

# Package ‘Aerith’

May 14, 2026

**Title** visualization and annotation of isotopic enrichment patterns of peptides and metabolites with stable isotope labeling from proteomics and metabolomics

**Version** 1.0.1

**Description** Visualisation of peptide isotopic peaks and SIP peptide spectra match (PSM). Filtration of high quality PSM. Accurate isotopic abundance calculation of peptide and metabolites. Visualisation of SIP proteomics results.

**Depends** R (>= 4.4.0)

**License** GPL-3

**Encoding** UTF-8

**ByteCompile** TRUE

**Suggests** knitr, rmarkdown, testthat, tidyr

**RoxygenNote** 7.3.3

**Roxygen** list(markdown = TRUE)

**LinkingTo** Rcpp

**Imports** mzR, MSnbase, methods, Rcpp, ggplot2, ggrepel, dplyr, stringr, data.table, scales

**SystemRequirements** C++17

**VignetteBuilder** knitr

**Config/build/vignettes** FALSE

**biocViews** Proteomics, Metabolomics, MassSpectrometry, Software, Visualization, QualityControl, Annotation

**URL** <https://github.com/xyz1396/Aerith>

**BugReports** <https://github.com/xyz1396/Aerith/issues>

**git\_url** <https://git.bioconductor.org/packages/Aerith>

**git\_branch** RELEASE\_3\_23

**git\_last\_commit** 35fbela

**git\_last\_commit\_date** 2026-05-01

**Repository** Bioconductor 3.23

**Date/Publication** 2026-05-14

**Author** Yi Xiong [aut, cre] (ORCID: <<https://orcid.org/0009-0005-4591-2366>>)

**Maintainer** Yi Xiong <yixiong@ou.edu>

## Contents

Aerith-package . . . . .	4
.onLoad . . . . .	4
.onUnload . . . . .	5
AAspectra-class . . . . .	5
annotatePrecursor . . . . .	6
annotatePSM . . . . .	7
BYion_peak_calculator_DIY . . . . .	8
calBYAtomCountAndBaseMass . . . . .	9
calPepAtomCount . . . . .	10
calPepNeutronMass . . . . .	10
calPepPrecursorMass . . . . .	11
cal_isotope_numbers . . . . .	11
cal_isotope_numbers_SIP . . . . .	12
cal_isotope_peaks_fft . . . . .	12
denoiseOneMS2ScanHasCharge . . . . .	13
extractPSMfeatures . . . . .	14
extractPSMfeaturesTargetAndDecoy . . . . .	15
extractPSMfeaturesTargetAndDecoytoPercolatorPin . . . . .	16
generateCFGs . . . . .	17
generateOneCFG . . . . .	18
getFilterThreshold . . . . .	19
getFilterThresholdTopPSMs . . . . .	19
getFilterThresholdTopPSMsSpe2Pep . . . . .	20
getMZ . . . . .	21
getPrecursorSpectra . . . . .	22
getRealScan . . . . .	22
getRealScanFromList . . . . .	23
getRealScans . . . . .	23
getRealScansWithCharges . . . . .	24
getRealScanWithCharge . . . . .	25
getRetentionTimeAndPrecursorInfo . . . . .	25
getSipBYionSpectra . . . . .	26
getSipPrecursorSpectra . . . . .	27
getTIC . . . . .	28
getUnfilteredPeptides . . . . .	28
getUnfilteredPSMs . . . . .	29
plot,AAspectra,missing-method . . . . .	29
plotFilteredPCTIntensitySummary . . . . .	30
plotMolecularFFTisotopes . . . . .	31
plotMolecularIsotopes . . . . .	32
plotPrecursorAnnotation . . . . .	33

plotPrecursorMzFrequency . . . . .	35
plotProSipPct . . . . .	35
plotPSMannotation . . . . .	36
plotPSMs . . . . .	37
plotPSMsipPCT . . . . .	38
plotRealScan . . . . .	39
plotScanFrequency . . . . .	40
plotScanFrequencyMS2 . . . . .	41
plotScoreDistribution . . . . .	41
plotSipBYionLabel . . . . .	42
plotSIPfilteredPCTIntensityBySample . . . . .	42
plotTIC . . . . .	43
precursor_peak_calculator . . . . .	44
precursor_peak_calculator_DIY . . . . .	44
precursor_peak_calculator_DIY_averagine . . . . .	45
rankyfy . . . . .	46
readAllScanMS1 . . . . .	46
readAllScanMS2 . . . . .	47
readFilesScansTopPSMs . . . . .	47
readFilesScansTopPSMsFromOneFT2 . . . . .	48
readFTheader . . . . .	49
readMgf . . . . .	49
readMzmlMS1 . . . . .	50
readMzmlMS2 . . . . .	50
readOneScanMS1 . . . . .	51
readOneScanMS2 . . . . .	52
readPepXMLtable . . . . .	52
readPSMtsv . . . . .	53
readScansMS1 . . . . .	53
readScansMS1Vector . . . . .	54
readScansMS2 . . . . .	55
readScansMS2Vector . . . . .	55
readSip . . . . .	56
readSips . . . . .	57
readSpe2Pep . . . . .	57
readSpe2PepFilesScansTopPSMs . . . . .	58
readSpe2PepFilesScansTopPSMsFromEachFT2Parallel . . . . .	59
readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParallel . . . . .	59
readSpe2PepFilesScansTopPSMsFromOneFT2 . . . . .	60
readSpe2Peps . . . . .	61
residue_peak_calculator_DIY . . . . .	62
scoreIntensity . . . . .	62
scoreIntensityByCE . . . . .	63
scorePSM . . . . .	64
scorePSMsimple . . . . .	64
summaryPSMsipPCT . . . . .	65
writeAllScanMS1 . . . . .	66
writeAllScanMS2 . . . . .	67

writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel . . . . . 68

**Index** . . . . . 69

Aerith-package                    *Pre and post processing of SIP proteomics data*

## Description

Visualisation of peptide isotopic peaks and SIP peptide spectra match (PSM). Filtration of high quality PSM. Accurate isotopic abundance calculation of peptide and metabolites. Visualisation of SIP proteomics results.

## Details

Aerith streamlines pre-processing, quality control, and post-processing steps for stable-isotope proteomics workflows, providing plotting helpers, scoring utilities, and tidy outputs suited for downstream analysis.

## Author(s)

**Maintainer:** Yi Xiong <yixiong@ou.edu> ([ORCID](#))

## See Also

Useful links:

- <https://github.com/xyz1396/Aerith>
- Report bugs at <https://github.com/xyz1396/Aerith/issues>

.onLoad                            *Package initialization*

## Description

Sets up environment variables and package configuration when Aerith is loaded.

## Usage

```
.onLoad(libname, pkgname)
```

## Arguments

libname	character string giving the library directory where the package defining the namespace was found.
pkgname	character string giving the name of the package.

**Value**

Called for its side effects. Returns NULL invisibly.

---

.onUnload                      *Package cleanup*

---

**Description**

Cleanup when package is unloaded.

**Usage**

.onUnload(libpath)

**Arguments**

libpath                      character string giving the complete path to the package.

**Value**

Called for its side effects.

---

AAspectra-class                      *AAspectra S4 class for annotated mass spectra*

---

**Description**

Unified container storing theoretical or observed spectra, their charge states, and the associated peptide or compound identifier.

**Details**

Instances of this class are typically created by helper constructors like `getPrecursorSpectra()`, `getSipPrecursorSpectra()`, or converted from raw scans with `getRealScan()`. The class underpins downstream plotting and annotation methods.

**Slots**

spectra A `data.frame` with columns such as Mass, MZ, Prob, Kind, and Charge, holding peaks and metadata.

charges Numeric vector of precursor charge states carried alongside spectra.

AAstr Character string containing the peptide sequence or compound label used to generate the spectrum.

