Package 'IDSL.CSA'

June 29, 2023

Type Package

Title Composite Spectra Analysis (CSA) for High-Resolution Mass Spectrometry Analyses

Version 1.2

Depends R (>= 4.0)

Imports IDSL.MXP, IDSL.IPA, IDSL.FSA, readxl

Author Sadjad Fakouri-Baygi [aut] (<https://orcid.org/0000-0002-6864-6911>), Dinesh Barupal [cre, aut] (<https://orcid.org/0000-0002-9954-8628>)

Maintainer Dinesh Barupal <dinesh.barupal@mssm.edu>

Description A fragmentation spectra detection pipeline for high-throughput LC/HRMS data processing using peaklists generated by the 'IDSL.IPA' workflow <doi:10.1021/acs.jproteome.2c00120>. The 'IDSL.CSA' package can deconvolute fragmentation spectra from Composite Spectra Analysis (CSA), Data Dependent Acquisition (DDA) analysis, and various Data-Independent Acquisition (DIA) methods such as MS^E, All-Ion Fragmentation (AIF) and SWATH-MS analysis. The 'IDSL.CSA' package was introduced in <doi:10.1021/acs.analchem.3c00376>.

URL https://github.com/idslme/idsl.csa

BugReports https://github.com/idslme/idsl.csa/issues

Encoding UTF-8

Archs i386, x64

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-29 14:00:07 UTC

R topics documented:

aligned_fragmentation_spectra_annotator 2
CSA_adductAnnotator 3
CSA_alignedMetaSpectraCataloger 3

CSA_alignedPeaksTanimotoCoefficientCalculator 4
CSA_AlignedTable_xlsxAnalyzer 5
CSA_fragmentationPeakDetection 5
CSA_PARAM_SPEC 7
CSA_reference_xlsxAnalyzer 7
CSA_workflow 8
CSA_xlsxAnalyzer 9
DDA2msp 9
DDA_fragmentationPeakDetection 10
DDA_rawSpectraDeconvolution 11
DDA_workflow 11
DDA_xlsxAnalyzer 12
DIA_MS1_fragmentationPeakDetection 13
DIA_MS2_fragmentationPeakDetection 14
DIA_workflow 15
DIA_xlsxAnalyzer 16
IDSL.CSA_MSPgenerator 16
IDSL.CSA_referenceMSPgenerator 17
IDSL.CSA_workflow 17
IDSL.CSA_xlsxAnalyzer 18
negativeAdducts 18
positiveAdducts 19

Index **20**

`aligned_fragmentation_spectra_annotator`
Aligned Fragmentation Spectra Annotator

Description

This function detects frequent matched compounds across multiple samples on the aligned peak table matrix.

Usage

```
aligned_fragmentation_spectra_annotator(PARAM_AT, output_path)
```

Arguments

`PARAM_AT`	a parameter driven from the ‘CSA_AlignedTable_xlsxAnalyzer‘ module.
`output_path`	output path

Value

This function stores ‘.Rdata‘ and ‘.csv‘ data from dataframe of aligned fragmentation spectra.

CSA_adductAnnotator *CSA Adduct Annotator*

Description

This function updates IDSL.IPA peaklists with IDSL.CSA grouping

Usage

```
CSA_adductAnnotator(IPApeakList, CSA_peaklist, massError)
```

Arguments

IPApeakList	IDSL.IPA peaklist
CSA_peaklist	A dataframe peaklist of co-detected CSA analysis.
massError	Mass accuracy in Da

Value

IDSL.IPA peaklists with IDSL.CSA grouping

CSA_alignedMetaSpectraCataloger *CSA Aligned Meta-Spectra Cataloger*

Description

This function generates integrated and most abundant aligned spectra from the aligned spectra

Usage

```
CSA_alignedMetaSpectraCataloger(address_input_msp, peakXcol, peak_height,
CSA_aligned_property_table, groupedID, minTanimotoCoefficient = 0.5,
number_processing_threads = 1)
```

Arguments

address_input_msp	address of the .msp files generated via IDSL.IPA DIA grouping
peakXcol	aligned indexed peak table
peak_height	aligned peak height table
CSA_aligned_property_table	a matrix of three columns of "IPA detection frequency", "median_height", and "median_R13C" for the aligned peak table

groupedID A 2-column dataframe of ‘Co-detectedIDs‘ and ‘TanimotoCoefficients‘ from the ‘CSA_alignedPeaksTanimotoCoefficientCalculator‘ module

minTanimotoCoefficient
minimum Tanimoto coefficient

number_processing_threads
Number of processing threads for multi-threaded processing

Value

A list of two objects for ‘MSP_integrated_aligned_spectra‘ and ‘MSP_most_abundant_aligned_spectra‘

CSA_alignedPeaksTanimotoCoefficientCalculator

CSA Aligned Peaks Tanimoto Coefficient Calculator

Description

This function groups co-detected peaks on the aligned table.

