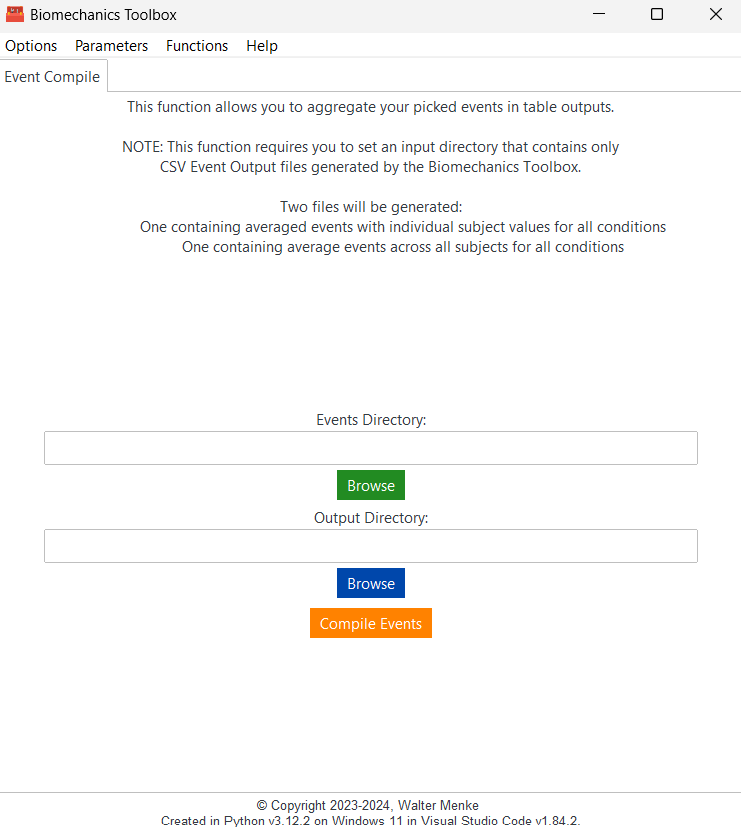
*Outputs*

Two files will be produced once “Save All Events” has been selected and successfully executed. Both have the prefix SX\_CY\_Minima.csv and SX\_CY\_Maxima.csv where X and Y are the specified subject and condition numbers, respectively. These output files contain metadata that are essential for input into the Event Compile function, so make sure no adjustments are made to those components.

# Function - Event Compile

This function is designed to allow the user to generate the mean and standard deviations of discrete events selected within the Event Pick function on a per-subject and per-condition basis. These can then be organized as inputs for statistical tests of significance, such as ANOVAs.

Figure 27: The blank Event Compile tab.



*Inputs*

Events Directory (string): Specify the full path to a directory containing *all* Event Pick files you are interested in aggregating. This should include Maxima and Minima output files for all subjects that you have picked events for and may also contain multiple, separately labeled conditions. For example, if you have Maxima and Minima for subject 1-8 for conditions 1-5, you should have a total of 80 files (8 subjects \* 2 files per condition \* 5 conditions) in the folder you set for this input. There is extensive logic for ensuring these files all contain the same events, but you should be using the Save and Load Parameters function to ensure consistency between event picking sessions. It is highly recommended to have no other files in this folder. The program checks for variable consistency within each condition only, not between conditions.

Output Directory (string): Specify the full path to a directory for the exported Excel files to be written.

*Outputs*

Two Excel files will be written by this function that contain the compiled event averages and standard deviations by subject (Figure 28), and for all subjects (Figure 30). If your input directory had multiple conditions, then these Excel files will have more than one Sheet- one for each condition.

Figure 28 shows an example output of the default name “Events\_By\_Subject.xlsx” file that is produced by the function. The SUBJECT column indicates which subject number the value comes from, as determined by the naming of the input files. The VARIABLE column indicates the variable from the file. These variables are listed in each Maxima and Minima file and are checked against each other to ensure the same variables are in all files. The TYPE column identifies which “type” of event value each row is. The options are Value, Index, and Per\_Loc and correspond to the actual time series value, frame location of that value, and index location as a percent of the whole trial. The NUMBER column corresponds to Max/Min 1, Max/Min 2, Max/Min 3.