**See Also**

[getPrecursorSpectra\(\)](#), [getSipPrecursorSpectra\(\)](#), [getSipBYionSpectra\(\)](#), [plot,AAspectra,missing-method](#)

**Examples**

```
AAstr <- "KHRIP"
spectra <- getPrecursorSpectra(AAstr, 1:2)
class(spectra)
```

---

annotatePrecursor      *annotatePrecursor*

---

**Description**

annotatePrecursor

**Usage**

```
annotatePrecursor(
  realMZ,
  realIntensity,
  realCharge,
  pepSeq,
  charge,
  Atom,
  Prob,
  isoCenter = 0,
  isoWidth = 0,
  calScores = FALSE
)
```

**Arguments**

realMZ	mz vector in precursor scan
realIntensity	intensity vector in precursor scan
realCharge	charge vector in precursor scan
pepSeq	a string of peptide
charge	charge of precursor ion in consideration
Atom	"C13" or "N15"
Prob	its SIP abundance (0.0~1.0)
isoCenter	isolation window center, set it 0 as default if not filtering by isolation window
isoWidth	isolation window width, set it 0 as default if not filtering by isolation window
calScores	FALSE as default, calculate matched spectra entropy score or not

**Value**

a List about matched precursor isotopic peaks information

**Examples**

```
realMZ <- c(
  894.9413, 895.4429, 895.9444, 896.3896, 896.4448, 896.9463,
  897.3890, 897.8896, 898.3930, 898.4734, 901.8851, 902.4465,
  902.9483, 903.4498, 903.9504, 910.8968, 911.4449, 912.3784
)
realIntensity <- c(
  16660537.0, 12344664.0, 6128400.5, 1448961.1, 1614148.1, 713238.8,
  1999402.4, 1124157.4, 567865.2, 647140.8, 709644.2, 7805729.0,
  8421993.0, 3200114.2, 1286055.5, 620246.8, 540861.6, 1079918.5
)
realCharge <- c(2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2)
anno <- annotatePrecursor(
  realMZ, realIntensity, realCharge,
  "GITINTSHVEYDTPTR", 2, "C13",
  0.01, 902.4471, 20, TRUE
)
```

---

annotatePSM

*annotatePSM*

---

**Description**

annotatePSM

**Usage**

```
annotatePSM(
  realMZ,
  realIntensity,
  realCharge,
  pepSeq,
  charges,
  Atom,
  Prob,
  isoCenter = 0,
  isoWidth = 0,
  calScores = FALSE
)
```

**Arguments**

realMZ            mz vector in MS2 scan  
 realIntensity    intensity vector in MS2 scan

realCharge	charge vector in MS2 scan
pepSeq	a string of peptide
charges	charges of product ions in consideration
Atom	"C13" or "N15"
Prob	its SIP abundance (0.0~1.0)
isoCenter	isolation window center, set it 0 as default if not remove peaks in isolation window
isoWidth	isolation window width, set it 0 as default if not remove peaks in isolation window
calScores	FALSE as default, calculate WDP MVH Xcor scores or not

**Value**

a List about matched peaks information of this PSM

**Examples**

```
demo_file <- system.file("extdata", "107728.FT2", package = "Aerith")
scan1 <- readOneScanMS2(ftFile = demo_file, 107728)
anno <- annotatePSM(
  scan1$peaks$mz, scan1$peaks$intensity,
  scan1$peaks$charge,
  "HSQVFSTAEDNQSAVTIHVLQGER", 1:2, "C13",
  0.0107, 886.65, 4.0, TRUE
)
```

---

BYion\_peak\_calculator\_DIY

*BY Ion Peak Calculator with User-Defined Isotopic Distribution*

---

**Description**

This function calculates the isotopic distribution of B and Y ions for a given amino acid string with a user-defined isotopic distribution and returns a DataFrame containing the mass, probability, and type of each ion.

**Usage**

```
BYion_peak_calculator_DIY(AAstr, Atom, Prob)
```

**Arguments**

AAstr	A string representing the amino acid sequence.
Atom	A string representing the isotope ("C13", "N15", "H2", "O18", "S34").
Prob	A double representing the abundance of the specified isotope (0.0 to 1.0).

**Value**

A DataFrame with three columns: "Mass" containing the mass of each ion, "Prob" containing the probability of each ion, and "Kind" indicating whether the ion is a B or Y ion.

**Examples**

```
# Example usage
df <- BYion_peak_calculator_DIY("PEPTIDE", "C13", 0.2)
df <- BYion_peak_calculator_DIY("PEPTIDE", "N15", 0.5)
```

---

calBYAtomCountAndBaseMass

*Simple calculator of C H O N P S atom count and mass without isotope of B Y ions*

---

**Description**

Simple calculator of C H O N P S atom count and mass without isotope of B Y ions

**Usage**

```
calBYAtomCountAndBaseMass(AAstrs)
```

**Arguments**

AAstrs            a CharacterVector of peptides

**Value**

a list of data.frame of C H O N P S atom count and each data.frame is for one peptide

**Examples**

```
peps <- calBYAtomCountAndBaseMass(c("HK~FL", "AD!CH", "~ILKMV"))
```

---

calPepAtomCount      *Simple calculator of C H O N P S atom count of peptide*

---

**Description**

Simple calculator of C H O N P S atom count of peptide

**Usage**

```
calPepAtomCount(AAstrs)
```

**Arguments**

AAstrs              a CharacterVector of peptides

**Value**

a dataframe of C H O N P S atom count each row is for one peptide

**Examples**

```
df <- calPepAtomCount(c("HKFL", "ADCH"))
```

---

calPepNeutronMass      *Simple calculator neutron mass by average delta mass of each isotope*

---

**Description**

Simple calculator neutron mass by average delta mass of each isotope

**Usage**

```
calPepNeutronMass(AAstrs, Atom, Probs)
```

**Arguments**

AAstrs              a CharacterVector of peptides  
Atom                a Character of "C13", "H2", "O18", "N15", or "S34"  
Probs                a NumericVector with the same length of AAstr for SIP abundances

**Value**

a vector of peptide neutron masses

**Examples**

```
masses <- calPepNeutronMass(c("HKFL", "ADCH"), "C13", c(0.2, 0.3))
```

---

calPepPrecursorMass     *Simple calculator of peptide precursor mass by binomial NP*

---

**Description**

Simple calculator of peptide precursor mass by binomial NP

**Usage**

```
calPepPrecursorMass(AAstrs, Atom, Probs)
```

**Arguments**

AAstrs            a CharacterVector of peptides  
Atom             a Character of "C13", "H2", "O18", "N15", or "S34"  
Probs            a NumericVector with the same length of AAstr for SIP abundances

**Value**

a vector of peptide precursor masses

**Examples**

```
masses <- calPepPrecursorMass(c("HKFL", "ADCH"), "C13", c(0.2, 0.3))
```

---

cal\_isotope\_numbers     *Calculate Isotope Numbers in natural abundance*

---

**Description**

This function calculates the isotope numbers for a given chemical formula using a Monte Carlo simulation approach.

**Usage**

```
cal_isotope_numbers(formula, num_simulations = 10000)
```

**Arguments**

formula            A character string representing the chemical formula.  
num\_simulations    An integer specifying the number of simulations to run. Default is 10,000.

**Value**

A data frame containing the results of the simulations.

**Examples**

```
cal_isotope_numbers("C6H12O6")
cal_isotope_numbers("CF3COOH", num_simulations = 5000)
```

---

```
cal_isotope_numbers_SIP
```

*Calculate Isotope Numbers for SIP*

---

**Description**

This function calculates the isotope numbers for Stable Isotope Probing (SIP) based on a given chemical formula using a Monte Carlo simulation approach.

**Usage**

```
cal_isotope_numbers_SIP(formula, num_simulations = 10000, ...)
```

**Arguments**

formula	A character string representing the chemical formula, "C6H12O6" for example.
num_simulations	An integer specifying the number of simulations to run. Default is 10,000.
...	Additional arguments passed to the function, C13=0.5 for example to set the abundance of C13 to 0.5.

**Value**

A dataframe containing the results of the isotope number and mass calculations.

**Examples**

```
cal_isotope_numbers_SIP("C6H12O6")
cal_isotope_numbers_SIP("C6H12O6", num_simulations = 10000, C13 = 0.5)
```

---

```
cal_isotope_peaks_fft
```

*Calculate Isotope Peaks using FFT*

---

**Description**

This function calculates the isotope peaks for a given chemical formula using Fast Fourier Transform (FFT). It approximates the delta mass of isotopes is one neutron mass and ignore the fine structure of isotopes.

**Usage**

```
cal_isotope_peaks_fft(formula, N_width = 100, min_abundance = 1e-04, ...)
```

**Arguments**

formula	A character string representing the chemical formula.
N_width	An integer specifying the width of the isotope envelope in FFT. Default is 100. Wider isotopic envelope will require larger N_width.
min_abundance	A numeric value specifying the minimum abundance threshold for the peaks. Default is 0.0001.
...	Additional arguments passed to the function. For example C13=0.5 will change the abundance of C13 to 0.5 and adjust the abundance of C12 accordingly.

**Value**

data.frame A data frame containing the calculated isotope peaks mass and their abundances.