Usage

```
CSA_alignedPeaksTanimotoCoefficientCalculator(address_input_msp, peakXcol,
minPercenetageDetection = 5, minNumberFragments = 2, minTanimotoCoefficient = 0.1,
RTtolerance = 0.05, number_processing_threads = 1)
```

Arguments

address_input_msp
address of the .msp files generated via IDSL.IPA CSA aggregation

peakXcol aligned indexed peak table

minPercenetageDetection
minimum CSA frequency detection

minNumberFragments
minimum frequency

minTanimotoCoefficient
minimum Tanimoto coefficient

RTtolerance retention time tolerance to detect common peaks

number_processing_threads
Number of processing threads for multi-threaded processing

Value

A 2-column dataframe of ‘Co-detectedIDs‘ and ‘TanimotoCoefficients‘

`CSA_AlignedTable_xlsxAnalyzer` *CSA Aligned Table xlsx Analyzer*

Description

This function processes the spreadsheet of the 'AlignedTable' tab to ensure the parameter inputs are consistent with the requirements of the IDSL.CSA pipeline.

Usage

```
CSA_AlignedTable_xlsxAnalyzer(spreadsheet)
```

Arguments

`spreadsheet`	'AlignedTable' tab of the parameter spreadsheet

Value

This function returns the aligned table parameters to feed the 'aligned_fragmentation_spectra_annotator' function.

`CSA_fragmentationPeakDetection` *CSA peakList MSP generation*

Description

This function detects fragmentation peaks for the Composite Spectra Analysis (CSA) using IDSL.IPA peaklists.

Usage

```
CSA_fragmentationPeakDetection(CSA_hrms_address, CSA_hrms_file,
tempAlignedTableSubsetsFolder = NULL, peaklist, selectedIPApeaks = NULL,
RTtolerance, massError, minSNRbaseline, smoothingWindowMS1, scanTolerance, nSpline,
topRatioPeakHeight, minIonRangeDifference, minNumCSApeaks, pearsonRHOthreshold,
outputCSAeic = NULL)
```

Arguments

`CSA_hrms_address`	path to the HRMS file
`CSA_hrms_file`	CSA HRMS file
`tempAlignedTableSubsetsFolder`	tempAlignedTableSubsetsFolder
`peaklist`	IDSL.IPA peaklist
`selectedIPApeaks`	A vector of selected IDSL.IPA peaks only when a number of IDSL.IPA peaks from one peaklist is processed. When 'NULL' is selected, the entire peaks in the peaklist are processed.
`RTtolerance`	retention time tolerance to detect common peaks
`massError`	Mass accuracy in Da
`minSNRbaseline`	A minimum baseline S/N threshold for IDSL.IPA pseudo-precursor m/z
`smoothingWindowMS1`	number of scans for peak smoothing.
`scanTolerance`	a scan tolerance to extend the chromatogram for better calculations.
`nSpline`	number of points for further smoothing using a cubic spline smoothing method to add more points to calculate Pearson correlation rho values
`topRatioPeakHeight`	The top percentage of the chromatographic peak to calculate Pearson correlation rho values
`minIonRangeDifference`	Minimum distance (Da) between lowest and highest m/z to prevent clustering isotopic envelopes
`minNumCSApeaks`	Minumum number of ions in a CSA cluster
`pearsonRHOthreshold`	Minimum threshold for Pearson correlation rho values
`outputCSAeic`	When 'NULL' CSA EICs are not plotted. 'outputCSAeic' represents an address to save CSA EICs figures.

Value

A dataframe peaklist of co-detected CSA analysis.

References

[1] Fakouri Baygi, S., Kumar, Y., Barupal, D.K. (2022). IDSL.IPA Characterizes the Organic Chemical Space in Untargeted LC/HRMS Data Sets. *Journal of Proteome Research*, 21(6), 1485-1494, doi:10.1021/acs.jproteome.2c00120

[2] Fakouri Baygi, S., Fernando, S., Hopke, P.K., Holsen, T.M., Crimmins, B.S. (2021). Non-targeted discovery of novel contaminants in the Great Lakes region: A comparison of fish fillets and fish consumers. *Environmental Science & Technology*, 55(6), 3765-3774, doi:10.1021/acs.est.0c08507

CSA_PARAM_SPEC *CSA PARAM SPEC*

Description

default values for PARAM SPEC

Usage

```
data("CSA_PARAM_SPEC")
```

Format

A data frame on the following 2 variables.