Figure 28: Example output of the Event Compile function for assessing subject values.

A table of numbers and letters

Description automatically generated

Figure 29 shows an example of a set of files that would compile events for 3 subjects across 2 conditions. Files must be named in the style as shown and the program will allow at most a two-digit number after ‘S’ and ‘C.’ This limits the subject and condition number to ninety-nine each.

Figure 29: The input files for three subjects and 2 conditions, properly named.

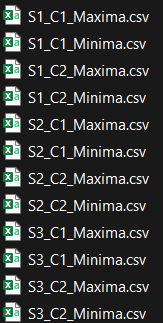


Figure 25 shows an example of the default name “Events\_Synthesized.xlsx” file that is produced by the function. These values are the average and standard deviations for a given variable and type across all subjects that were used as inputs.

Figure 30: Example output of the Event Compile function averaging all subjects for a given condition.

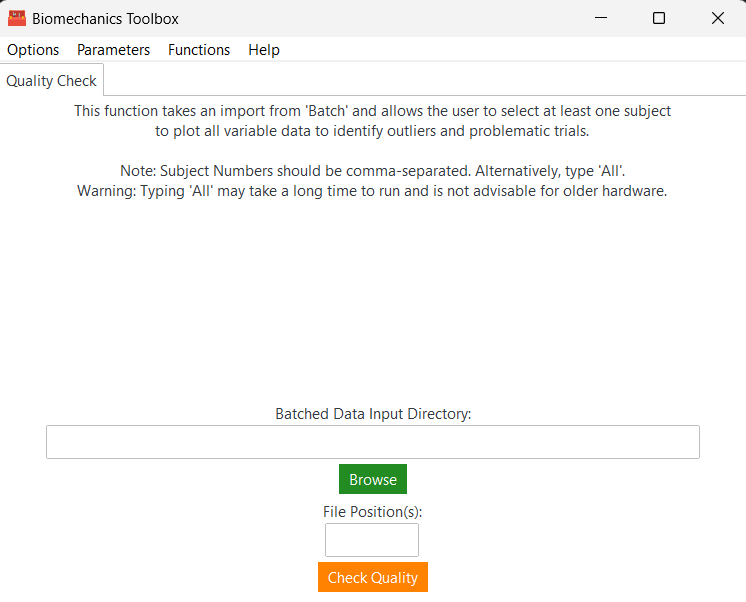
A screenshot of a data table

Description automatically generated

# Function - Quality Check

This function allows the user to assess the visual quality of a Batch function output file for outliers, marker label issues, or errors in processing among trials of the same variables. One or more subjects at a time can be investigated for a single condition.

Figure 31: A view of the Quality Check tab without entries.



*Inputs*

Batched Data Input Directory (string): Specify the full path to a file produced by the Batch function. This should ideally be the variables for a single condition.

File Position(s) (comma-separated list): Specify the number, as a position, of the file(s) you would like to assess separated by commas without whitespaces in between. Alternatively, you may enter ‘All’ if you would like to produce the quality check graphs for each file in the input file. This may strain systems with poor memory capacities. For example, if a Batch output file named ‘S1\_Right’ contains 5 different input files (each representing a condition), then typing ‘1,2,3’ would perform a quality check of the first three conditions. The values you place here have meaning *only* based on how your Batch output file was constructed.

Figure 32: A view of the Quality Check tab with both entries.

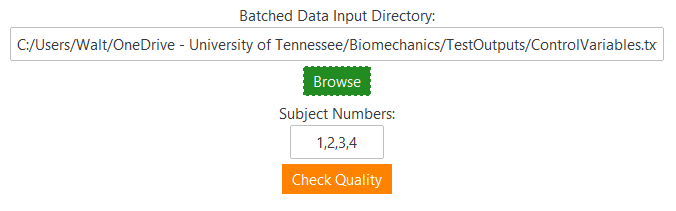


Figure 33 shows the Quality Checking window that appears after ‘Check Quality’ successfully executes. Please note that if you type ‘All’ for the Subject Numbers entry it will take some time to produce the plots. In the bottom right and left corners of the window the Next and Previous buttons will progress you through the quality checks. The Subject number is displayed in the title as the order in which the subject is in the file. For example, if the input file has true subject numbers 3,4,6,8 then this window will display Subject Number 1 for your true subject number 3. This function produces 3x3 plots of the variables in the input file and the Page number is displayed in the title. A file with 27 variables will have three pages per subject (9 variables X 3 pages).

