MTQuant++ Documentation

Contents

1 MTQuant++ Overview

MTQuant++ builds on the original MTQuant by providing functions for the following:

1. Calculating statistical significance of different organization parameters
2. Splitting the axon and calculating organization parameters for each split
3. Microtubule Simulations

2 Setup & System Requirements

Download both MTQuant and MTQuant++ from their respective websites:

- MTQuant: http://roscoope.github.io/MTQuant
- MTQuant++: http://roscoope.github.io/MTQuant++

The code can be downloaded as a ZIP file or checked out as a Github repository. Download all of the files and keep them together. The run the following code to add all of the subdirectories to the path:

```
>> addpath('C:/path/to/MTQuant')
>> addpath('C:/path/to/MTQuant/em')
>> addpath('C:/path/to/MTQuant/misc')
>> addpath('C:/path/to/MTQuant/neuronTracing')
>> addpath('C:/path/to/MTQuant/redScan')
>> addpath('C:/path/to/MTQuant/singleMT')
>> addpath('C:/path/to/MTQuant++')
>> addpath('C:/path/to/MTQuant/misc')
>> addpath('C:/path/to/MTQuant/simulations')
>> addpath('C:/path/to/MTQuant/splitNeuron')
>> addpath('C:/path/to/MTQuant/stats')
```

The setup.m file contains these commands assuming that you are in the MTQuant++ directory, and that the MTQuant and MTQuant++ directories are in the same parent directory. Modify setup.m as necessary with the correct path names for easy startup.

If you copy and paste commands from this document into MATLAB directly, some characters, such as curly braces ({}) might not copy correctly; be sure to review copied commands before running them.

Thus far, MTQuant has been tested on Windows 7 and 8, using MATLAB versions R2014b and R2013b.

3 Calculate Statistics

There are two functions that take a CSV written by `MTQuant` and calculate the statistical significance of the organization parameter relationships. One is used for rank correlation (e.g., how a parameter changes as a function of age), and the other is used to compare various groups to wild-type.

3.1 `compareStatsAge`

This function calculates the correlation and p-value using the Spearman Rank Correlation.

3.1.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
`statsFile`	the name of an output file generated by `MTQuant`.
`groups` (optional)	a list of group numbers to compare. These are the values of the `DirNum` column of the output file (see Section 4 of the `MTQuant` documentation for more information). If `groups` is not inputted, or it is empty (i.e. `groups=[];`), the function compares all groups.
`stats` (optional)	list of columns in `statsFile` for which to calculate the correlation. This can be a list of column numbers or a list of strings, where each string is the column name (See Section 5.2 of `MTQuant` documentation for complete list). By default, `stats = {'Avg_Spacing','Std_Dev_Spacing','Avg_Coverage','Avg_Length'};`
OUTPUT ARGUMENTS	
`allRhos`	cell array with Spearman Rank Correlation for parameters in `stats`
`allPs`	cell array containing the p-values associated with `allRhos`

3.1.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2.csv'
>> groups = [1;2;5;6]
>> [allRhos,allPs]= compareStatsAge(statsFile,groups,stats)
allRhos =
    'Avg_Spacing''Std_Dev_Spacing''Avg_Coverage''Avg_Length'
    [ -0.1156] [ -0.1657] [ 0.7217] [ 0.7556]
allPs =
    'Avg_Spacing''Std_Dev_Spacing''Avg_Coverage''Avg_Length'
    [ 0.0709] [ 0.0094] [ 1.0424e-40] [ 1.5150e-46]
```

3.2 compareStatsToWT

This function calculates the two-sample t-test between every group of animals and wild-type.

3.2.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
`statsFile`	the name of an output file generated by `MTQuant`
`groups` (optional)	a list of group numbers to compare. These are the values of the `DirNum` column of the output file (see Section 4 of the `MTQuant` documentation for more information). If `groups` is not inputted, or it is empty (i.e. `groups=[];`), the function compares all groups.
`wtGroup` (optional)	The group number for the wild-type animals. If this input is not included, `wtGroup` is set to be the smallest value of `DirNum` in `statsFile`
`stats` (optional)	list of columns in `statsFile` for which to calculate the correlation. This can be a list of column numbers or a list of strings, where each string is the column name (See Section 5.2 of `MTQuant` documentation for complete list)
OUTPUT ARGUMENTS	
`allPs`	cell array containing the p-values of two-way t-tests for each parameter in `stats`. `N` is the number of animals in each group
`allHs`	cell array containing the decision to reject the null hypothesis (1 for p-value less than 0.05)
`allMeans`	cell array containing the mean values of each parameter for each group
`allStds`	cell array containing the standard deviation of each parameter for each group

3.2.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2.csv';
>> groups = [3;4];
>> wtGroup = 2;
>> stats = {'Avg_Spacing';'Avg_Coverage';'Avg_Length'};
>> [allPs,allHs,allMeans,allStds] = compareStatsToWT(statsFile,groups,wtGroup,stats)

allPs =
    [] 'N''Avg_Spacing''Avg_Coverage''Avg_Length'
    'L4 17-4'[45] [ 0.7808] [ 0.2911] [ 0.8126]
    'L4 arl-8'[41] [ 0.2968] [ 0.1671] [ 0.0194]
allHs =
    [] 'N''Avg_Spacing''Avg_Coverage''Avg_Length'
```

```
    'L4 17-4'[45] [ 0] [ 0] [ 0]
    'L4 arl-8'[41] [ 0] [ 0] [ 1]
allMeans =
    [] 'N''Avg_Spacing''Avg_Coverage''Avg_Length'
    'L4 17-4'[45] [ 12.3305] [ 1.7442] [ 21.7853]
    'L4 arl-8'[41] [ 11.4005] [ 1.7129] [ 19.9497]
allStds =
    [] 'N''Avg_Spacing''Avg_Coverage''Avg_Length'
    'L4 17-4'[45] [ 1.7165] [ 0.3266] [ 2.6545]
    'L4 arl-8'[41] [ 1.4945] [ 0.3592] [ 3.2746]
```