`Parameter ID` a character vector

`User provided input` a numerical vector

Examples

```
data(CSA_PARAM_SPEC)
```

CSA_reference_xlsxAnalyzer *CSA reference xlsxAnalyzer*

Description

CSA reference xlsxAnalyzer

Usage

```
CSA_reference_xlsxAnalyzer(ref_xlsx_file, input_path_hrms = NULL, PARAM = NULL,
PARAM_ID = "", checkpoint_parameter = TRUE)
```

Arguments

`ref_xlsx_file`	ref_xlsx_file
`input_path_hrms`	input_path_hrms
`PARAM`	PARAM
`PARAM_ID`	PARAM_ID
`checkpoint_parameter`	checkpoint_parameter

Value

ref_table	ref_table
PARAM	PARAM
checkpoint_parameter	checkpoint_parameter

CSA_workflow *CSA workflow*

Description

This function executes the CSA workflow.

Usage

```
CSA_workflow(PARAM_CSA)
```

Arguments

PARAM_CSA	PARAM_CSA

Value

This module generates '.msp' files from DDA analysis.

Examples

```
s_path <- system.file("extdata", package = "IDSL.CSA")
SSh1 <- paste0(s_path,"/CSA_parameters.xlsx")
## To see the results, use a known folder instead of the `tempdir()` command
temp_wd <- tempdir()
temp_wd_zip <- paste0(temp_wd, "/idsl_csa_test_files.zip")
spreadsheet <- readxl::read_xlsx(SSh1, sheet = "CSA")
PARAM_CSA <- cbind(spreadsheet[, 2], spreadsheet[, 4])
download.file(paste0("https://github.com/idslme/IDSL.CSA/blob/main/",
                     "CSA_educational_files/idsl_csa_test_files.zip?raw=true"),
              destfile = temp_wd_zip, mode = "wb")
unzip(temp_wd_zip, exdir = temp_wd)
PARAM_CSA[2, 2] <- "NO"
PARAM_CSA[3, 2] <- "NO"
PARAM_CSA[5, 2] <- temp_wd
PARAM_CSA[8, 2] <- temp_wd
PARAM_CSA[9, 2] <- "NA"
PARAM_CSA[11, 2] <- temp_wd
## To ensure `PARAM_CSA` is consistent with the `CSA_workflow`
PARAM_CSA <- CSA_xlsxAnalyzer(PARAM_CSA)
##
CSA_workflow(PARAM_CSA)
```

CSA_xlsxAnalyzer *CSA xlsx Analyzer*

Description

This function processes the spreadsheet of the CSA parameters to ensure the parameter inputs are consistent with the requirements of the IDSL.CSA pipeline.

Usage

```
CSA_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	CSA tab of the parameter spreadsheet

Value

This function returns the CSA parameters to feed the 'CSA_workflow' function.

DDA2msp *DDA to msp*

Description

DDA to msp

Usage

```
DDA2msp(input_path_hrms, file_name_hrms = NULL, number_processing_threads = 1)
```

Arguments

input_path_hrms	path to the HRMS file
file_name_hrms	file_name_hrms
number_processing_threads	Number of processing threads for multi-threaded processing

Value

This module generates '.msp' files from DDA analysis.

Examples

```
## To see the results, use a known folder instead of the `tempdir()` command
temp_wd <- tempdir()
temp_wd_zip <- paste0(temp_wd, "/idsl_rawdda_test_files.zip")
download.file(paste0("https://github.com/idslme/IDSL.CSA/blob/main/",
                     "CSA_educational_files/idsl_rawdda_test_files.zip?raw=true"),
              destfile = temp_wd_zip, mode = "wb")
unzip(temp_wd_zip, exdir = temp_wd)
DDA2msp(input_path_hrms = temp_wd, file_name_hrms = NULL, number_processing_threads = 1)
```

`DDA_fragmentationPeakDetection` *DDA Fragmentation Peaks Detection*

Description

This function detects fragmentation peaks for the Data-Dependent Acquisition (DDA) analysis.

Usage