Figure 33:A view of the Quality Checking window with sample data.

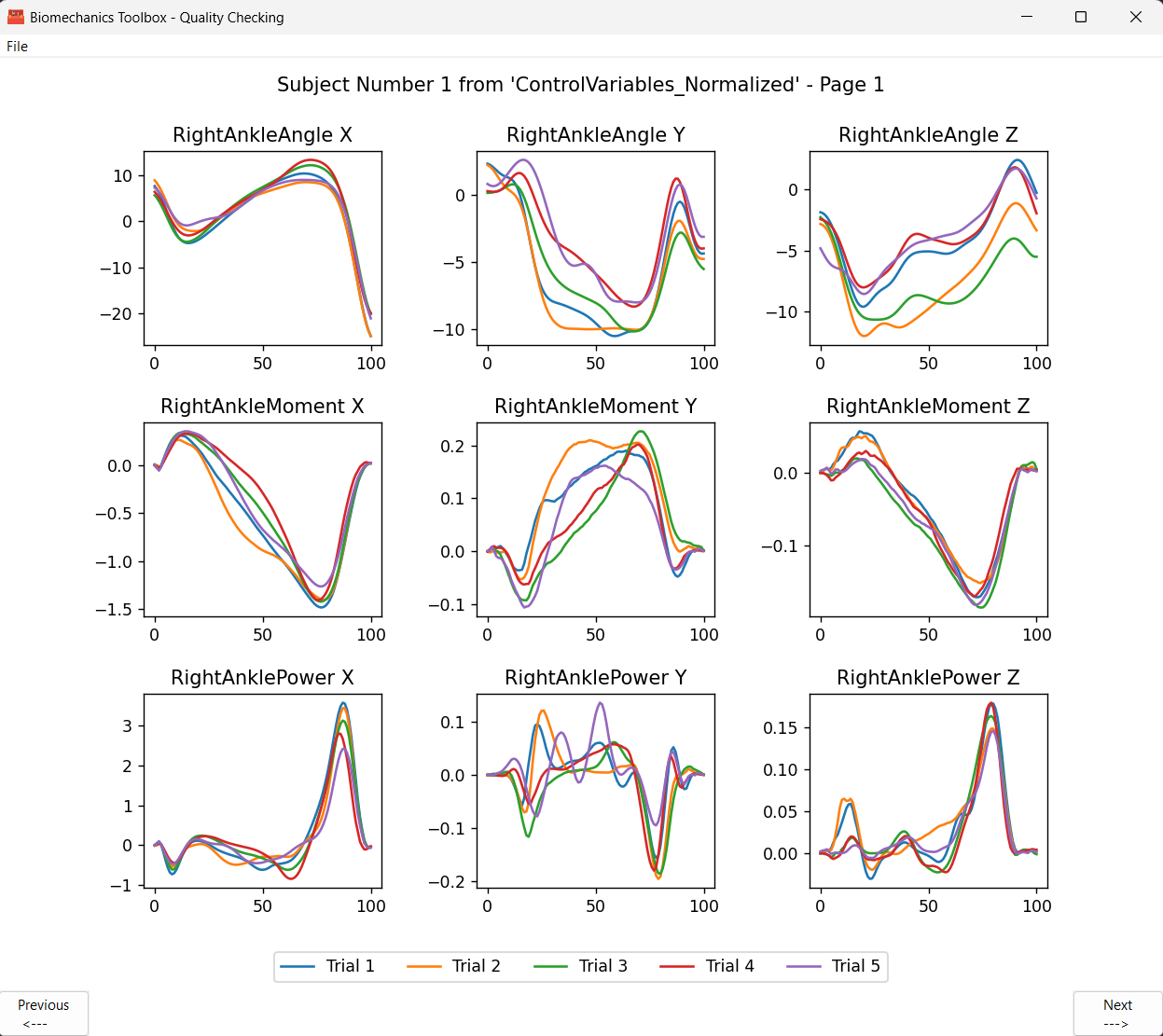
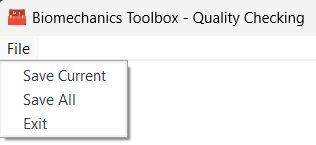