**Examples**

```
# Example usage:
cal_isotope_peaks_fft("C6H12O6")
cal_isotope_peaks_fft("C6H12O6", N_width = 200, min_abundance = 0.001, C13 = 0.5)
```

---

```
denoiseOneMS2ScanHasCharge
      denoise one MS2 scan has charge
```

---

**Description**

denoise one MS2 scan has charge

**Usage**

```
denoiseOneMS2ScanHasCharge(scanList, window, step, threshold)
```

**Arguments**

scanList	a list of one MS2 scan has charge
window	a float of mz window size for denoise
step	a float of mz step for denoise
threshold	a float of top N threshold for denoise

**Value**

a denoised MS2 scan has charge

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
ft2 <- readAllScanMS2(demo_file)
plot(getRealScanFromList(ft2[["1346"]]))
ms2 <- denoiseOneMS2ScanHasCharge(ft2[["1346"]], 100, 10, 5)
plot(getRealScanFromList(ms2))
```

---

extractPSMfeatures     *extractPSMfeatures extract features of top PSMs from multiple .Spe2Pep.txt files*

---

**Description**

extractPSMfeatures extract features of top PSMs from multiple .Spe2Pep.txt files

**Usage**

```
extractPSMfeatures(Spe2PepFilePath, topN, ftFilepath, ThreadNumber = 3L)
```

**Arguments**

Spe2PepFilePath     a full path with .Spe2Pep.txt files in it

topN                 store top N PSMs of each scan of one .FT2 file

ftFilepath          a full path with .FT1 and .FT2 files in it

ThreadNumber        read ThreadNumber of FT file at the same time, it will increase ram usage

**Details**

Set OpenMP stack size to avoid stack overflow in parallel processing before loading Aerith package:  
 Sys.setenv(OMP\_STACKSIZE = "16M") Sys.setenv(OMP\_NUM\_THREADS = parallel::detectCores())

**Value**

A named list of data frames, each containing the extracted PSM features from the corresponding .Spe2Pep.txt file.

**Examples**

```
tmp <- tempdir()
target_dir <- file.path(tmp, "target")
dir.create(target_dir, showWarnings = FALSE)
target_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(target_file, file.path(target_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
ft_dir <- file.path(tmp, "ft")
dir.create(ft_dir, showWarnings = FALSE)
ft_file <- system.file("extdata", "demo_target_decoy.FT1.rds", package = "Aerith")
file_content <- readRDS(ft_file)
```

```
writeLines(file_content, file.path(ft_dir, "Pan_052322_X13.FT1"))
print(list.files(c(ft_dir, target_dir), full.names = TRUE, recursive = TRUE))
psm <- extractPSMfeatures(target_dir, 5, ft_dir, 3)
```

---

```
extractPSMfeaturesTargetAndDecoy
```

*extractPSMfeaturesTargetAndDecoy extract features of top PSMs from target and decoy .Spe2Pep.txt files*

---

## Description

extractPSMfeaturesTargetAndDecoy extract features of top PSMs from target and decoy .Spe2Pep.txt files

## Usage

```
extractPSMfeaturesTargetAndDecoy(
  targetPath,
  decoyPath,
  topN,
  ftFilepath,
  ThreadNumber = 3L
)
```

## Arguments

targetPath	a full path with target .Spe2PepFile.txt files in it
decoyPath	a full path with decoy .Spe2PepFile.txt files in it
topN	store top N PSMs of each scan of one .FT2 file
ftFilepath	a full path with .FT1 and .FT2 files in it
ThreadNumber	read ThreadNumber of FT file at the same time, it will increase ram usage

## Details

Set OpenMP stack size to avoid stack overflow in parallel processing before loading Aerith package:  
 Sys.setenv(OMP\_STACKSIZE = "16M") Sys.setenv(OMP\_NUM\_THREADS = parallel::detectCores())

## Value

the PSMs in a dataframe in a list

**Examples**

```

tmp <- tempdir()
target_dir <- file.path(tmp, "target")
dir.create(target_dir, showWarnings = FALSE)
target_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(target_file, file.path(target_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
decoy_dir <- file.path(tmp, "decoy")
dir.create(decoy_dir, showWarnings = FALSE)
decoy_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(decoy_file, file.path(decoy_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
ft_dir <- file.path(tmp, "ft")
dir.create(ft_dir, showWarnings = FALSE)
ft_file <- system.file("extdata", "demo_target_decoy.FT1.rds", package = "Aerith")
file_content <- readRDS(ft_file)
writeLines(file_content, file.path(ft_dir, "Pan_052322_X13.FT1"))
print(list.files(c(ft_dir, target_dir, decoy_dir), full.names = TRUE, recursive = TRUE))
psm <- extractPSMfeaturesTargetAndDecoy(target_dir, decoy_dir, 3, ft_dir, 3)

```

---

extractPSMfeaturesTargetAndDecoytoPercolatorPin

*extractPSMfeaturesTargetAndDecoytoPercolatorPin extract features of top PSMs from target and decoy .Spe2Pep.txt files to percolator pin format*

---

**Description**

extractPSMfeaturesTargetAndDecoytoPercolatorPin extract features of top PSMs from target and decoy .Spe2Pep.txt files to percolator pin format

**Usage**

```

extractPSMfeaturesTargetAndDecoytoPercolatorPin(
  targetPath,
  decoyPath,
  topN,
  ftFilepath,
  ThreadNumber = 3L,
  doProteinInference = FALSE,
  fileName = "a.pin"
)

```

**Arguments**

targetPath	a full path with target .Spe2PepFile.txt files in it
decoyPath	a full path with decoy .Spe2PepFile.txt files in it
topN	store top N PSMs of each scan of one .FT2 file
ftFilepath	a full path with .FT1 and .FT2 files in it

ThreadNumber read ThreadNumber of FT file at the same time, it will increase ram usage  
doProteinInference  
output put protein inference format or only PSM format  
fileName output path of the percolator tsv file

### Details

Set OpenMP stack size to avoid stack overflow in parallel processing before loading Aerith package:  
Sys.setenv(OMP\_STACKSIZE = "16M") Sys.setenv(OMP\_NUM\_THREADS = parallel::detectCores())

### Value

NULL (invisibly).

### Examples

```
tmp <- tempdir()
target_dir <- file.path(tmp, "target")
dir.create(target_dir, showWarnings = FALSE)
target_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(target_file, file.path(target_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
decoy_dir <- file.path(tmp, "decoy")
dir.create(decoy_dir, showWarnings = FALSE)
decoy_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(decoy_file, file.path(decoy_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
ft_dir <- file.path(tmp, "ft")
dir.create(ft_dir, showWarnings = FALSE)
ft_file <- system.file("extdata", "demo_target_decoy.FT1.rds", package = "Aerith")
file_content <- readRDS(ft_file)
writeLines(file_content, file.path(ft_dir, "Pan_052322_X13.FT1"))
pin_path <- file.path(tmp, "a.pin")
extractPSMfeaturesTargetAndDecoytoPercolatorPin(target_dir, decoy_dir, 3, ft_dir, 3, FALSE, pin_path)
print(list.files(c(ft_dir, target_dir, decoy_dir), full.names = TRUE, recursive = TRUE))
print(file.info(pin_path))
```

---

generateCFGs

*generateCFGs*

---

### Description

generateCFGs

### Usage

generateCFGs(cfgPath, outPath, element)

**Arguments**

cfgPath	a full path of .cfg file
outPath	a full path for .cfg file output
element	a string of element name, "N" for example

**Value**

a bool value if generate succeed or not

**Examples**

```
cfg <- system.file("extdata", "SiprosConfig.cfg", package = "Aerith")
tmp <- tempdir()
tmp <- file.path(tmp, "configs")
generateCFGs(cfg, tmp, "N")
list.files(tmp, full.names = TRUE)
```

---

generateOneCFG	<i>generateOneCFG</i>
----------------	-----------------------

---

**Description**

generateOneCFG

**Usage**

```
generateOneCFG(cfgPath, outPath, element, pct, center, width)
```

**Arguments**

cfgPath	a full path of .cfg file
outPath	a full path for .cfg file output
element	a string of element name, "N" for example
pct	a integer of element SIP abundance
center	a integer of mass window center
width	a integer of mass half window width

**Value**

a bool value if generate succeed or not

**Examples**

```
cfg <- system.file("extdata", "SiprosConfig.cfg", package = "Aerith")
tmp <- tempdir()
tmp <- file.path(tmp, "configs")
generateOneCFG(cfg, tmp, "N", 50, 0, 2)
list.files(tmp, full.names = TRUE)
```

---

```
getFilterThreshold    getFilterThreshold
```

---

**Description**

```
getFilterThreshold
```

**Usage**

```
getFilterThreshold(workingPath, OverallThreshold)
```

**Arguments**

```
workingPath    a full path with .sip files in it
OverallThreshold
                FDR threshold of peptides
```

**Value**

a dataframe about filter threshold and FDR results

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
getFilterThreshold(demo_dir, 0.01)
```

---

```
getFilterThresholdTopPSMs
                getFilterThresholdTopPSMs get filter threshold of top PSMs of each
                scan from multiple .sip file
```

---

**Description**

getFilterThresholdTopPSMs get filter threshold of top PSMs of each scan from multiple .sip file