```
DDA_fragmentationPeakDetection(DDA_hrms_address, DDA_hrms_file, peaklist,
selectedIPApeaks, massErrorPrecursor, DDAprocessingMode = 'MostIntenseDDAspectra',
outputDDAspectra = NULL, number_processing_threads = 1)
```

Arguments

`DDA_hrms_address`
: path to the HRMS file

`DDA_hrms_file`
: DDA HRMS file

`peaklist`
: IDSL.IPA peaklist

`selectedIPApeaks`
: A vector of selected IDSL.IPA peaks only when a number of IDSL.IPA peaks from one peaklist is processed.

`massErrorPrecursor`
: Mass accuracy (Da) to find precursor m/z in IDSL.IPA peaklists

`DDAprocessingMode`
: c('MostIntenseDDAspectra', c('DDAspectraIntegration', massErrorIntegration), c('IonFiltering', massErrorIonFiltering, minPercentageDetectedScans, rsdCutoff, pearsonRHOthreshold)). Required variables for each DDA processing mode should be provided in this vector.

`outputDDAspectra`
: When 'NULL' DDA spectra are not plotted. 'outputDDAspectra' represents an address to save DDA spectra figures.

`number_processing_threads`
: Number of processing threads for multi-threaded processing

Value

A dataframe peaklist of co-detected DDA analysis.

`DDA_rawSpectraDeconvolution` *DDA Raw Spectra Deconvolution*

Description

This function stacks all DDA scans.

Usage

```
DDA_rawSpectraDeconvolution(DDA_hrms_address, DDA_hrms_file, rawDDAspectraVar = NULL,
number_processing_threads = 1)
```

Arguments

`DDA_hrms_address`	path to the HRMS file
`DDA_hrms_file`	DDA HRMS file
`rawDDAspectraVar`	c(NULL, list(precursorMZvec, precursorRTvec, massError, RTtolerance)). When NULL, all scans with precursor values are used for DDA peaklist generation. When the list is provided, it filters the scans with respect to predefined 'precursorMZvec' and 'precursorRTvec' values.
`number_processing_threads`	Number of processing threads for multi-threaded processing

Value

A dataframe stacked DDA scans.

`DDA_workflow` *DDA Workflow*

Description

This function runs the Data-Dependent Acquisition (DDA) analysis.

Usage

```
DDA_workflow(PARAM_DDA)
```

Arguments

PARAM_DDA	DDA parameters

Value

This module generates '.msp' files from DDA analysis.

Examples

```
s_path <- system.file("extdata", package = "IDSL.CSA")
SSh1 <- paste0(s_path,"/CSA_parameters.xlsx")
## To see the results, use a known folder instead of the `tempdir()` command
temp_wd <- tempdir()
temp_wd_zip <- paste0(temp_wd, "/idsl_dda_test_files.zip")
spreadsheet <- readxl::read_xlsx(SSh1, sheet = "DDA")
PARAM_DDA <- cbind(spreadsheet[, 2], spreadsheet[, 4])
download.file(paste0("https://github.com/idslme/IDSL.CSA/blob/main/",
                     "CSA_educational_files/idsl_dda_test_files.zip?raw=true"),
              destfile = temp_wd_zip, mode = "wb")
unzip(temp_wd_zip, exdir = temp_wd)
PARAM_DDA[2, 2] <- "no"
PARAM_DDA[4, 2] <- temp_wd
PARAM_DDA[7, 2] <- temp_wd
PARAM_DDA[8, 2] <- "NA"
PARAM_DDA[11, 2] <- temp_wd
## To ensure `PARAM_DDA` is consistent with the `DDA_workflow`
PARAM_DDA <- DDA_xlsxAnalyzer(PARAM_DDA)
##
DDA_workflow(PARAM_DDA)
```

DDA_xlsxAnalyzer *xlsx Analyzer for DDA analysis*

Description

This function processes the spreadsheet of the DDA spreadsheet tab to ensure the parameter inputs are in agreement with requirements of the Data-Dependent Acquisition (DDA) analysis.

Usage

```
DDA_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	DDA spreadsheet tab

Value

DDA parameters to feed the 'DDA_workflow' function.

DIA_MS1_fragmentationPeakDetection
CSA DIA MS1 Fragmentation Peaks Detection

Description

This function detects fragmentation peaks for the Data-Independent Acquisition (DIA) analysis at ms level 1.