Figure 34 shows the options for saving quality check plots. ‘Save Current’ will save the current plot as one of the following filetypes: PDF, JPG, PNG, or .TIF. ‘Save All’ will save every quality check plot produced into a single PDF file.

Figure 34: The Quality Check toolbar with sub-entries options.



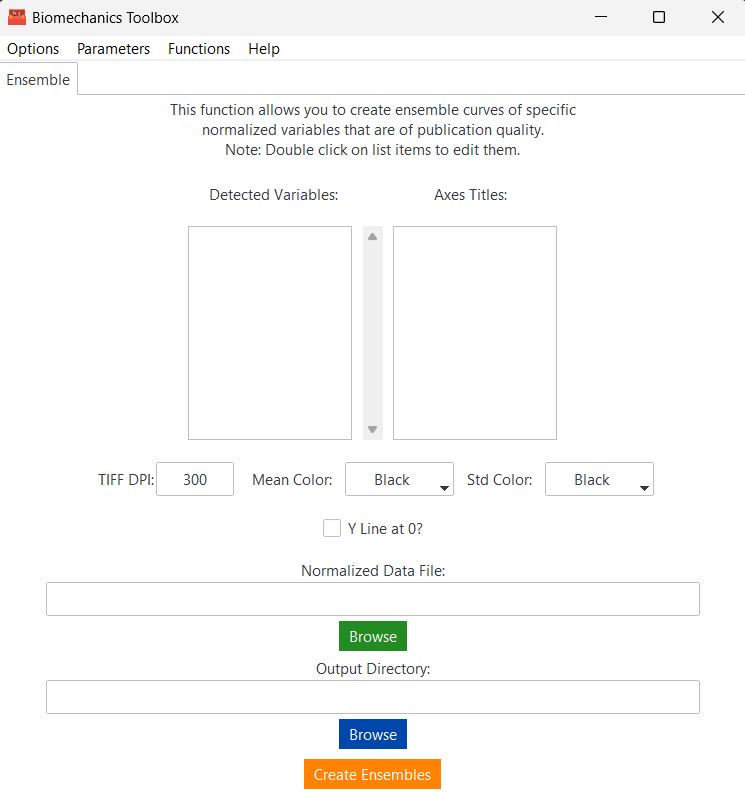
*Outputs*

No outputs are automatically generated for the Quality Check function. As stated above, images of individual or all the quality check plots can be optionally saved.

# Function – Ensemble

This function takes a user specified Batch function output that is normalized and produces publication quality ensemble plot(s) of one group with user specified axes, colors, and .TIFF dots per inch (DPI) specifications.

Figure 35: The blank Ensemble tab with no entries.



*Inputs*

Normalized Data File (string): Specify the full path to a normalized file of data that follows the format of the Batch function output. This function does not accept non-normalized data.

Output Directory (string): Specify the full path to the output ensemble image(s).

Detected Variables: This is automatically generated from the input file once it has been selected. It displays each detected variable and allows the user to double-click on an entry to change the variable name to their liking (Figure 37). This entry will be the ensemble plot title. The variables you would like to plot *must* be selected before pressing the ‘Create Ensembles’ button (Figure 38).

Axes Titles: This is automatically generated according to the length of the Detected Variables. The order of the variables is numbered in parentheses so the user can match variables with their respective axes. Double-clicking on the axes titles allows the user to specify the X and Y axis titles, separated by a comma. The axes do not need to be selected to create ensemble plots.