**Usage**

```
getFilterThresholdTopPSMs(workingPath, OverallThreshold, topN)
```

**Arguments**

```
workingPath    a full path with .sip files in it
OverallThreshold
                FDR threshold of peptides
topN           store top N PSMs of each scan of one .FT file
```

**Value**

a dataframe about filter threshold and FDR results

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
re <- getFilterThresholdTopPSMs(demo_dir, 0.01, 3)
re$threshold
head(re$topPSMs)
```

---

`getFilterThresholdTopPSMsSpe2Pep`

*getFilterThresholdTopPSMsSpe2Pep* get filter threshold of top PSMs of each scan from multiple .sip file

---

**Description**

`getFilterThresholdTopPSMsSpe2Pep` get filter threshold of top PSMs of each scan from multiple .sip file

**Usage**

```
getFilterThresholdTopPSMsSpe2Pep(
  workingPath,
  OverallThreshold,
  topN,
  decoyPrefix
)
```

**Arguments**

<code>workingPath</code>	a full path with .Spe2Pep files in it
<code>OverallThreshold</code>	FDR threshold of peptides
<code>topN</code>	store top N PSMs of each scan of one .FT file
<code>decoyPrefix</code>	the prefix of decoy sequence

**Value**

a dataframe about filter threshold and FDR results, rows of ", 0, 0 ,0" means cannot find threshold at this charge

**Examples**

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
list.files(sip_dir, full.names = TRUE)
a <- getFilterThresholdTopPSMsSpe2Pep(sip_dir, 1, 3, "Decoy_")
a$threshold
```

---

getMZ

*add MZ to spectra data.frame*

---

**Description**

add MZ to spectra data.frame

**Usage**

```
getMZ(spectra, charges = c(1, 2))
```

**Arguments**

spectra	a dataframe of spectra
charges	ion charges

**Value**

a dataframe of spectra

**Examples**

```
spectra <- precursor_peak_calculator("KHRIP")
spectra <- getMZ(spectra, 1:2)
```

---

getPrecursorSpectra	<i>Get AAspectra object of precursor from AA sequence with natural SIP abundance</i>
---------------------	--

---

**Description**

Get AAspectra object of precursor from AA sequence with natural SIP abundance

**Usage**

```
getPrecursorSpectra(AAstr, charges = c(1, 2))
```

**Arguments**

AAstr	Amino acid string
charges	Integer vector of charges. Default is 1:2

**Value**

AAspectra object

**Examples**

```
getPrecursorSpectra("KHRIP", 1:2)
```

---

getRealScan	<i>Convert one scan with charges=1 normalized by highest peak in scans list of ft file to AAspectra class</i>
-------------	---

---

**Description**

Convert one scan with charges=1 normalized by highest peak in scans list of ft file to AAspectra class

**Usage**

```
getRealScan(scanNumber, ft)
```

**Arguments**

scanNumber	ScanNumber of one scan
ft	Scans list of ft file

**Value**

AAspectra object

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getRealScan(1388, a)
plot(b)
```

---

getRealScanFromList     *Convert one scan in list format to AAspectra class*

---

**Description**

Convert one scan in list format to AAspectra class

**Usage**

```
getRealScanFromList(scan)
```

**Arguments**

scan                    one scan in list format

**Value**

AAspectra object

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getRealScanFromList(a[[1]])
plot(b)
```

---

getRealScans             *Get real scans from a scans list of one FT file with charges converted to 1 and intensities normalized by the highest peak.*

---

**Description**

Get real scans from a scans list of one FT file with charges converted to 1 and intensities normalized by the highest peak.

**Usage**

```
getRealScans(ft, scanNumbers)
```

**Arguments**

`ft` A list of scans from one FT file.  
`scanNumbers` An integer vector of scan numbers of PSMs.

**Value**

A list of AAspectra objects representing the real scans.

**Examples**

```
scanNumbers <- c("2596", "8182")
demo_file <- system.file("extdata", "X13_4068_2596_8182.FT2", package = "Aerith")
ft2 <- readAllScanMS2(demo_file)
realScans <- getRealScans(ft2, scanNumbers)
```

---

`getRealScansWithCharges`

*get real scans with real charges and raw intensities from scans list of one ft file*

---

**Description**

get real scans with real charges and raw intensities from scans list of one ft file

**Usage**

```
getRealScansWithCharges(ft, scanNumbers)
```

**Arguments**

`ft` Scans list of one ft file  
`scanNumbers` Integer vector of scan number of PSMs

**Value**

List of AAspectra objects of real scans

**Examples**

```
scanNumbers <- c("2596", "8182")
demo_file <- system.file("extdata", "X13_4068_2596_8182.FT2", package = "Aerith")
ft2 <- readAllScanMS2(demo_file)
realScans <- getRealScansWithCharges(ft2, scanNumbers)
```

---

getRealScanWithCharge *Convert one scan in scans with real charges and raw intensities from list of scans of ft file to AAspectra class*

---

**Description**

Convert one scan in scans with real charges and raw intensities from list of scans of ft file to AAspectra class

**Usage**

```
getRealScanWithCharge(scanNumber, ft)
```

**Arguments**

scanNumber	Integer. The scan number of one scan.
ft	List. The list of scans from an ft file.

**Value**

AAspectra object containing the scan data.

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getRealScanWithCharge(1388, a)
head(slot(b, "spectra"))
```

---

getRetentionTimeAndPrecursorInfo

*get retention time and precursor mass from scans list of ft file*

---

**Description**

get retention time and precursor mass from scans list of ft file

**Usage**

```
getRetentionTimeAndPrecursorInfo(ft)
```

**Arguments**

ft	Scans list of ft file
----	-----------------------

**Value**

A data.frame of retention time and precursor mass

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getRetentionTimeAndPrecursorInfo(a)
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
a <- readAllScanMS1(demo_file)
b <- getRetentionTimeAndPrecursorInfo(a)
```

---

getSipBYionSpectra	<i>Get AAspectra object of B and Y ions from AA sequence with labeled SIP abundance</i>
--------------------	---

---

**Description**

Get AAspectra object of B and Y ions from AA sequence with labeled SIP abundance

**Usage**

```
getSipBYionSpectra(
  AAstr,
  Atom = "C13",
  Prob = 0.0107,
  charges = 1,
  precursorCharges = 2
)
```

**Arguments**

AAstr	Amino acide string
Atom	"C13" or "N15". Default is "C13"
Prob	C13 or N15's abundance. Default is 0.0107
charges	NumericVector of product ion's charge. Default is 1:2
precursorCharges	NumericVector of precursor's charge. Default is 2

**Value**

AAspectra object

**Examples**

```
# add precursor
a <- getSipBYionSpectra("KHRIPCDRK", "C13", 0.05, 1:2, 2)
tail(slot(a, "spectra"))
# not add precursor
a <- getSipBYionSpectra("KHRIPCDRK", "C13", 0.05, 1:2, 0)
tail(slot(a, "spectra"))
```

---

getSipPrecursorSpectra

*Get AAspectra object of precursor from AA sequence with labeled SIP abundance*

---

**Description**

Get AAspectra object of precursor from AA sequence with labeled SIP abundance

**Usage**

```
getSipPrecursorSpectra(AAstr, Atom = "C13", Prob = 0.0107, charges = c(1, 2))
```

**Arguments**

AAstr	Amino acide string
Atom	"C13" or "N15". Default is "C13"
Prob	C13 or N15's abundance. Default is 0.0107
charges	Integer vector of charges. Default is 1:2

**Value**

AAspectra object

**Examples**

```
getSipPrecursorSpectra("KHRIPCDRK", "C13", 0.05, 1:3)
```

---

getTIC *get TIC and retention time from scans list of ft file*

---

**Description**

TIC: total ion chromatogram.

**Usage**

```
getTIC(ft)
```

**Arguments**

ft Scans list of ft file

**Value**

A data.frame of TIC, relative TIC and retention time

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getTIC(a)
```

---

getUnfilteredPeptides *getUnfilteredPeptides*

---

**Description**

getUnfilteredPeptides

**Usage**

```
getUnfilteredPeptides(workingPath)
```

**Arguments**

workingPath a full path with .sip files in it

**Value**

a dataframe of unique peptides and whether it is decoy sequence

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
head(getUnfilteredPeptides(demo_dir))
```

---

```
getUnfilteredPSMs      getUnfilteredPSMs
```

---

**Description**

```
getUnfilteredPSMs
```

**Usage**

```
getUnfilteredPSMs(sipPath, ftPath, topN)
```

**Arguments**

```

sipPath      a full path with .sip files in it
ftPath       a full path with .ft files in it
topN         store top N PSMs of each scan of one .FT file

```

**Value**

```
data.frame of PSMs
```

**Examples**

```

demo_dir <- system.file("extdata", package = "Aerith")
head(getUnfilteredPSMs(demo_dir, demo_dir, 10))

```

---

```
plot, AAspectra, missing-method
```

*Plot an AAspectra object*

---

**Description**

S4 plot method producing a ggplot2 spectrum for an AAspectra object.

