

Package ‘WMBrukerParser’

January 2, 2012

Type Package

Title William and Mary Parser For Bruker-Ultraflex Mass Spec Data

Version 1.2

Date 2009-03-01

Author William Cooke, Maureen Tracy and Dariya Malyarenko

Maintainer William Cooke <wecook@wm.edu>

Description The package parses Bruker Ultraflex TOF mass spectrometry data and experimental parameter files into R structures for further signal processing and statistical analysis.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2010-03-01 20:53:49

R topics documented:

WMBrukerParser-package	2
BrukerParser	2
ParseAndSave	4
ParserParams	6
tofList	7
tofListMetaData	8

Index	11
--------------	-----------

WMBrukerParser-package

William and Mary Parser for Bruker-Ultraflex TOF-MS Data

Description

The package parses Bruker-Ultraflex TOF mass spectrometry data and experimental parameter files into R structures for further signal processing and statistical analysis.

Details

Package: WMBrukerParser
Type: Package
Version: 1.2
Date: 2010-03-01
License: GPL(version 2 or later)

The package contains two routines. The core routine, BrukerParser, reads binary TOF data (obtained using either Linear or Reflector mode) and associated experimental parameter files, and parses data and meta-data into R structures, tofList and tofListMetaData. This involves accessing files in different locations within a data directory tree and reading different file formats. The second routine, ParseAndSave calls the BrukerParser routine as directed by the parameter list, ParserParams (provided in an OptionsAndParameters.txt file), and saves the tofList and tofListMetaData structures that are returned in .Rdat files. Parameters in OptionsAndParameters can be edited to select options for parsing data from multiple bioprocessor runs, concatenating parsed data and printing memory usage information.

Author(s)

William Cooke, Maureen Tracy and Dariya Malyarenko, College of William and Mary
Maintainer: William Cooke <wecook@wm.edu>

See Also

[BrukerParser](#), [ParserParams](#), [ParseAndSave](#), [tofList](#), [tofListMetaData](#)

BrukerParser*Parser for Bruker-Ultraflex TOF Mass Spectrometry Data*

Description

This routine reads binary fid Bruker files from a single bioprocessor run, along with associated experimental files and creates R structures, tofList and tofListMetaData.

Usage

```
BrukerParser(dataSource, dataDirectory)
```

Arguments

dataSource Typically laboratory where data was obtained
 dataDirectory Data directory path.

Details

The BrukerParser routine is called by the ParseAndSave routine using ParserParams\$dataSource and the path built from ParserParams\$dataDirLeft, ParserParams\$runIndices and ParserParams\$dataDirRight.

This routine looks for the following contents in the dataDirectory:

1. a subdirectory with "calibration" in the name (optional)
2. a file with the extension .par (optional)
3. a file with the extension .axe (optional)
4. a file with the extension .isset (optional)
5. a file named sample.xml (optional)
6. spot subdirectories with names like 0_A1 and 0_L5 (required), each containing:
 - EITHER (for Linear TOF data):
 - /1/1SLin/fid - the binary data file (required)
 - /1/1SLin/acqu - the acquisition information file (required)
 - /1/1SLin/pdata/1/proc - the processing file (required)
 - OR (for Reflector TOF data):
 - /1/1SRef/fid - the binary data file (required)
 - /1/1SRef/acqu - the acquisition information file (required)
 - /1/1SRef/pdata/1/proc - the processing file (required)

Value

tofList A list of time-of-flight mass spectrum vectors addressable by spectrumName.
 tofListMetaData A data.frame containing string values for experimental meta-data, with rows and columns addressable by spectrumName and attributeName.

Note

If any of the optional files are missing, the user has the option updating meta-data fields in the parsed meta-data structure, tofListMetaData. If the sample.xml file is missing, a field used in naming data and meta-data structures is missing and placeholders will be used. In this case, the user may wish to update structure names.

If the final line of the .axe file is not blank, R will generate a warning message such as: "In readLines(qq, n = -1) : incomplete final line found in 0-20KDaLin030308.axe" This has no effect on the reader and can be ignored.