TIFF DPI (integer): Specify an integer for the dots per inch you would like the ensemble plot(s) to be saved at. The default value is 300 as this is frequently the minimum accepted by journals.

Mean Color: The desired mean line color for the ensemble plots. It is recommended to leave it black.

Std Color: The desired standard deviation shading color for the ensemble plots. It is recommended to leave it as black because this will become opaque (alpha = 0.2).

Y Line at 0: A checkbox to specify if you would like a black, dotted reference line at y = 0 across the ensemble plots. It is important to note that this should only be checked if you are sure that y = 0 is within the plot bounds.

Figure 36: A view of the Ensemble tab with entries filled.

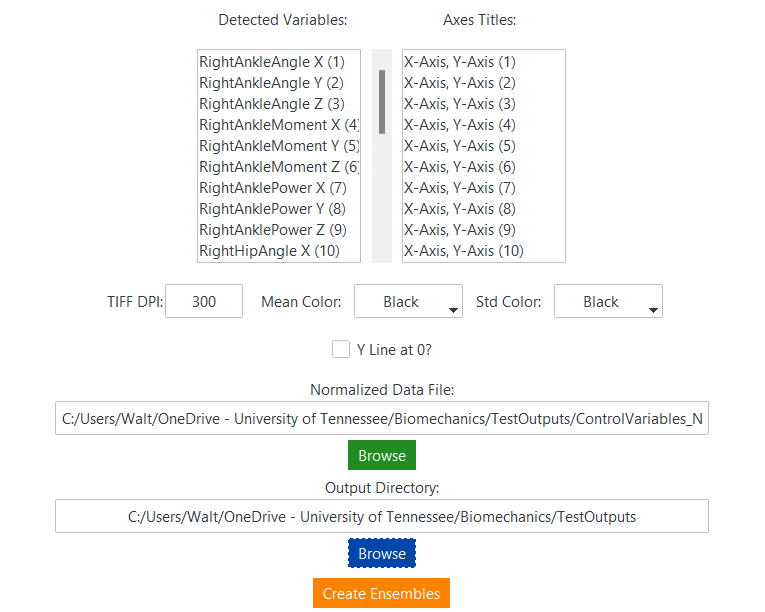


Figure 37: Result of double clicking on a variable entry in the list box for renaming.

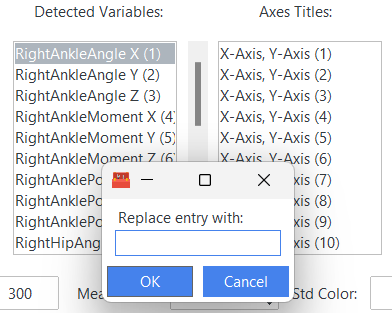


Figure 38: Example usage of multiple variable selection after renaming variables and axes.

A screenshot of a computer

Description automatically generated

*Outputs*

Figure 23 displays two separate ensemble plots that were generated from the options as shown in Figure 22. The user defined the X axis label, Y axis label, and the plot title. The legend entries are Mean and Std Dev by default. By default, the matplotlib package is used for plot settings. However, with an accompanying installation of LaTeX, the [SciencePlots package by John Garrett](https://github.com/garrettj403/SciencePlots) can be used to produce higher quality plot appearances.

Figure 39: Example outputs of an ensemble curve from Figure 38.

A graph of a normal body

Description automatically generated with medium confidence

# Function – SPM

This function currently has partial functionality for 2 groups only. Beware!

This function allows for multiple types of statistical parametric mapping (SPM) analyses to be carried out using the [SPM1D package created by Todd Pataky](https://spm1d.org/index.html). Testing options for 1, 2, and 3 groups are currently selectable (Figures 43, 44, and 45, respectively). It is important to note that data inputs for this function *must* be normalized to the same length. It is recommended to thoroughly read the SPM1D documentation for assistance with interpreting results.

SPM should not be conducted without some prior statistical knowledge. Important considerations that should be accounted for before conducting these tests are normality assumptions and sphericity. Nonparametric tests should be used if your data violate the normality assumption. Included below is a brief overview of the SPM tests made available in this program.