**Usage**

```

## S4 method for signature 'AAspectra,missing'
plot(x, y, linewidth = 0.1, ...)

```

**Arguments**

```

x           AAspectra object.
y           (ignored, must be missing)
linewidth   Numeric width of MS peaks. Default 0.1.
...         Passed on (currently unused).

```

**Value**

A ggplot2 object.

**See Also**

AASpectra

**Examples**

```
a <- getPrecursorSpectra("KHRIP", 2)
plot(a) +
  ggplot2::scale_x_continuous(breaks = seq(324, 329, by = 0.5)) +
  ggplot2::geom_linerange(linewidth = 0.2)
```

---

plotFilteredPCTIntensitySummary

*Plot decoy-filtered PSMs' PCT and intensity summary by each input file*

---

**Description**

This function reads all filtered PSM files in from output directory of Sipros 5, combines them, filters for labeled PSMs, extracts file names, computes log<sub>2</sub> intensities, and generates a hexbin plot faceted by each input file.

**Usage**

```
plotFilteredPCTIntensitySummary(
  psms_dir = "psms/",
  output_file = "decoy_filtered_PCT_and_intensity_summary_by_file.pdf",
  width = 16,
  height = 12
)
```

**Arguments**

psms_dir	Directory containing filtered PSM files (default: "psms/").
output_file	Output PDF file name for the plot.
width	Width of the output PDF (default: 16).
height	Height of the output PDF (default: 12).

**Value**

A ggplot2 object representing the hexbin plot.

---

`plotMolecularFFTisotopes`*Plot Molecular isotopes without fine structure by FFT algorithm*

---

## Description

This function plots the molecular isotopes generated by Fast Fourier Transform (FFT).

## Usage

```
plotMolecularFFTisotopes(  
  isotope_numbers,  
  charge = 1,  
  minProb = 1e-04,  
  yshift = -1,  
  peakWidth = 0.5,  
  textSize = 15  
)
```

## Arguments

<code>isotope_numbers</code>	A data.frame representing the isotope mass and abundance to be plotted.
<code>charge</code>	An integer representing the charge. Default is 1.
<code>minProb</code>	A numeric value representing the minimum probability. Default is 0.0001.
<code>yshift</code>	A numeric value representing the vertical shift applied to the plot for better visualization of the abundance close to 0. Default is -1.
<code>peakWidth</code>	A numeric value representing the width of the peaks in the plot. Default is 0.5.
<code>textSize</code>	A numeric value representing the size of the text in the plot.

## Value

A ggplot object of molecular isotopes without fine structure by FFT algorithm

## Examples

```
isotope_numbers <- cal_isotope_peaks_fft("C6H12O6", N_width = 200, min_abundance = 0.001, C13 = 0.5)  
plotMolecularFFTisotopes(isotope_numbers)
```

---

plotMolecularIsotopes *Plot Molecular Isotopes with fine structure by Montecarlo algorithm*

---

### Description

This function generates a plot of molecular isotopes based on the provided isotope numbers.

### Usage

```
plotMolecularIsotopes(  
  isotope_numbers,  
  charge = 1,  
  minProb = 1e-04,  
  jitterAmount = 0.03,  
  yshift = -1,  
  peakWidth = 0.5,  
  labelN = 35,  
  labelSize = 3,  
  textSize = 15  
)
```

### Arguments

isotope_numbers	A data.frame representing the isotope mass and abundance to be plotted.
charge	An integer representing the charge of the molecule. Default is 1.
minProb	A numeric value representing the minimum probability threshold for plotting. Default is 0.0001.
jitterAmount	A numeric value representing the amount of jitter to be added to the plot for better visualization of adjacent MZ. Default is 0.03.
yshift	A numeric value representing the vertical shift applied to the plot for better visualization of the abundance close to 0. Default is -1.
peakWidth	A numeric value representing the isotopic peak width in the plot. Default is 0.5.
labelN	An integer setting top N peaks to be annotated. Default is 35.
labelSize	A numeric value representing the size of the isotopologues labels. Default is 3.
textSize	A numeric value representing the size of the text in the plot. Default is 15.

### Value

ggplot A ggplot object of molecular isotopes generated by Montecarlo algorithm.

### Examples

```
isotope_numbers <- cal_isotope_numbers_SIP("C6H12O6", num_simulations = 10000, C13 = 0.5)  
plotMolecularIsotopes(isotope_numbers)
```

---

plotPrecursorAnnotation  
*plot precursor annotation*

---

## Description

plot precursor annotation

## Usage

```
plotPrecursorAnnotation(
  observedSpect,
  pep,
  charge,
  Atom,
  Prob,
  isoCenter = 0,
  isoWidth = 0,
  xwidth = 0,
  ifRemoveNotFoundIon = FALSE,
  ifShowPrecursorChargeAnnotation = TRUE,
  ifShowIsoCenter = TRUE,
  ifShowIsoWindow = TRUE,
  ifAdjustSIPabundance = TRUE,
  breakSize = 1/5,
  linewidth = 0.3
)
```

## Arguments

observedSpect	AAspectra object of precursor scan
pep	peptide sequence
charge	precursor charge in consideration
Atom	SIP labeled atom "13C" or "15N" for exmaple
Prob	its SIP abundance (0.0~1.0)
isoCenter	isolation window center. Defaults to the center of observedSpect when 0.
isoWidth	isolation window width. Defaults to the m/z span of observedSpect when 0.
xwidth	x-axis display width around isoCenter. Defaults to the m/z span of the observed and expected peaks when 0.
ifRemoveNotFoundIon	set it FALSE as default
ifShowPrecursorChargeAnnotation	Logical. If TRUE, show precursor charge annotation labels with ggrepel. Default TRUE.

<code>ifShowIsoCenter</code>	Logical. If TRUE, show <code>isoCenter</code> as a black dashed vertical reference line. Default TRUE.
<code>ifShowIsoWindow</code>	Logical. If TRUE, show grey dashed vertical lines at isolation window boundaries ( <code>isoCenter ± isoWidth/2</code> ). Default TRUE.
<code>ifAdjustSIPabundance</code>	Logical. If TRUE, estimate and re-fit SIP abundance from initial precursor annotation. Default TRUE.
<code>breakSize</code>	Numeric multiplier for isolation window to compute x-axis break step. Default is 1/5.
<code>linewidth</code>	Numeric width of peaks. Default 0.3.

**Value**

ggplot2 layer

**Examples**

```

realMZ <- c(
  894.9413, 895.4429, 895.9444, 896.3896, 896.4448, 896.9463,
  897.3890, 897.8896, 898.3930, 898.4734, 901.8851, 902.4465,
  902.9483, 903.4498, 903.9504, 910.8968, 911.4449, 912.3784
)
realIntensity <- c(
  16660537.0, 12344664.0, 6128400.5, 1448961.1, 1614148.1, 713238.8,
  1999402.4, 1124157.4, 567865.2, 647140.8, 709644.2, 7805729.0,
  8421993.0, 3200114.2, 1286055.5, 620246.8, 540861.6, 1079918.5
)
realCharge <- c(2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 0, 2)
scan <- list(
  peaks = data.frame(
    mz = realMZ,
    intensity = realIntensity,
    charge = realCharge
  ),
  precursorCharges = 2
)
observedSP <- getRealScanFromList(scan)
p <- plotPrecursorAnnotation(
  observedSpect = observedSP,
  pep = "GITINTSHVEYDTPTR", charge = 2,
  Atom = "C13", Prob = 0.01,
  isoCenter = 902.4471, isoWidth = 5.0, xwidth = 20.0,
  ifRemoveNotFoundIon = TRUE
)
p

```

---

`plotPrecursorMzFrequency`*Plot precursor MZ frequency per 5 per minute of MS2*

---

**Description**

Plot precursor MZ frequency per 5 per minute of MS2

**Usage**

```
plotPrecursorMzFrequency(  
  info,  
  timeBinWidth = 1,  
  x_breaks = seq(0, 200, by = 10)  
)
```

**Arguments**

<code>info</code>	A data.frame of retention time and precursor mass
<code>timeBinWidth</code>	A numeric value of retention time bin width. Default is 1
<code>x_breaks</code>	A numeric vector of breaks for x axis. Default is seq(0, 200, by = 10)

**Value**

A ggplot layer of scan frequency of MS2

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")  
a <- readAllScanMS2(demo_file)  
a <- getRetentionTimeAndPrecursorInfo(a)  
plotPrecursorMzFrequency(a, timeBinWidth = 0.1, x_breaks = seq(8, 11, by = 0.2))
```

---

`plotProSipPct`*plot the distribution of SIP percent of proteins*

---

**Description**

plot the distribution of SIP percent of proteins

**Usage**