Usage

```
DIA_MS1_fragmentationPeakDetection(DIA_hrms_address, DIA_hrms_file, peaklist,
selectedIPApeaks, massError, smoothingWindowMS1, scanTolerance, nSpline,
topRatioPeakHeight, intensityThresholdFragment, pearsonRHOthreshold, outputDIAeic = NULL,
number_processing_threads = 1)
```

Arguments

DIA_hrms_address	path to the HRMS file
DIA_hrms_file	DIA HRMS file
peaklist	IDSL.IPA peaklist
selectedIPApeaks	A vector of selected IDSL.IPA peaks only when a number of IDSL.IPA peaks from one peaklist is processed.
massError	Mass accuracy in Da
smoothingWindowMS1	number of scans for peak smoothing.
scanTolerance	a scan tolerance to extend the chromatogram for better calculations.
nSpline	number of points for further smoothing using a cubic spline smoothing method to add more points to calculate Pearson correlation rho values
topRatioPeakHeight	The top percentage of the chromatographic peak to calculate Pearson correlation rho values
intensityThresholdFragment	a value to represent intensity threshold for the fragment at the apex chromatogram scan
pearsonRHOthreshold	Minimum threshold for Pearson correlation rho values
outputDIAeic	When 'NULL' DIA EICs are not plotted. 'outputDIAeic' represents an address to save DIA EICs figures.
number_processing_threads	Number of processing threads for multi-threaded processing

Value

A dataframe peaklist of co-detected DIA analysis.

References

Fakouri Baygi, S., Fernando, S., Hopke, P.K., Holsen, T.M., Crimmins, B.S. (2021). Nontargeted discovery of novel contaminants in the Great Lakes region: A comparison of fish fillets and fish consumers. *Environmental Science & Technology*, 55(6), 3765-3774, doi:10.1021/acs.est.0c08507

`DIA_MS2_fragmentationPeakDetection`

CSA DIA MS2 Fragmentation Peaks Detection

Description

This function detects fragmentation peaks for the DIA analysis at MS level 2.

Usage

```
DIA_MS2_fragmentationPeakDetection(DIA_hrms_address, DIA_hrms_file, peaklist,
selectedIPApeaks, massError, smoothingWindowMS1, smoothingWindowMS2,
scanTolerance, nSpline, topRatioPeakHeight, intensityThresholdFragment,
pearsonRHOthreshold,  outputDIAeic = NULL, number_processing_threads = 1)
```

Arguments

`DIA_hrms_address`	path to the HRMS file
`DIA_hrms_file`	DIA HRMS file
`peaklist`	IDSL.IPA peaklist
`selectedIPApeaks`	A vector of selected IDSL.IPA peaks only when a number of IDSL.IPA peaks from one peaklist is processed.
`massError`	Mass accuracy in Da
`smoothingWindowMS1`	Number of scans for peak smoothing in MS1 channel
`smoothingWindowMS2`	Number of scans for peak smoothing in MS2 channel
`scanTolerance`	a scan tolerance to extend the chromatogram for better calculations.
`nSpline`	number of points for further smoothing using a cubic spline smoothing method to add more points to calculate Pearson correlation rho values
`topRatioPeakHeight`	The top percentage of the chromatographic peak to calculate Pearson correlation rho values

`intensityThresholdFragment`
: a value to represent intensity threshold for the fragment at the apex chromatogram scan in MS2 channel

`pearsonRHOthreshold`
: Minimum threshold for Pearson correlation rho values

`outputDIAeic`
: When 'NULL' DIA EICs are not plotted. 'outputDIAeic' represents an address to save DIA EICs figures.

`number_processing_threads`
: Number of processing threads for multi-threaded processing

Value

A dataframe peaklist of co-detected DIA analysis.

References

Fakouri Baygi, S., Fernando, S., Hopke, P.K., Holsen, T.M., Crimmins, B.S. (2021). Nontargeted discovery of novel contaminants in the Great Lakes region: A comparison of fish fillets and fish consumers. *Environmental Science & Technology*, 55(6), 3765-3774, doi:10.1021/acs.est.0c08507

`DIA_workflow` *DIA Workflow*

Description

This function runs the Data-Independent Acquisition (DIA) analysis.

Usage

```
DIA_workflow(PARAM_DIA)
```

Arguments

`PARAM_DIA` DIA parameters

Value

This module generates '.msp' files from DDA analysis.

DIA_xlsxAnalyzer *DIA xlsx Analyzer for DIA analysis*

Description

This function processes the spreadsheet of the DIA spreadsheet tab to ensure the parameter inputs are in agreement with requirements of the Data-Independent Acquisition (DIA) analysis.