One-Sample T Test: One group test. This test determines if the mean of a sample is significantly different from a known or hypothesized population mean.

Paired T Test: Two group test. This test compares two *dependent* groups to determine if there is a significant difference between them. Data sets should come from the same subject pool as opposed to two independent groups.

Two-Sample T Test: Two group test. This test compares two *independent* groups to determine if there is a significant difference between them.

One-Way ANOVA: Three group test. This test is used to compare three or more *independent* groups to determine if there is a significant difference between them. This program currently only allows for three groups to be compared for this test.

One-Way Repeated Measures ANOVA: Three group test. This test is used to compare three or more *dependent* groups to determine if there is a significant difference between them. This program currently only allows for three groups to be compared for this test.

Figure 40: A view of the SPM tab without selecting a Group number.

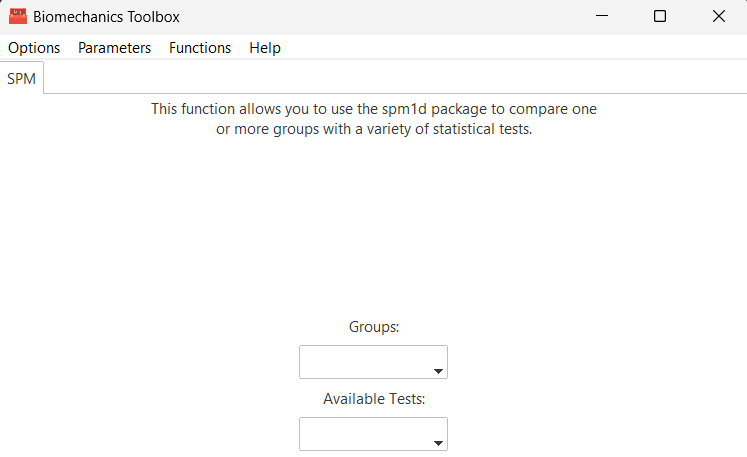


Figure 41: A view of the SPM tab with Group set to 2 and the resulting entry and button generation.

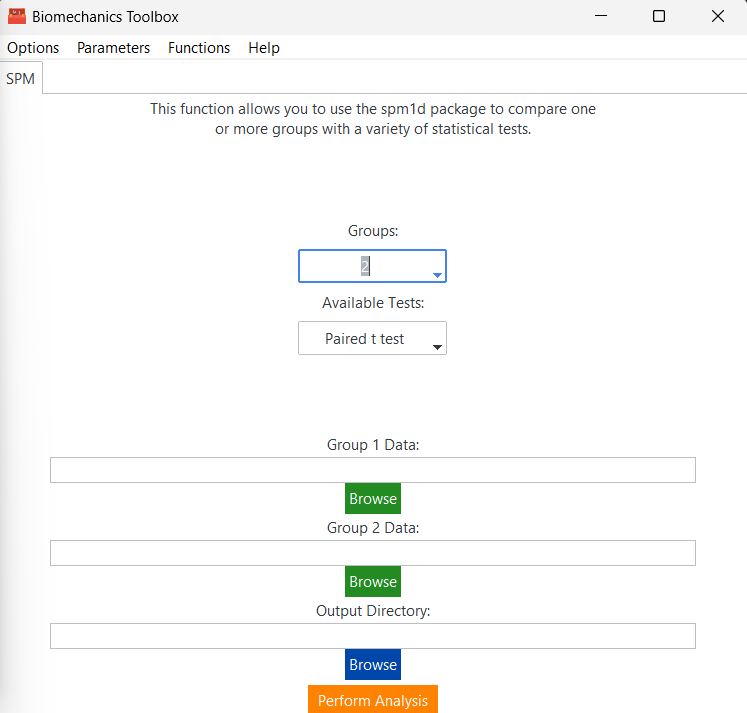


Figure 42: Group selection options from 1-3.

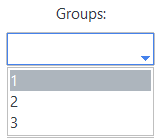


Figure 43: The test option when 1 Group is selected.

A screenshot of a test

Description automatically generated

Figure 44: The test options when 2 Groups are selected.