4 Split the Neuron

4.1 splitNeuron

This function takes every line scan file catalogued in an output file of MTQuant and calculates the organization parameters for each segment of the neuron.

4.1.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
statsFile	the name of an output file generated by MTQuant
splitSegLen	length of each segment in the split in pixels (not microns)
groups (optional)	a list of group numbers to compare. These are the values of the DirNum column of the output file (see Section 4 of the MTQuant documentation for more information). If groups is not inputted, or it is empty (i.e. groups=[];), the function compares all groups.
rPeakTol (optional)	See MTQuant documentation
toUseRandRPeaks (optional)	See MTQuant documentation
rPeakCorr (optional)	See MTQuant documentation
toUseHalfMTs (optional)	See MTQuant documentation
OUTPUT ARGUMENTS	
fileOut	name of the output file containing the split information. It is the same as statsFile, but with the string '_split' appended to it. The first three columns of the output file are the same as the first three columns of statsFile. The remaining columns are the average spacing, std dev of spacing, single MT brightness, average coverage, std dev of coverage, and average length for each segment. The column names are the same as in statsFile, but with 'S*_' prepended to each, where the asterisk (*) is the segment number. The segment number 1 refers to the segment with the green star in Figure 1 in the MTQuant documentation.

4.1.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2.csv';
>> splitSegLen = 15/0.170;
>> fileOut = splitNeuron(statsFile,splitSegLen);
```

4.2 splitAxonComm

This function is almost identical to `splitNeuron`, but it also creates segments for the axon close to the cell body and the commissure. It expects the line scan files to be `'*_acbLineScans.csv'`, `'*_commLineScans.csv'`, and `'*_axonLineScans.csv'` for the line scans around the cell body, of the commissure, and the remainder of the axon, respectively.

4.2.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
`statsFile`	the name of an output file generated by `MTQuant`
`splitSegLen`	length of each segment in the split, in pixels (not microns)
`groups` (optional)	a list of group numbers to compare. These are the values of the `DirNum` column of the output file (see Section 4 of the `MTQuant` documentation for more information). If `groups` is not inputted, or it is empty (i.e. `groups=[];`), the function compares all groups.
`rPeakTol` (optional)	See `MTQuant` documentation
`toUseRandRPeaks` (optional)	See `MTQuant` documentation
`rPeakCorr` (optional)	See `MTQuant` documentation
`toUseHalfMTs` (optional)	See `MTQuant` documentation
OUTPUT ARGUMENTS	
`fileOut`	name of the output file containing the split information. It is the same as `statsFile`, but with the string `'_splitComm'` appended to it. The first three columns of the output file are the same as the first three columns of `statsFile`. The remaining columns are the average spacing, std dev of spacing, single MT brightness, average coverage, std dev of coverage, and average length for each segment. The column names are the same as in `statsFile`, but with `'S*_'` prepended to each, where the asterisk (*) is the segment name or number. The segment number 1 refers to the segment with the green star in Figure 1 in the `MTQuant` documentation.

4.2.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2.csv';
>> splitSegLen = 15/0.170;
>> fileOut = splitAxonComm(statsFile,splitSegLen);
```

4.3 splitNeuronEM

This function takes every segment from a '*_split.csv' file and applies the distribution refinement to every organization parameter, DirNum, and segment combination.

4.3.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
statsFile	the name of a '*_split.csv' of a '*_splitComm.csv' output file generated by splitNeuron or splitAxonComm
OUTPUT ARGUMENTS	
fileOut	name of the output file containing the split information. It is the same as statsFile, but with the string '_em' appended to it. The first three columns of the output file are the same as the first three columns of statsFile. The remaining columns are the average spacing, std dev of spacing, single MT brightness, average coverage, std dev of coverage, and average length for each segment. The column names are the same as in statsFile, but with 'S*_' prepended to each, where the asterisk (*) is the segment name or number. The segment number 1 refers to the segment with the green star in Figure 1 in the MTQuant documentation.