Author(s)

William E. Cooke, College of William and Mary, wecook@wm.edu

See Also

[WMBrukerParser](#), [ParserParams](#), [ParseAndSave](#), [tofList](#), [tofListMetaData](#)

Examples

```
dataSource <- "Lab1"

directory = system.file("Examples", package = "WMBrukerParser")
dataDirectory <-
paste(directory, "/C3ValidationExtractSmall/RobotRun1/2-100kDa", sep="")

BrukerParser(dataSource, dataDirectory)
```

ParseAndSave

Parse and Save Bruker-Ultraflex TOF Mass Spectrometry Data

Description

This routine calls the BrukerParser to parse data from individual or multiple bioprocessor runs. It provides the option for concatenating data and meta-datastructures for multiple runs. It saves individual and concatenated tofList and tofListMetaData files as .Rdat files

Usage

```
ParseAndSave(ParserParams)
```

Arguments

ParserParams A list of seven parameters which identify the data to be parsed, and specify whether or not data from multiple bioprocessor runs are to be concatenated and if memory use information is to be printed during parsing. For convenience, it is stored in a file named OptionsAndParameters.txt file which can be edited and sourced prior to execution of ParseAndSave. It contains the parameters: dataSource, multipleRuns, concatLists, runIndices, dataDirLeft, dataDirRight and printMemoryUse

Details

Data directories contain spot directories with names such as 0_A1 or 0_B17. The code constructs the data directory paths using ParserParams\$dataDirLeft, ParserParams\$runIndices and ParserParams\$dataDirRight. If multiple runs are to be parsed, the ParserParams\$runIndices MUST appear in the paths. The "run branch" can occur at the level of the data directory or above. If it occurs at the data directory level,

ParserParams\$dataDirRight <- "". If a single run is to be parsed, ParserParams\$dataDirLeft is assigned the complete data directory path, ParserParams\$runIndices <- 1 and ParserParams\$dataDirRight <- "".

tofListRun#.Rdat and tofListMetaDataRow#.Rdat files are saved in dataDirectory (the directory containing the spot subdirectories) for "Run#". Concatenated files are saved in the directory of the final data parsed. For instance if runs 1, 3, and 2 were selected for concatenation (ParserParams\$runIndices <- c(1,3,2)), the files tofListCat_1_3_2.Rdat and tofListMetaDataRowCat_1_3_2.Rdat would be found in the dataDirectory for the Run 2.

Value

tofList A list of time-of-flight mass spectrum vectors addressable by spectrumName.
tofListMetaDataRow A data.frame containing string values for experimental meta-data, with rows and columns addressable by spectrumName and attributeName.

Warning

Since the data files are large, it may not be desirable or possible to concatenate data parsed from all runs in an experiment. The option to print memory usage through the parsing of multiple runs allows the user to consider which files to select for concatenation. Prior to selecting the option to concatenate data, it would be wise to parse all the data of interest, examine the memory usage throughout parsing in R and size of the resulting .Rdat files on the disk, keeping in mind the doubling of memory required for the further processing of this data.

Author(s)

Maureen Tracy, College of William and Mary, mbtrac@wm.edu

See Also

[WMBrukerParser](#), [BrukerParser](#), [ParserParams](#), [tofList](#), [tofListMetaDataRow](#)

Examples

```
## Example 1: Parse data from single run.

directory = system.file("Examples", package = "WMBrukerParser")
source(paste(directory, "/OptionsAndParametersParse1Run.txt", sep=""))

ParseAndSave(ParserParams)

## Example 2: Parse and concatenate data from two runs
## with memory use printed during parsing.

directory = system.file("Examples", package = "WMBrukerParser")
source(paste(directory, "/OptionsAndParametersParseAndCat2Runs.txt", sep=""))

ParseAndSave(ParserParams)
```

Description

ParserParams is a list of seven parameters which identify the data to be parsed, and specify whether or not data from multiple bioprocessor runs are to be concatenated and if memory use information is to be printed during parsing. For convenience, it is stored in a file named OptionsAndParameters.txt file which can be edited and sourced prior to execution of ParseAndSave.

Format

ParserParams\$dataSource : character string (e.g. "Lab1")
 ParserParams\$multipleRuns : character string ("yes" or "no")
 ParserParams\$concatLists : character string ("yes" or "no")
 ParserParams\$runIndices : numeric array (e.g. 1 or c(1,3,2))
 ParserParams\$dataDirLeft : character string (e.g. "scr/MyRun")
 ParserParams\$dataDirRight : character string (e.g. "QC")
 ParserParams\$printMemoryUse: character string ("yes" or "no")

Details

Data directories contain the spot directories with names such as 0_A1 or 0_B17. ParserParams\$dataDirLeft, ParserParams\$runIndices and ParserParams\$dataDirRight are defined so that pasting them together generates the data directory paths desired. If multiple runs are to be parsed, the ParserParams\$runIndices need not be in order but must appear in the paths. The "run branch" can occur at the level of the data directory or above. If it occurs at the data directory level, ParserParams\$dataDirRight <- "". If a single run is to be parsed, ParserParams\$dataDirLeft is assigned the complete data directory path, ParserParams\$runIndices <- 1 and ParserParams\$dataDirRight <- "".