```
plotProSipPct(proPath)
```

**Arguments**

proPath            a pro.cluster.txt file's path

**Value**

a ggplot2 obj

**Examples**

```
demo_file <- system.file("extdata", "demo.pro.cluster.txt", package = "Aerith")
p <- plotProSipPct(demo_file)
p
```

---

plotPSMannotation      *plot PSM annotation*

---

**Description**

plot PSM annotation

**Usage**

```
plotPSMannotation(
  observedSpect,
  pep,
  Atom,
  Prob,
  charges,
  isoCenter = 0,
  isoWidth = 0,
  ifRemoveNotFoundIon = FALSE
)
```

**Arguments**

observedSpect    AAspectra object of real scan

pep                peptide sequence

Atom              SIP labeled atom "13C" or "15N" for exmaple

Prob              its SIP abundance (0.0~1.0)

charges           charges numeric vector for B/Y ions in consideration

isoCenter         isolation window center, set it 0 as default if not remove peaks in isolation window

isoWidth          isolation window width, set it 0 as default if not remove peaks in isolation window

ifRemoveNotFoundIon    set it False as default

**Value**

ggplot2 layer

**Examples**

```
demo_file <- system.file("extdata", "107728.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
a <- getRealScan("107728", a)
p <- plotPSMannotation(
  observedSpect = a,
  pep = "HSQVFSTAEDNQSAVTIHVLQGER", Atom = "C13", Prob = 0.01,
  charges = c(2), isoCenter = 886.65, isoWidth = 4.0,
  ifRemoveNotFoundIon = TRUE
)
p
```

---

plotPSMs

*plot PSMs from FT2 files and PSM results*


---

**Description**

plot PSMs from FT2 files and PSM results

**Usage**

```
plotPSMs(
  realScans,
  charges,
  Atom = "C13",
  Probs,
  BYcharge = c(1, 2),
  ftFileNames,
  scanNumbers,
  pepSeqs,
  proNames,
  path = "."
)
```

**Arguments**

realScans	List of AAspectra objects of real scan
charges	Integer vector of precursor charge of PSMs
Atom	"C13" or "N15". Default is "C13"
Probs	C13 or N15's abundance of PSMs
BYcharge	Integer vector of B Y ion. Default is 1:2
ftFileNames	Character vector of FT2 file names

scanNumbers	Integer vector of scan number of PSMs
pepSeqs	Character vector of peptide sequence of PSMs
proNames	Character vector of protein name of PSMs
path	Output path of pdf. Default is "."

**Value**

PDF files saved in the specified path

**Examples**

```

element <- "C13"
demo_file <- system.file("extdata", "demo.psm.txt", package = "Aerith")
psm <- readPSMtsv(demo_file)
psm <- psm[psm$Filename == "Pan_052322_X13.FT2", ]
psm <- psm[psm$ScanNumber %in% c("4068", "2596", "8182"), ]
demo_file <- system.file("extdata", "X13_4068_2596_8182.FT2", package = "Aerith")
ft2 <- readAllScanMS2(demo_file)
ftFileNames <- psm$Filename
scanNumbers <- psm$ScanNumber
proNames <- psm$ProteinNames
charges <- psm$ParentCharge
pep <- psm$OriginalPeptide
pep <- stringr::str_sub(pep, 2, -2)
pct <- psm$SearchName
pct <- as.numeric(stringr::str_sub(
  stringr::str_split(pct, "_", simplify = TRUE)[, 2], 1, -4
)) / 100 / 1000
realScans <- getRealScans(ft2, scanNumbers)
tmp <- tempdir()
plotPSMs(
  realScans,
  charges,
  element,
  pct,
  BYcharge = 1:2,
  ftFileNames,
  scanNumbers,
  pep,
  proNames,
  path = tmp
)
list.files(tmp, pattern = ".pdf", full.names = TRUE)

```

**Description**

This function reads a PSM file, processes the SIP percent values, and generates a histogram with a dashed vertical line indicating the median.

**Usage**

```
plotPSMsipPCT(psmPath)
```

**Arguments**

psmPath            A character string specifying the path to the PSM file.

**Value**

A ggplot2 object representing the histogram of SIP percent values.

**Examples**

```
demo_file <- system.file("extdata", "demo.psm.txt", package = "Aerith")
p <- plotPSMsipPCT(demo_file)
p
```

---

plotRealScan            *plot real scan layer under the B Y ion peaks*

---

**Description**

plot real scan layer under the B Y ion peaks

**Usage**

```
plotRealScan(spect, linewidth = 0.1)
```

**Arguments**

spect            AAspectra object of real scan  
linewidth

**Value**

ggplot2 layer

**Examples**

```

a <- getSipBYionSpectra("HSQVFSTAEDNQSAVTIHVLQGER", "C13", 0.01, 1:2)
p <- plot(a)
p <- p + plotSipBYionLabel(a)
demo_file <- system.file("extdata", "107728.FT2", package = "Aerith")
b <- readAllScanMS2(demo_file)
c <- getRealScan(107728, b)
p <- p + plotRealScan(c)
p

```

---

plotScanFrequency	<i>Plot scan frequency</i>
-------------------	----------------------------

---

**Description**

Plot scan frequency

**Usage**

```
plotScanFrequency(info, binwidth = 1, breaks = seq(0, 200, by = 10))
```

**Arguments**

info	A data.frame of retention time and precursor mass
binwidth	A numeric value of bin width. Default is 1
breaks	A numeric vector of breaks for x axis. Default is seq(0, 200, by = 10)

**Value**

A ggplot of scan frequency per minute

**Examples**

```

demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
b <- getRetentionTimeAndPrecursorInfo(a)
plotScanFrequency(b, binwidth = 0.1, breaks = seq(9, 10, by = 0.2))
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
a <- readAllScanMS1(demo_file)
b <- getRetentionTimeAndPrecursorInfo(a)
plotScanFrequency(b, binwidth = 0.1, breaks = seq(9, 10, by = 0.2))

```

---

plotScanFrequencyMS2 *Plot scan frequency layer of MS2*

---

**Description**

Plot scan frequency layer of MS2

**Usage**

```
plotScanFrequencyMS2(info, binwidth = 1)
```

**Arguments**

info	A data.frame of retention time and precursor mass
binwidth	A numeric value of bin width. Default is 1

**Value**

A ggplot layer of scan frequency of MS2

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
a <- readAllScanMS2(demo_file)
a <- getRetentionTimeAndPrecursorInfo(a)
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
b <- readAllScanMS1(demo_file)
b <- getRetentionTimeAndPrecursorInfo(b)
plotScanFrequency(a, binwidth = 0.1, breaks = seq(9, 10, by = 0.2)) + plotScanFrequencyMS2(b, binwidth = 0.1)
```

---

plotScoreDistribution *plot the score of PSMs at different charge and mass error*

---

**Description**

plot the score of PSMs at different charge and mass error

**Usage**

```
plotScoreDistribution(sipFile)
```

**Arguments**

sipFile	a .sip file's path
---------	--------------------

**Value**

a ggplot2 obj

**Examples**

```
sipFile <- system.file("extdata", "demo.sip", package = "Aerith")
plotScoreDistribution(sipFile)
```

---

plotSipBYionLabel      *Draw AAspectra MS plot with B Y ion Labels*

---

**Description**

Draw AAspectra MS plot with B Y ion Labels

**Usage**

```
plotSipBYionLabel(spect)
```

**Arguments**

spect                  AAspectra object of B Y ions

**Value**

ggplot2 layer

**Examples**

```
a <- getSipBYionSpectra("KHRIPCDRK", "C13", 0.05, 1:2)
p <- plot(a)
p + plotSipBYionLabel(a)
```

---

plotSIPfilteredPCTIntensityBySample  
*Plot SIP-filtered PCT and intensity summary by each input file*

---

**Description**

This function reads a SIP filtered PSM file, processes the data, and generates a hexbin plot of log<sub>2</sub> precursor intensity vs. MS1 isotopic abundances, faceted by each input file.

**Usage**

```
plotSIPfilteredPCTIntensityBySample(  
  psm_file = "SIP_filtered_psms.tsv",  
  output_file = "SIP_filtered_PCT_and_intensity_summary_by_sample.pdf",  
  width = 16,  
  height = 12  
)
```

**Arguments**

psm_file	Path to the filtered PSM file (e.g., "SIP_filtered_psms.tsv").
output_file	Output PDF file name for the plot.
width	Width of the output PDF (default: 16).
height	Height of the output PDF (default: 12).

**Value**

A ggplot2 object representing the hexbin plot.

