# MzqLibrary user guide

Version 1.0-beta

Date: 17/11/2014

Contents

[MzqLibrary user guide 1](#_Toc404082175)

[1 Overview 2](#_Toc404082176)

[2 Importer 3](#_Toc404082177)

[2.1 ProgenesisConverter 3](#_Toc404082178)

[2.2 MaxquantConverter 3](#_Toc404082179)

[2.3 ConsensusXMLConverter 4](#_Toc404082180)

[3 Exporter 4](#_Toc404082181)

[3.1 MzTabConverter 4](#_Toc404082182)

[3.2 HtmlConverter 4](#_Toc404082183)

[3.3 CsvConverter 5](#_Toc404082184)

[3.4 XlsConverter 5](#_Toc404082185)

[4 Processing 6](#_Toc404082186)

[4.1 MzqMzIdMapping 6](#_Toc404082187)

[4.2 Normalisation 6](#_Toc404082188)

[4.3 ProteinInference 7](#_Toc404082189)

[4.4 AnovaPValue 7](#_Toc404082190)

[5 MzqViewer 8](#_Toc404082191)

[6 Common errors when using the library 9](#_Toc404082192)

# Overview

The library can be used either in command line mode or in GUI (visualized) mode; both are available to download from home page (<http://code.google.com/p/mzq-lib/> ).

In the following section, we describe the purpose of each tool in the library, the input and output file types and the mandatory and optional parameters. Optional parameters are denoted by square brackets.

To run the mzqLibrary from the comConvertermand line, the following command structure and ordering **must** be used:

java -jar "path\_to\_jar\mzqLibrary-version.jar" FUNCTION INPUT\_FILE OUTPUT\_FILE PARAMS

where FUNCTION = “ProgenesisConverter”, “MaxquantConverter”, “ConsensusXMLConverter”, “MzTabConverter”, “HtmlConverter”, “CsvConverter”, “XlsConverter”, “MzqMzIdMapping”, “Normalisation”, “ProteinInference”, “AnovaPValue”.

The function name, input file path, output file path and each parameter may be enclosed with double quotes. Double quotes are essential around the file paths if they contain a space, such as:

“C:\Windows\Documents and settings\jonesar\myfilename.mzq”

If the file name includes the character of ‘%’, escape character ‘%’ is needed. Hence replace any ‘%’ by “%%” in the file name.

***Please note – the input file name and output file name MUST be different otherwise an error will result.***

To run the mzqLibrary in the GUI interface, the TOOL\_NAME, INPUT\_FILE and OUTPUT\_FILE are selected as shown in Figure 1, and the parameters need to be entered manually in the “Options” text box.

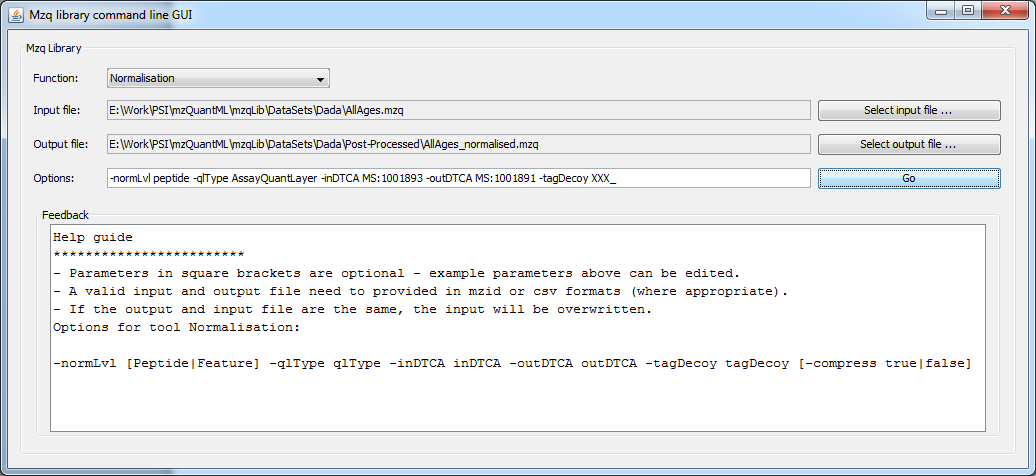


Figure 1 The graphical interface for the mzqLibrary. The user should select the tool (Function), the input file and output file as shown. The structure of the parameters entered under “Options” is the same as for the command line mode.