See Also

[WMBrukerParser](#), [BrukerParser](#), [ParserParams](#), [ParseAndSave](#), [tofListMetaData](#)

Examples

```
## Example1: Parse data from a single bioprocessor run:

ParserParams<-list()
ParserParams$dataSource <- "Lab1"
ParserParams$multipleRuns <- "no"
ParserParams$concatLists <- "no"
ParserParams$runIndices <- 1

directory = system.file("Examples", package = "WMBrukerParser")
ParserParams$dataDirLeft <-
paste(directory, "/C3ValidationExtractSmall/RobotRun1/2-100kDa", sep="")
```

```
ParserParams$dataDirRight <- ""
ParserParams$printMemoryUse <- "no"

ParseAndSave(ParserParams)

## Example2: Parse and concatenate data from two bioprocessor
## two runs with memory information printed:

ParserParams<-list()
ParserParams$dataSource <- "Lab1"
ParserParams$multipleRuns <- "yes"
ParserParams$concatLists <- "yes"
ParserParams$runIndices <- c(1,2)

directory = system.file("Examples", package = "WMBrukerParser")
ParserParams$dataDirLeft <-
paste(directory, "/C3ValidationExtractSmall/RobotRun", sep="")

ParserParams$dataDirRight <- "/2-100kDa"
ParserParams$printMemoryUse <- "yes"

ParseAndSave(ParserParams)
```

tofList

Parsed TOF Spectrum List

Description

A list of time-of-flight mass spectrum vectors addressable by spectrumName.

Details

Spectrum vectors can also be addressed by spectrum index. This option should be used with caution since data and meta-data can be reordered independently during subsequent signal processing or statistical analysis.

In the event that default assignments were used for spectrumNames (due to missing sampleInfo.sampleNames meta-data) they can be updated.

See Also

[WMBrukerParser](#), [BrukerParser](#), [ParserParams](#), [ParseAndSave](#), [tofListMetaData](#)

Examples

```
## To access a spectrum vector:
```

```

data(tofListRun2)

spectrum <- tofList[[ "Pool 8_1"]];

## or:

spectrum <- tofList[[1]]; # Use with caution!

## To update a spectrum's vector:

  tofList[ "Pool 8_2"] <- list(spectrum)

## To update the tofList names:

newSpecNames<-list();
numSpec<-length(tofList);
for (i in 1:numSpec) {newSpecNames[i]<-paste("spec",i, sep="")};
names(tofList)<-newSpecNames;

```

tofListMetaData

Parsed TOF-MS Meta-Data

Description

tofListMetaData is a data.frame containing string values for experimental meta-data, with rows and columns addressable by spectrumName and attributeName, respectively.

Details

Meta-data can also be addressed using spectrum and attribute indices. This option should be used with caution since data and meta-data can be reordered independently during signal processing or statistical analysis.

Values can be updated if desired when tofListMetaData includes NAs due to missing optional parameter files.

Required meta-data attributes are parsed from required files.

Meta-data attributeName include:

acquisitionInfo.arrayBarcode - Target (Bruker plate) identifier string, includes target type and target serial number - Required

acquisitionInfo.ionPolarity - "positive" or "negative" - Required

acquisitionInfo.refId - Spectrum identifier, used for spectrum vector names in tofList and rows names in tofListMetaData (generated by concatenating strings: sampleInfo.sampleName, "_", and replicateNumber - Optional

acquisitionInfo.sourceFile - File name (with path) containing binary mass spectrum data, (from directory structure) - Automatic