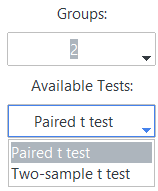


Figure 45: The test options when 3 Groups are selected.

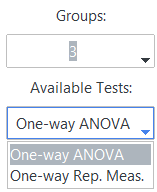


Figure 46: The child window of the SPM tab when two Groups have been selected.

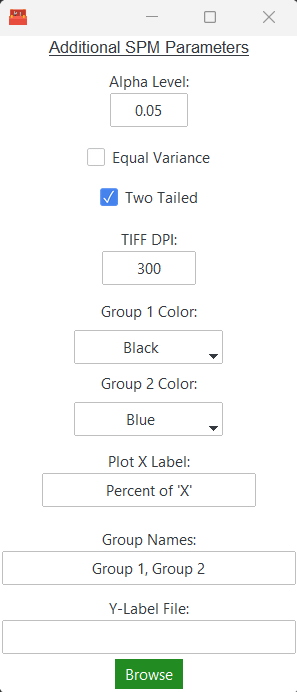
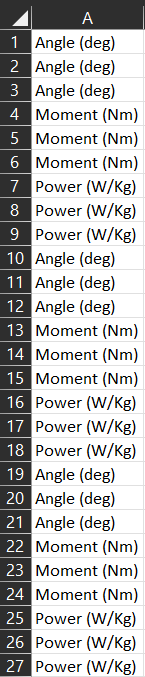


Figure 47: Format of the expected 'Y-Label File' for SPM plots when the data has 27 variables.



*Inputs*

Groups: Select an option from 1 to 3 corresponding to the group amount you would like to compare.

Available Tests: Select an option from the available tests. These options will change depending on the number of groups selected (Figures 43, 44, and 45).

Group ‘X’ Data: Specify the full path to the Batch function output file for group number “X.” The number of generated entries for groups will adjust based on Group selection.

Alpha Level: The desired significance level for SPM analyses.

Equal Variance: Whether or not to assume equal variances in the input data. Default is False (unchecked), equal variance is *not* assumed.

Two Tailed: Whether or not to conduct a two tailed SPM analysis. Default is True (checked).

TIFF DPI: Specify an integer for the dots per inch you would like the ensemble plot(s) to be saved to. The default value is 300 as this is frequently the minimum accepted by journals.

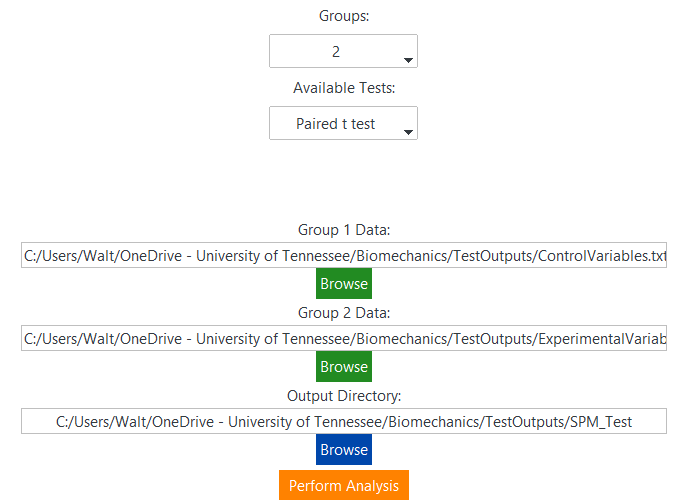
Group ‘X’ Color: Select the color for the mean line and shaded standard deviation area for each group number “X.” The number of generated entries for groups will adjust based on Group selection.

Plot X Label: Specify a plot X label to set for all SPM outputs. SPM data must be the same length and therefore, ‘Percent of X’ is the default entry.

Group Names: Specify a comma separated list of names for each group present in the inputs. For two group tests, an acceptable entry would be “Control, Intervention.”

Y-Label File: Specify the full path to an Excel file (.xlsx) that contains the plot y-labels for each variable in the input data (Figure 47). Be sure to have a single entry for each variable.