---

plotTIC	<i>Plot TIC of MS1 or MS2</i>
---------	-------------------------------

---

**Description**

Plot TIC of MS1 or MS2

**Usage**

```
plotTIC(tic, breaks = seq(0, 200, by = 10))
```

**Arguments**

tic	A data.frame of TIC and retention time
breaks	A vector of breaks for x axis

**Value**

a ggplot2 object

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")  
a <- readAllScanMS2(demo_file)  
b <- getTIC(a)  
plotTIC(b, seq(9, 10, by = 0.2))
```

precursor\_peak\_calculator

*Precursor Peak Calculator*

---

### Description

This function calculates the isotopic distribution of a given amino acid string and returns a DataFrame containing the mass and probability of each isotope.

### Usage

```
precursor_peak_calculator(AAstr)
```

### Arguments

AAstr            A string representing the amino acid sequence.

### Value

A DataFrame with two columns: "Mass" containing the mass of each isotope and "Prob" containing the probability of each isotope.

### Examples

```
a <- precursor_peak_calculator("PEPTIDE")
```

---

precursor\_peak\_calculator\_DIY

*Precursor Peak Calculator with User-Defined Isotopic Distribution*

---

### Description

This function calculates the isotopic distribution of a given amino acid string with a user-defined isotopic distribution and returns a DataFrame containing the mass and probability of each isotope.

### Usage

```
precursor_peak_calculator_DIY(AAstr, Atom, Prob)
```

### Arguments

AAstr            A string representing the amino acid sequence.

Atom            A string representing the isotope ("C13", "N15", "H2", "O18", "S34").

Prob            A double representing the abundance of the specified isotope (0.0 to 1.0).

**Value**

A DataFrame with two columns: "Mass" containing the mass of each isotope and "Prob" containing the probability of each isotope.

**Examples**

```
# Example usage
df <- precursor_peak_calculator_DIY("PEPTIDE", "C13", 0.2)
df <- precursor_peak_calculator_DIY("PEPTIDE", "N15", 0.5)
```

---

precursor\_peak\_calculator\_DIY\_averagine

*Simple peak calculator of user defined isotopic distribution of one peptide by averagine*

---

**Description**

Simple peak calculator of user defined isotopic distribution of one peptide by averagine

**Usage**

```
precursor_peak_calculator_DIY_averagine(AAstrs, Atom, Prob)
```

**Arguments**

AAstrs	a CharacterVector of peptides
Atom	a CharacterVector C13 or N15
Prob	a NumericVector for its abundance

**Value**

a list of DataFrames of spectra

**Examples**

```
demoSpectra <- precursor_peak_calculator_DIY_averagine(c("PEPTIDE", "ACDEFGHIK"), "C13", 0.25)
demoSpectra[[1]]
```

---

rankyfify                      *rankify numeric vector via ftFileWriter*

---

**Description**

rankify numeric vector via ftFileWriter

**Usage**

```
rankyfify(a)
```

**Arguments**

a                      A numeric vector whose values will be rank-transformed.

**Value**

A numeric vector containing the ranks of the input values.

**Examples**

```
demo_vec <- c(12.5, 3.2, 7.7, 3.2)
rankyfify(demo_vec)
```

---

readAllScanMS1                      *read MS1 scans with scanNumber as index*

---

**Description**

read MS1 scans with scanNumber as index

**Usage**

```
readAllScanMS1(ftFile)
```

**Arguments**

ftFile                      a ft1 file's full path

**Value**

a list of MS1 scans with names of scan number

**Examples**

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
ft1 <- readAllScanMS1(demo_file)
```

---

readAllScanMS2      *read MS2 scans with scanNumber as index*

---

**Description**

read MS2 scans with scanNumber as index

**Usage**

```
readAllScanMS2(ftFile)
```

**Arguments**

ftFile              a ft2 file's full path

**Value**

a list of MS2 scans with names of scan number

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
ft2 <- readAllScanMS2(demo_file)
```

---

readFilesScansTopPSMs      *readFilesScansTopPSMs*

---

**Description**

Aggregate the top-ranked peptide-spectrum matches (PSMs) from each scan across all .sip files found in a directory.

**Usage**

```
readFilesScansTopPSMs(workingPath, topN)
```

**Arguments**

workingPath      Character vector of length one giving the directory that contains .sip files.  
topN              Integer specifying how many top-ranked PSMs per scan to retain for each .sip file.

**Details**

This helper constructs an internal sipFileReader, loads every .sip file located in the supplied path, and extracts the best topN PSMs per scan. The result is returned as a single data.frame with file identifiers, scan numbers, charge states, mass measurements, scores, peptide assignments, and protein annotations.

**Value**

An R data.frame with one row per retained PSM and the columns fileNames, scanNumbers, parentCharges, measuredParentMasses, calculatedParentMasses, searchNames, scores, identifiedPeptides, originalPeptides, and proteinNames.

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
re <- readFilesScansTopPSMs(demo_dir, 10)
head(re)
```

---

readFilesScansTopPSMsFromOneFT2

*readFilesScansTopPSMsFromOneFT2 read each scan's top PSMs from multiple .sip files of one .FT2 file*

---

**Description**

readFilesScansTopPSMsFromOneFT2 read each scan's top PSMs from multiple .sip files of one .FT2 file

**Usage**

```
readFilesScansTopPSMsFromOneFT2(workingPath, pattern, topN)
```

**Arguments**

workingPath	a full path with .sip files in it
pattern	a regex pattern of the .FT2 file
topN	store top N PSMs of each scan of one .FT2 file

**Value**

a dataframe of top N PSMs

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
re <- readFilesScansTopPSMsFromOneFT2(demo_dir, ".*demo.*", 3)
head(re)
```

---

readFTheader	<i>read FT file header</i>
--------------	----------------------------

---

**Description**

read FT file header

**Usage**

```
readFTheader(ftFile)
```

**Arguments**

ftFile            a ft1 file's full path

**Value**

a list of ft file header

**Examples**

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
header <- readFTheader(demo_file)
```

---

readMgf	<i>Read spectra from .mgf file</i>
---------	------------------------------------

---

**Description**

Read spectra from .mgf file

**Usage**

```
readMgf(mgf)
```

**Arguments**

mgf            A .mgf file's path

**Value**

A list of spectra with names of scan number

**Examples**

```
# MSnbase can be installed from bioconductor
# library(MSnbase)
demo_file <- system.file("extdata", "demo.mgf", package = "Aerith")
a <- readMgf(demo_file)
```

---

**readMzmlMS1***Read MS1 spectra from .mzML file*

---

**Description**

Read MS1 spectra from .mzML file

**Usage**

```
readMzmlMS1(ms)
```

**Arguments**

ms                    A .mzML files's path

**Value**

A list of MS1 scans with names of scan number

**Examples**

```
# mzR can be installed from bioconductor
# library(mzR)
demo_file <- system.file("extdata", "demo.mzML", package = "Aerith")
a <- readMzmlMS1(demo_file)
```

---

**readMzmlMS2***Read MS2 spectra from .mzML file*

---

**Description**

Read MS2 spectra from .mzML file

**Usage**

```
readMzmlMS2(ms)
```

**Arguments**

ms                    A .mzML files's path

**Value**

A list of MS2 scans with names of scan number

**Examples**

```
# mzR can be installed from bioconductor
# library(mzR)
demo_file <- system.file("extdata", "demo.mzML", package = "Aerith")
a <- readMzmlMS2(demo_file)
```

---

readOneScanMS1	<i>readOneScanMS1</i>
----------------	-----------------------

---

**Description**

readOneScanMS1

**Usage**

```
readOneScanMS1(ftFile, scanNumber)
```

**Arguments**

ftFile	a ft1 file's full path
scanNumber	the scan at scanNumber

**Value**

a list of MS1 scan

**Examples**

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
ft1 <- readOneScanMS1(demo_file, 1588)
```

readOneScanMS2      *readOneScanMS2*

---

**Description**

readOneScanMS2

**Usage**

```
readOneScanMS2(ftFile, scanNumber)
```

**Arguments**

ftFile            a ft2 file's full path  
scanNumber        the scan at scanNumber

**Value**

a list of MS2 scan

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")  
ft2 <- readOneScanMS2(demo_file, 1633)
```

---

readPepXMLtable      *Read PSM table from .pepXML file*

---

**Description**

Read PSM table from .pepXML file

**Usage**

```
readPepXMLtable(pepXML)
```

**Arguments**

pepXML            A .pepXML files's path

**Value**

A dataframe of psm table

**Examples**

```
# mzR can be installed from bioconductor
# library(mzR)
demo_file <- system.file("extdata", "demo.pepXML", package = "Aerith")
a <- readPepXMLtable(demo_file)
```

---

readPSMtsv	<i>Read PSM TSV File</i>
------------	--------------------------

---

**Description**

This function reads a Peptide-Spectrum Match (PSM) file in TSV (Tab-Separated Values) format.