Usage

```
DIA_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	DIA spreadsheet tab

Value

DIA parameters to feed the 'DIA_workflow' function.

IDSL.CSA_MSPgenerator *IDSL.CSA MSP Generator*

Description

This function creates standard .msp files that can also be used for Pepsearch.

Usage

```
IDSL.CSA_MSPgenerator(CSA_peaklist, msLevel, spectral_search_mode = "dda",
spectral_search_mode_option = NA, number_processing_threads = 1)
```

Arguments

CSA_peaklist	A dataframe peaklist of co-detected peaks
spectral_search_mode	Type of analysis. spectral_search_mode = c("dda", "dia", "csa")
msLevel	MS level = c(1, 2)
spectral_search_mode_option	Secondary type of analysis. spectral_search_mode_option = c(NA, "rawddaspectra", "alignedtable")
number_processing_threads	Number of processing threads for multi-threaded processing

Value

A string of standard .msp file

IDSL.CSA_referenceMSPgenerator *IDSL.CSA Reference MSP Generator*

Description

This function creates reference standard .msp files.

Usage

```
IDSL.CSA_referenceMSPgenerator(REF_peaklist, refTable, selectedIPApeaks_IDref, msLevel,
spectral_search_mode = "dda", spectral_search_mode_option = NA)
```

Arguments

REF_peaklist	A dataframe peaklist of co-detected peaks
refTable	reference CSA table
selectedIPApeaks_IDref	selectedIPApeaks_IDref
msLevel	MS level = c(1, 2)
spectral_search_mode	Type of analysis. spectral_search_mode = c("dda", "dia", "csa")
spectral_search_mode_option	Secondary type of analysis. spectral_search_mode_option = c(NA, "rawddaspectra", "alignedtable")

Value

A string of standard .msp file

IDSL.CSA_workflow *IDSL.CSA workflow*

Description

This function executes the CSA workflow.

Usage

```
IDSL.CSA_workflow(spreadsheet)
```

Arguments

spreadsheet	CSA spreadsheet

Value

This function organizes the IDSL.CSA file processing for better performance using the template spreadsheet.

`IDSL.CSA_xlsxAnalyzer` *IDSL.CSA workflow xlsx Analyzer*

Description

This function processes the spreadsheet of the CSA parameters to ensure the parameter inputs are consistent with the requirements of the IDSL.CSA pipeline.

Usage

```
IDSL.CSA_xlsxAnalyzer(spreadsheet)
```

Arguments

`spreadsheet`	'Start' tab of the parameter spreadsheet

Value

This function returns the CSA parameters to feed the 'IDSL.CSA_workflow' function.

`negativeAdducts` *negative adducts*

Description

This data consists of adducts and mass differences for common ionization pathways in negative modes.

Usage

```
data("negativeAdducts")
```

Format

A data frame on the following 2 variables.

`Adduct` a character vector

`massAdduct` a numerical vector

Examples

```
data(negativeAdducts)
```

positiveAdducts *positive adducts*

Description

This data consists of adducts and mass differences for common ionization pathways in positive modes.

Usage

```
data("positiveAdducts")
```

Format

A data frame on the following 2 variables.

`Adduct` a character vector

`massAdduct` a numerical vector

Examples

```
data(positiveAdducts)
```

Index

∗ **datasets**
CSA_PARAM_SPEC, 7
negativeAdducts, 18
positiveAdducts, 19

aligned_fragmentation_spectra_annotator, 2

CSA_adductAnnotator, 3
CSA_alignedMetaSpectraCataloger, 3
CSA_alignedPeaksTanimotoCoefficientCalculator, 4
CSA_AlignedTable_xlsxAnalyzer, 5
CSA_fragmentationPeakDetection, 5
CSA_PARAM_SPEC, 7
CSA_reference_xlsxAnalyzer, 7
CSA_workflow, 8
CSA_xlsxAnalyzer, 9

DDA2msp, 9
DDA_fragmentationPeakDetection, 10
DDA_rawSpectraDeconvolution, 11
DDA_workflow, 11
DDA_xlsxAnalyzer, 12
DIA_MS1_fragmentationPeakDetection, 13
DIA_MS2_fragmentationPeakDetection, 14
DIA_workflow, 15
DIA_xlsxAnalyzer, 16

IDSL.CSA_MSPgenerator, 16
IDSL.CSA_referenceMSPgenerator, 17
IDSL.CSA_workflow, 17
IDSL.CSA_xlsxAnalyzer, 18

negativeAdducts, 18

positiveAdducts, 19