Figure 48: A view of the filled entries for a 2 Group paired t-test SPM analysis.



*Outputs*

The SPM function will automatically conduct the chosen SPM test for all variables in the data input file(s). A .TIFF file will be generated for each variable in the file and a PDF of all SPM outputs will additionally be generated.

Figure 49: The SPM save dialog after successful execution.

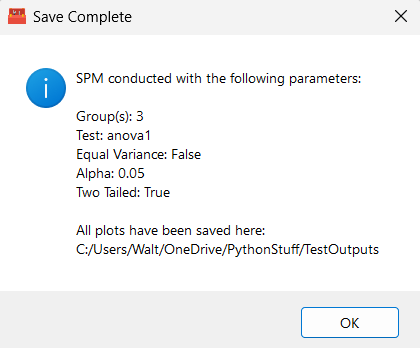


Figure 50: An example output from the entries in Figure 23.

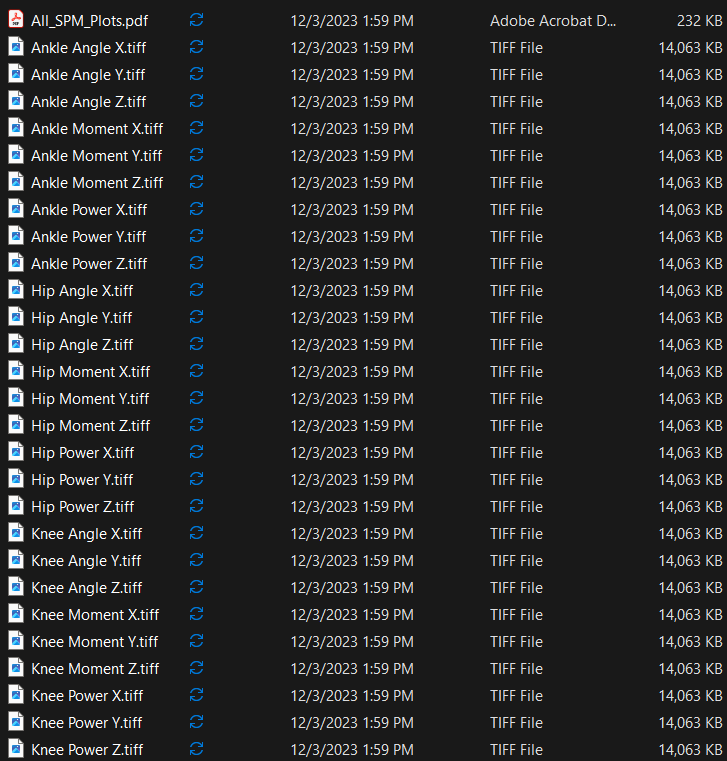


Figure 51: An individual SPM analysis graph (Knee Moment Z.tiff) produced by the SPM tab.



# Example Files

Several files have been included in the GitHub repository under the “ExampleFiles” folder. These contain subfolders with files that can be used as inputs for specific functions so you can see how the function takes inputs, how to Save and Load parameters, and get a better understanding of the file structure requirements that some functions have. Additionally, example outputs have been provided that result from using the given inputs in the functions so you can test the program yourself.

Currently, the following functions have working example setups:

Batch

Ensemble

Event Pick

Event Compile

Quality Check

Script Gen

SPM

# Suggested Citations

You are not under any obligation to cite software/code used within the *Biomechanics Toolbox* or the program itself. However, this section provides a handful of recommended citations if you would like to use them.

*SPM1D*

Pataky TC. One-dimensional statistical parametric mapping in Python. *Comput Methods Biomech Biomed Engin*. 2012;15(3):295-301. doi:10.1080/10255842.2010.527837

*SciencePlots*

John Garrett, Echedey Luis, H.-H. Peng, Tim Cera, gobinathj, Josh Borrow, Mehmet Keçeci, et al. “Garrettj403/scienceplots: 2.1.1”. Zenodo, November 25, 2023. <https://doi.org/10.5281/zenodo.10206719>.

*Biomechanics Toolbox*

*(TBD)*