To run the mzqLibrary command line GUI, mzqView has to be run first. See section MzqViewer for detail.

# Importer

## ProgenesisConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].csv |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -pepList peptideListFile [-sep CsvSeparator] [-proteinGroupList true|fasle] [-rawPlusNorm norm|raw] |
| **Description** | This tool converts Progenesis result files to a single mzQuantML file. User needs to provide protein list CSV file as the input file and provide peptide list (or feature list) CSV file via option. User can also specify the separator used in the input CSV files in order for correctly reading them (although Progenesis output CSV files used comma separator by default). User has the choice to convert protein group list or not and to export only raw or normalised abundance by setting the ‘rawPlusNorm’ parameter. If none of the option parameters is provided, the default is to use comma separator, includes protein group list and all the existing abundance measurement. |

Example options for GUI:

-pepList peptide\_list.csv -proteinGroupList false -rawPlusNorm raw

Example command line:

java -Xms1024m -jar "mzqLibrary-1.0-beta.jar" ProgenesisConverter "Progenesis\_protein.csv" "mzqLibraray-demo-no-proteinGroupList-no-normalised.mzq" -pepList "Progenesis\_peptide.csv" -proteinGroupList false -rawPlusNorm raw

## MaxquantConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input] evidence.txt |
| **OUTPUT\_FILE** | [output]. mzq |
| **PARAMS** | -summary summaryFile -peptides peptidesFile -proteinGroups proteinGroupsFile -template experimentalDesignTemplateFile |
| **Description** | This tool converts Maxquant result files to a single mzQuantML file. User needs to select the "evidence.txt" from Maxquant as the input file. Provide the other mandatory files via options. To obtain all the required information for mzq file from MaxQuant result files, user must have five files ready: evidence.txt (as INPUT\_FILE), summary.txt, peptides.txt, proteinGroups.txt, templateExperimentalDesignTemplate.txt (as PARAMS). |

Example options for GUI:

-summary summary.txt -peptides peptides.txt -proteinGroups proteinGroups.txt -template ExperimentalDesignTemplate.txt

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" MaxquantConverter "Z:\MaxQuant\combined\txt\evidence.txt" "Z:\MaxQuant\output.mzq" -summary "Z:\MaxQuant\combined\txt\summary.txt" -peptides "Z:\MaxQuant\combined\txt\peptides.txt" -proteinGroups "Z:\MaxQuant\combined\txt\proteinGroups.txt" -template "Z:\MaxQuant\combined\txt\ExperimentalDesignTemplate.txt"

## ConsensusXMLConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input]. consensusXML |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -compress true|false |
| **Description** | This tool converts a consensusXML file from openMS to an mzQuantML file. |

Example options for GUI:

-compress false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" ConsensusXMLConverter "E:\OpenMS\FLUQT\CPTAC\_study6\_FLUQT.consensusXML" "E:\OpenMS\CPTAC\_study6\_FLUQT.mzq" –compress false

# Exporter

## MzTabConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input]. mzq or [input]. mzq.gz |
| **OUTPUT\_FILE** | [output].mztab |
| **PARAMS** | -compress true|false |
| **Description** | This tool converts an mzQuantML file to an mzTab file. |

Example options for GUI:

- -compress false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" MzTabConverter "E:\OpenMS\CPTAC\_study6\_FLUQT.mzq" "E:\OpenMS\FLUQT\CPTAC\_study6\_FLUQT.mztab" –compress false

## HtmlConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq or [input].mzq.gz |
| **OUTPUT\_FILE** | [output].html |
| **PARAMS** | -compress true|false |
| **Description** | This tool converts an mzQuantML file to a HTML file. |

Example options for GUI:

-compress true|false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" HtmlConverter "E:\OpenMS\CPTAC\_study6\_FLUQT.mzq" "E:\OpenMS\FLUQT\CPTAC\_study6\_FLUQT.html" –compress false

## CsvConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq or [input].mzq.gz |
| **OUTPUT\_FILE** | [output].csv |
| **PARAMS** | -compress true|false |
| **Description** | This tool converts an mzQuantML file to a CSV file. |

Example options for GUI:

-compress true|false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" CsvConverter "E:\OpenMS\CPTAC\_study6\_FLUQT.mzq" "E:\OpenMS\FLUQT\CPTAC\_study6\_FLUQT.csv" –compress false

## XlsConverter

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq or [input].mzq.gz |
| **OUTPUT\_FILE** | [output].xls |
| **PARAMS** | -compress true|false |
| **Description** | This tool converts an mzQuantML file to a XLS file. |

Example options for GUI:

-compress true|false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" XlsConverter "E:\OpenMS\CPTAC\_study6\_FLUQT.mzq" "E:\OpenMS\FLUQT\CPTAC\_study6\_FLUQT.xls" –compress false

# Processing

## MzqMzIdMapping

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq or [input].mzq.gz |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -rawToMzidMap rawToMzidMapString [-compress true|false] |
| **Description** | This tool reassigns consensus peptide sequence by mapping the identifications across input mzid files on quantified features in input mzq file. The -rawToMzidMap parameter is a string composed of pairs of a raw file name (file name only) and its mzid file (with **absolute file path**), using semicolon as separator. The raw file names must be the same as those in the input mzq file under *<InputFiles>* element. |

Example options for GUI:

-rawToMzidMap filename1.raw;C:\Data\Mzid\filename1.mzid;filename2.raw;C:\Data\Mzid\filename2.mzid -compress false

Example command line:

java -Xms1024m -jar "E:\Lib\trunk\dist\mzqLibrary-1.0-beta.jar" MzqMzIdMapping "E:\OpenMS\ CPTAC\_study6\_2400\_3600\_FLUQT.consensusXML.mzq" "CPTAC\_study6\_2400\_3600\_FLUQT.consensusXML-mapped.mzq" –rawToMzidMap "***mam\_042408o\_CPTAC\_study6\_6B011.raw***; **E:\OpenMS\mam\_042408o\_CPTAC\_study6\_6B011\_rt.mzid**; ***mam\_050108o\_CPTAC\_study6\_6B011.raw***; **E:\OpenMS\ mam\_050108o\_CPTAC\_study6\_6B011\_rt.mzid**" –compress false

## Normalisation

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -normLvl Peptide|Feature -inDTCA inputDataTypeCvAccession -outDTCA outputDataTypeCvAccession -outDTCN outputDataTypeCvName -tagDecoy tagDecoy [-compress true|false] |
| **Description** | This tool calculates normalised measurement of specified AssayQuantLayer in one specified list (peptide or feature). The –inDTCA indicate the QuantLayer to be used in the calculation. The result is exported in an extra QuantLayer of the original mzq file with the CV accession and name specified in –outDTCA and –outDTCN. The tool automatically selects the best reference assay. |

Example options for GUI:

-normLvl peptide -inDTCA MS:1001840 -outDTCA MS:1001891 -outDTCN Progenesis:peptide normalised abundance -tagDecoy XXX\_

Example command line:

java -Xms1024m -jar "mzqLibrary-1.0-beta.jar" Normalisation "DemoOutputs/mzqLibraray-demo-no-proteinGroupList-no-normalised.mzq" "DemoOutputs/mzqLibrary-demo-no-proteinGroupList-normalised.mzq" -normLvl peptide -inDTCA MS:1001893 -outDTCA MS:1001891 -outDTCN "Progenesis:peptide normalised abundance" -tagDecoy XXX\_ -compress false