4.3.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2_split.csv';
>> fileOut = splitNeuronEM(statsFile);
```

4.4 splitNeuronPlots

This function takes a '*_split.csv', '*_splitComm.csv', '*_split_em.csv', or '*_splitComm_em.csv' file and generates multiple plots of the organization parameters over the segments in the file. For every parameter specified by the input stats, and for every animal group specified by the input groups, the function generates two plots. One is the average value of the parameter over each segment for that group with error bars. The second plot is the parameter for each animal over each segment. The plots are saved with different suffixes added to the string in fileOut.

4.4.1 Input & Output Arguments

Argument	Description
Input Arguments	
statsFile	the name of an output file generated by MTQuant
fileOut	the base name of the output plots. The names of the parameters and/or the numbers of the groups are added to fileOut and the plots are saved as .BMP files.
groups (optional)	a list of group numbers to compare. These are the values of the DirNum column of the output file (see Section 4 of the MTQuant documentation for more information). If groups is not inputted, or it is empty (i.e. groups=[];), the function compares all groups.
stats (optional)	list of columns in statsFile for which to calculate the correlation. This can be a list of column numbers or a list of strings, where each string is the column name (See Section 5.2 of MTQuant documentation for complete list)

4.4.2 Example Usage

```
>> splitNeuronPlots('C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2_split.csv')
```

5 Simulations (runModel)

To run the model, use the function `runModel`. This function accepts as input the number of times to run the model, i.e. re-simulate the MTs and analyze them, and the scaling factors by which to change the spacing and the length. It also accepts the same name-value parameters that would be sent to `MTQuant` in order to see the results of the model with various settings in the algorithm. The function generates `numRuns` MTs and corresponding blurred, noisy line scans. It saves those line scans into `'*LineScans.csv'` files in a temporary directory created in the location from which `runModel` is being called. Then MTQuant is called, on the line scans in that temporary directory. That temporary directory is deleted at the end of the function.

5.1 Input & Output Arguments

Argument	Description
Input Arguments	
`numRuns`	The number of times to run the model
`spcFactor`	The factor by which to multiply the randomly drawn spacing of each MT. If in doubt, set this input to 1. To increase coverage, set this input to something less than 1
`lenFactor`	The factor by which to multiply the randomly drawn spacing of each MT. If in doubt, set this input to 1. To increase coverage, set this number of something greater than 1
`MTQuantArgs` (optional)	Cell array of strings containing a list of name-value pairs to send to `MTQuant`
Output Arguments	
`modelB`	Randomly selected single MT brightness for each run of the simulation
`modelS`	Actual spacing for each model run, extracted from generated MTs
`modelC`	Actual coverage for each model run, extracted from generated MTs
`modelL`	Actual length for each each model run, extracted from generated MTs
`calcB`	Single MT brightness for each model run, calculated with `MTQuant`
`calcS`	Spacing for each run of the simulation, calculated using `MTQuant`
`calcC`	Coverage for each run of the simulation, calculated using `MTQuant`
`calcL`	Length for each run of the simulation, calculated using `MTQuant`

5.2 Example Usage

```
>> MTQuantArgs = {'toEM',false,'rPeakTol',0.01};
>> [modelB,modelS,modelC,modelL,calcB,calcS,calcC,calcL] = ...
      runModel(100,1,1,MTQuantArgs);
```

6 Visualize MTs (`makeVisualization`)

The function `makeVisualization` takes a cell array of `'*LineScans.csv'` and saves visualization plots for each file. It also requires the identified single MT brightness and number of MTs corresponding to each file, which can be found in the output data files of `MTQuant`. The output files are saved in the same place as the `'*LineScans.csv'` files, but with the endings `'*_visualization1.bmp'` and `'*_visualization2.bmp'`. The first file is a visualization with the MTs tiled one after another, while the second file is a visualization with the MTs more densely packed.

6.1 Input & Output Arguments

Argument	Description
INPUT ARGUMENTS	
`filesToPrint`	cell string array of `'*LineScans.csv'` files to visualize
`brightnessArray`	array of single MT brightnesses corresponding to the files in `filesToPrint`. Expected to be the same length as the array `filesToPrint`.
`numMTArray`	array of extracted number of MTs for each file in `filesToPrint`. Expected to be the same length as the array `filesToPrint`.
`roundLevel` (optional)	level at which to round the green line scan up or down (default is 0.5, i.e. normal rounding) when calculating the quantized signal
`pairMode` (optional)	0 to pair MT ends with MT start that generates the longest MT 1 to pair MT ends with MT start that generates shortest MT 2 to pair MT ends with a random MT start.

6.2 Example Usage

```
>> statsFile = 'C:/Users/Roshni/Desktop/MTQuantTest/axonTest_2.csv'
>> T = readtable(statsFile); % Easy way to read statsFile
>> C = table2cell(T)); % Convert to cell for easy manipulation
>> % Create array of files for which to generate visualizations
>> fileNames = strcat(C(50:59,1),'/',C(50:59,2));
>> brightnessArray = cell2mat(C(50:59,11)); % Extract brightnesses from statsFile
>> numMTArray = cell2mat(C(50:59,8)); % Extract num MTs from statsFile
>> makeVisualization(fileNames,brightnessArray,numMTArray);
```