acquisitionInfo.spotIndex - Spot index on MALDI plate - Automatic

acquisitionInfo.spotName - Spot name on MALDI plate - Automatic

adcGain.high - Digitizer gain factor - Optional

adcGain.low - Linear detector voltage - Optional
array.affinityType - Capture agent used for purification of biofluid samples - Optional
deflector.mode - 1 = linear, 2 = quadratic - Optional
experiment.id - Same as acquisitionInfo.refId - Optional
instrument.instrumentType - "Ultraflex Series" (hard coded) - Automatic
instrument.instrumentVendor - Name of instrument vendor - Required
instrument.serial - Instrument serial number - Required
instrumentSpecificSettings.adcBandwidth - Digitizer bandwidth limit - Optional
instrumentSpecificSettings.adcOffset - Digitizer offset, depends on spectrum type and gain factor - Optional
instrumentSpecificSettings.adcScale - Digitizer sensitivity, depends on gain factor - Optional
instrumentSpecificSettings.detectorBaseVoltage - Detector base voltage, depends on spectrum type - Optional
instrumentSpecificSettings.laserIntensityBaseRange - Minimum laser power - Optional
instrumentSpecificSettings.laserShots - Number of laser shots per spot - Required
instrumentSpecificSettings.timeZero - Initial time in counts (calculated) - Required
instrumentSpecificSettings.TOFmode - Spectrometer mode: LINEAR or REFLECTOR - Automatic
instrumentSpecificSettings.warmingShots - Number of warming laser shots - Optional
laserIntensity.units - % (hard coded) - Automatic
laserIntensity.value - Maximum laser power, percent of full scale - Optional
laserIntensityWarming.units - % (hard coded) - Automatic
laserIntensityWarming.value - Warming Laser power, percent of full scale - Optional
mass.units - "Da" (hard coded) - Automatic
mass.value - Low mass cut off for matrix suppression - Optional
massCalibration.calibrationSpectrumFile - Calibration folder name. (If unavailable "?" is assigned as default) - Optional
massCalibration.dateCalibrated - Calibration date - Required
massCalibration.equation - $c1*((T0+(X-1)*Tdelta)/U)^2+c0*((T0+(X-1)*Tdelta)/U)+c2$ (hard coded) - Automatic
massEnd.units - Mass units, "Da" (hard coded) - Automatic
massEnd.value - Highest mass in acquisition range - Optional
massStart.units - Mass units, "Da" (hard coded) - Automatic
massStart.value - Lowest mass in acquisition range - Optional
param.c0 - linear coefficient in calibration equation - Required
param.c1 - quadratic coefficient in calibration equation - Required
param.c2 - constant term in calibration equation - Required
param.Mode - Calibration mode: 1 = linear, 2 = quadratic - Optional
param.T0 - Digitizer delay in ns - Required
param.TDelta - Dwell time in ns - Required
param.U - Nanoseconds/milliseconds conversion factor, "1e6" (hard coded) - Automatic
replicateNumber - Index of occurrence of sampleInfo.sampleName - Automatic
sampleInfo.groupName - Clinical group sample is associated with, used for classification analysis - Optional
sampleInfo.sampleDescription - Additional sample description - Optional
sampleInfo.sampleName - Sample identifier (if missing, the directory name is used as the default value) - Optional
sampleInfo.sampleSource - Laboratory where data was acquired (user supplied input parameter: ParserParams.dataSource) - Required

software.version - version of XACQ software - Required
spottingInfo.spotProtocol - Laser movement pattern during data acquisition - Optional
timeOfFlightData.dateCreated - Data file creation date (from date stamp on fid file) - Automatic
timeOfFlightData.domain - Data domain, "time" (hard coded) - Automatic
timeOfFlightData.encoding - Data encoding, "base64" (hard coded) - Automatic
timeOfFlightData.end - Number of data points in spectra - Required
timeOfFlightData.offset - Initial value for offset of spectra, will be updated during alignment procedure, "0.0" (Hard coded) - Automatic
timeOfFlightData.pairOrder - Order of measurement pairs for TOF data, e.g., time - intensity, "t-int" (hard coded) - Automatic
timeOfFlightData.scale - Initial value for linear scale coefficient for spectra, updated during alignment: "1.0" (hard coded) - Automatic
timeOfFlightData.start - Index for onset of data acquisition, hard coded: "1" - Automatic

See Also

[WMBrukerParser](#), [BrukerParser](#), [ParserParams](#), [ParseAndSave](#), [tofList](#)

Examples

```

## To access a value:

data(tofListMetaDataRun2)

sampleName <- tofListMetaData["Pool 8_1", "sampleInfo.sampleName"];

## To update a value

tofListMetaData[ "Pool 8_2", "sampleInfo.sampleDescription" ] <- "Pool";

## To update the tofListMetaData rownames (desirable if default
## assignments were used due to missing sampleInfo.sampleNames):

newSpecNames<-list();
numSpec<-dim(tofListMetaData)[1];
for (i in 1:numSpec) { newSpecNames[i]<-paste("spec",i,sep="")};
row.names(tofListMetaData)<-newSpecNames;

```

Index

*Topic **IO**

- BrukerParser, [2](#)
- ParseAndSave, [4](#)
- WMBrukerParser-package, [2](#)

*Topic **classes**

- toList, [7](#)
- toListMetaData, [8](#)
- WMBrukerParser-package, [2](#)

*Topic **misc**

- ParserParams, [6](#)

[BrukerParser](#), [2](#), [2](#), [5-7](#), [10](#)

[ParseAndSave](#), [2](#), [4](#), [4](#), [6](#), [7](#), [10](#)

[ParserParams](#), [2](#), [4](#), [5](#), [6](#), [6](#), [7](#), [10](#)

[toList](#), [2](#), [4](#), [5](#), [7](#), [10](#)

[toListMetaData](#), [2](#), [4-7](#), [8](#)

[WMBrukerParser](#), [4-7](#), [10](#)

[WMBrukerParser](#)

([WMBrukerParser-package](#)), [2](#)

[WMBrukerParser-package](#), [2](#)