## ProteinInference

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -op sum|mean|median -inPepNormCA inPeptideNormCvAccession -inPepRawCA inPeptideRawCvAccession -outPGNormCA outProteinGroupNormCvAccession -outPGNormCN outProteinGroupNormCvName -outPGRawCA outProteinGroupRawCvAccession -outPGRawCN outProteinGroupRawCvName |
| **Description** | This tool performs protein inference and calculates the abundance from specified *AssayQuantLayer*. |

Example options for GUI:

-op sum -inPepNormCA MS:1001891 -inPepRawCA MS:1001893 -outPGNormCA MS:1002518 -outPGNormCN "Progenesis:protein group normalised abundance" -outPGRawCA MS:1002519 -outPGRawCN "Progenesis:protein group raw abundance"

Example command line:

java -Xms1024m -jar "mzqLibrary-1.0-beta.jar" ProteinInference "DemoOutputs/mzqLibrary-demo-no-proteinGroupList-normalised.mzq" "DemoOutputs/mzqLibrary-demo-with-proteinGroupList-normalised.mzq" -op sum -inPepRawCA MS:1001893 -inPepNormCA MS:1001891 -outPGNormCA MS:1002518 -outPGNormCN "Progenesis:protein group normalised abundance" -outPGRawCA MS:1001892 -outPGRawCN "Progenesis:protein group raw abundance"

## AnovaPValue

|  |  |
| --- | --- |
| **INPUT\_FILE** | [input].mzq |
| **OUTPUT\_FILE** | [output].mzq |
| **PARAMS** | -listType Protein|ProteinGroup -qlDTCA qlDataTypeCvAccession -assayIdsGroup assayIdsGroup –[compress true|false] |
| **Description** | This tool calculates one-way ANOVA p value of specified QuantLayer for either ProteinGroupList or ProteinList. The QuantLayer is specified by passing Cv accession to qlDTCA option. The group of assays included in the ANOVA calculation is provided by the flat string option of assayIdsGroup. The whole string is divided into groups which are separated by ";" (semicolon) and in each group, the member assay ids are separated by "," (comma). |

Example options for GUI:

-listType ProteinGroup -qlDTCA MS:1002518 -assayIdsGroup ass\_0,ass\_1,ass\_2,ass\_3,ass\_4;ass\_5,ass\_6,ass\_7,ass\_8,ass\_9

Example command line:

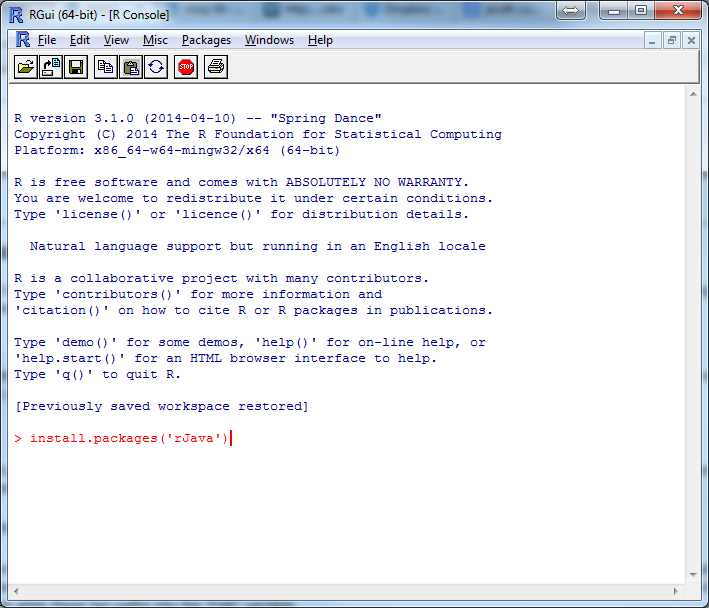
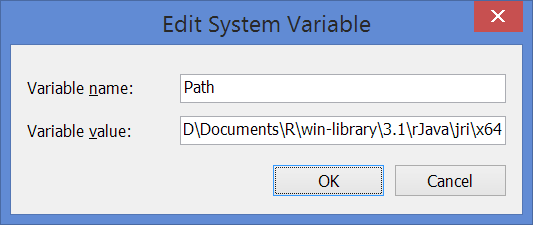
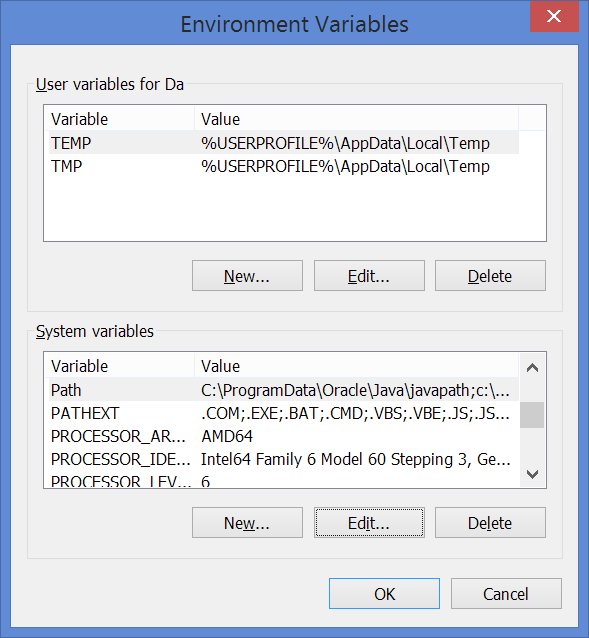
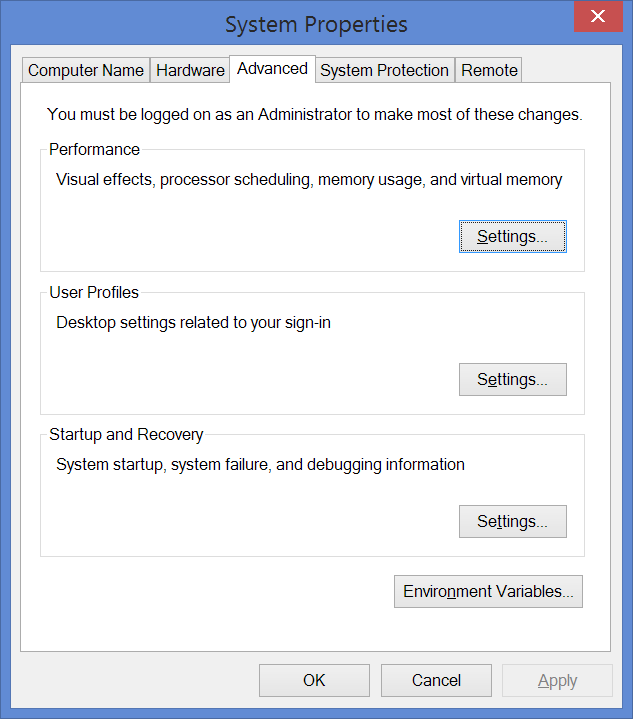
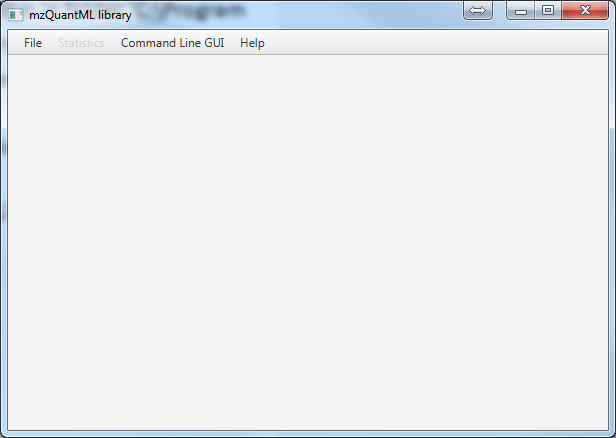
java -Xms1024m -jar "mzqLibrary-1.0-beta.jar" AnovaPValue "DemoOutputs/mzqLibrary-demo-with-proteinGroupList-normalised.mzq" "DemoOutputs/mzqLibrary-demo-final.mzq" -listType ProteinGroup -qlDTCA MS:1002518 -assayIdsGroup ass\_0,ass\_1,ass\_2,ass\_3,ass\_4,ass\_5,ass\_6,ass\_7,ass\_8,ass\_9,ass\_10,ass\_11,ass\_12;ass\_13,ass\_14,ass\_15,ass\_16,ass\_17,ass\_18,ass\_19,ass\_20,ass\_21,ass\_22,ass\_23,ass\_24,ass\_25 -compress false

# MzqViewer

The mzqViewer (Figure 4) can perform some statistic routines (e.g. heat map) based on R language by calling JRI API. The mzqViewer is also the gateway to call up the mzqLibrary command line GUI window. Please follow the steps below to install required files and libraries before using mzqViewer:

1. Install R

Download the latest version of R project from <http://cran.r-project.org/mirrors.html> and install. There are detailed guides of installing R on various OS <http://cran.r-project.org/manuals.html>.

1. Install JRI  
   Run "install.packages('rJava')" either in R GUI or command line to install the rJava library (rJava and JRI used to be separated library. JRI is now bundled in rJava). By default rJava is installed in R\_Folder/library/rJava, sometimes rJava will ask user to choose an installation folder due to access denied to the default folder. Remember where rJava is installed.  
     
     
   Figure Install rJava package from RGui
2. Find out if the PC is 32 bit or 64 bit. <http://windows.microsoft.com/en-GB/windows7/find-out-32-or-64-bit>.
3. Set up environment variables (Windows only). Add the path to R.dll and jri.dll into the system variable 'Path' (see figures below). For example, on a 64 bit PC, the R.dll is in folder 'C:\Program Files\R-3.1\bin\x64' and jri.dll is in folder 'C:\Program Files\R-3.1\library\rJava\jri\x64' or 'C:\Users\myusrname\Documents\R\win-library\3.1\rJava\jri\x64', then adds these two paths into the 'Path' variable.   
     
     
   Figure Set system variable Path before run mzqViewer
4. To run the MzqViewer use the following command or simply double click mzqViewer.bat file from the latest download zip file:  
   java -Xms1024m -cp "mzqLibrary-1.0-beta.jar" uk.ac.liv.mzqlib.MainApp  
     
     
   Figure 4 The interface of mzqViewer

# Common errors when using the library

The most common problems result from the following:

* The input file or output file is not in the correct format. All mzQuantML files must be valid with respect to version 1.0. This can be checked using the mzQuantML validator (<http://code.google.com/p/mzquantml-validator/>)
* The parameters are incorrectly specified, please check these carefully against the instructions before submitting a bug report.
* An invalid CV terms is given. Please check that the accession for the CV term(s) you provide are valid (sourced from <http://psidev.cvs.sourceforge.net/viewvc/psidev/psi/psi-ms/mzML/controlledVocabulary/psi-ms.obo>) and they are present in your file.
* Out of memory errors. You may need to increase the memory available to Java by editing the command for the routine or the graphical interface. Example providing a large amount of memory (3072 MB):  
  java –Xms3072m –jar "mzqLibrary-version.jar"

If you believe you have identified a bug in the library, or you cannot solve an error yourself. Please post on the issues list, supplying a full description of your problem, the command you used, and the input file(s).gz (zipped): <http://code.google.com/p/mzq-lib/issues/list>.