**Usage**

```
readPSMtsv(tsv)
```

**Arguments**

tsv                    A character string specifying the path to the PSM TSV file.

**Value**

A data frame containing the data from the PSM TSV file.

**Examples**

```
demo_file <- system.file("extdata", "demo.psm.txt", package = "Aerith")
a <- readPSMtsv(demo_file)
```

---

readScansMS1	<i>read MS1 scans with scanNumber as index in a range</i>
--------------	---

---

**Description**

read MS1 scans with scanNumber as index in a range

**Usage**

```
readScansMS1(ftFile, startScanNumber, endScanNumber)
```

**Arguments**

ftFile                a ft1 file's full path  
startScanNumber      read scans starting from this scanNumber  
endScanNumber        read scans ending at this scanNumber

**Value**

a list of MS1 scans with names of scan number

**Examples**

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
ft1 <- readScansMS1(demo_file, 1398, 1503)
```

---

readScansMS1Vector	<i>read MS1 scans with scanNumber as index in a vector</i>
--------------------	--

---

**Description**

read MS1 scans with scanNumber as index in a vector

**Usage**

```
readScansMS1Vector(ftFile, scanNumbersVector)
```

**Arguments**

ftFile            a ft1 file's full path  
scanNumbersVector    a NumericVector of scan numbers

**Value**

A named list of MS1 scans with names of scan number. The names of the list correspond to the scan numbers.

**Examples**

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
ft1 <- readScansMS1Vector(demo_file, c(1398, 1503, 1508))
```

---

readScansMS2	<i>read MS2 scans with scanNumber as index in a range</i>
--------------	---

---

**Description**

read MS2 scans with scanNumber as index in a range

**Usage**

```
readScansMS2(ftFile, startScanNumber, endScanNumber)
```

**Arguments**

ftFile	a ft2 file's full path
startScanNumber	read scans starting from this scanNumber
endScanNumber	read scans ending at this scanNumber

**Value**

a list of MS2 scans with names of scan number

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")  
ft2 <- readScansMS2(demo_file, 1350, 1355)
```

---

readScansMS2Vector	<i>read MS2 scans with scanNumber as index in a vector</i>
--------------------	--

---

**Description**

read MS2 scans with scanNumber as index in a vector

**Usage**

```
readScansMS2Vector(ftFile, scanNumbersVector)
```

**Arguments**

ftFile	a ft2 file's full path
scanNumbersVector	read scans starting of these scanNumbers

**Value**

a list of MS2 scans with names of scan number

**Examples**

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
ft2 <- readScansMS2Vector(demo_file, c(1350, 1355, 1359))
```

---

readSip	<i>readSip</i>
---------	----------------

---

**Description**

Read a single .sip file and convert its peptide-spectrum matches (PSMs) into an R list.

**Usage**

```
readSip(sipFile)
```

**Arguments**

`sipFile` A character vector of length one containing the full path to a .sip file.

**Details**

The reader parses the provided .sip file and returns metadata together with a data.frame of PSM-level attributes.

**Value**

An R list with file-level metadata (`fileName`, `scanType`, `searchName`, `scoringFunction`) and a PSM data frame containing scan numbers, charges, masses, scores, ranks, peptides, and protein names.

**Examples**

```
demo_file <- system.file("extdata", "demo.sip", package = "Aerith")
re <- readSip(demo_file)
head(re$PSM)
```

---

readSips	<i>readSips</i>
----------	-----------------

---

**Description**

Read every .sip file found in the provided directory and transform each file's peptide-spectrum matches into an R list entry.

**Usage**

```
readSips(workingPath)
```

**Arguments**

`workingPath` Character vector of length one giving the directory that contains .sip files.

**Details**

This function constructs a `sipFileReader` for the supplied folder, loads all .sip files, and for each file returns a list containing metadata and a `data.frame` of peptide-spectrum matches (PSMs). The resulting object is an R list whose elements correspond to individual input files.

**Value**

An R list; each element represents one .sip file and provides file-level descriptors (`fileName`, `scanType`, `searchName`, `scoringFunction`) together with a PSM data frame holding scan numbers, precursor charges, observed/calculated masses, scores, ranks, peptide sequences, and protein assignments.

**Examples**

```
demo_dir <- system.file("extdata", package = "Aerith")
re <- readSips(demo_dir)
head(re[[1]]$PSM)
```

---

readSpe2Pep	<i>readSpe2Pep</i>
-------------	--------------------

---

**Description**

```
readSpe2Pep
```

**Usage**

```
readSpe2Pep(Spe2PepFile)
```

**Arguments**

Spe2PepFile     a .Spe2PepFile file's full path

**Value**

the PSMs in a dataframe in a list

**Examples**

```
target_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
psm <- readSpe2Pep(target_file)
psm <- psm$PSM
```

---

readSpe2PepFilesScansTopPSMs

*readSpe2PepFilesScansTopPSMs read each scan's top PSMs from multiple .Spe2Pep.txt files*

---

**Description**

readSpe2PepFilesScansTopPSMs read each scan's top PSMs from multiple .Spe2Pep.txt files

**Usage**

```
readSpe2PepFilesScansTopPSMs(workingPath, topN = 5L)
```

**Arguments**

workingPath     a full path with .Spe2Pep.txt files in it  
topN             store top N PSMs of each scan of one .FT2 file

**Value**

the PSMs in a dataframe in a list

**Examples**

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
list.files(sip_dir, full.names = TRUE)
psm <- readSpe2PepFilesScansTopPSMs(sip_dir, 3)
```

---

```
readSpe2PepFilesScansTopPSMsFromEachFT2Parallel
    readSpe2PepFilesScansTopPSMsFromEachFT2Parallel read each
    scan's top PSMs from multiple .Spe2PepFile.txt files of each .FT2 file
```

---

**Description**

readSpe2PepFilesScansTopPSMsFromEachFT2Parallel read each scan's top PSMs from multiple .Spe2PepFile.txt files of each .FT2 file

**Usage**

```
readSpe2PepFilesScansTopPSMsFromEachFT2Parallel(workingPath, topN = 5L)
```

**Arguments**

workingPath      a full path with .Spe2PepFile.txt files in it  
topN                store top N PSMs of each scan of one .FT2 file

**Value**

a dataframe of top N PSMs

**Examples**

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
list.files(sip_dir, full.names = TRUE)
top3 <- readSpe2PepFilesScansTopPSMsFromEachFT2Parallel(sip_dir, 3)
```

---

```
readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParallel
    readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParalle
    read each scan's top PSMs from multiple .Spe2PepFile.txt files of each
    .FT2 file
```

---

**Description**

readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParalle read each scan's top PSMs from multiple .Spe2PepFile.txt files of each .FT2 file

**Usage**

```
readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParallel(
  targetPath,
  decoyPath,
  topN = 5L
)
```

**Arguments**

targetPath	a full path with target .Spe2PepFile.txt files in it
decoyPath	a full path with decoy .Spe2PepFile.txt files in it
topN	store top N PSMs of each scan of one .FT2 file

**Value**

a dataframe of top N PSMs

**Examples**

```
tmp <- tempdir()
target_dir <- file.path(tmp, "target")
dir.create(target_dir, showWarnings = FALSE)
target_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(target_file, file.path(target_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
decoy_dir <- file.path(tmp, "decoy")
dir.create(decoy_dir, showWarnings = FALSE)
decoy_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(decoy_file, file.path(decoy_dir, "Pan_052322_X13.SIP_C13_050_000Pct.Spe2Pep.txt"))
list.files(target_dir, full.names = TRUE)
list.files(decoy_dir, full.names = TRUE)
top3 <- readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParallel(target_dir, decoy_dir, 3)
```

---

```
readSpe2PepFilesScansTopPSMsFromOneFT2
```

*readSpe2PepFilesScansTopPSMsFromOneFT2 read each scan's top PSMs from multiple .Spe2PepFile.txt files of one .FT2 file*

---

**Description**

readSpe2PepFilesScansTopPSMsFromOneFT2 read each scan's top PSMs from multiple .Spe2PepFile.txt files of one .FT2 file

**Usage**

```
readSpe2PepFilesScansTopPSMsFromOneFT2(workingPath, pattern, topN = 5L)
```

**Arguments**

workingPath     a full path with .Spe2PepFile.txt files in it  
 pattern         a regex pattern of the .FT2 file  
 topN            store top N PSMs of each scan of one .FT2 file

**Value**

a dataframe of top N PSMs

**Examples**

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
list.files(sip_dir, full.names = TRUE)
top3 <- readSpe2PepFilesScansTopPSMsFromOneFT2(sip_dir, ".*X13.*", 3)
```

---

readSpe2Peps	<i>readSpe2Peps</i>
--------------	---------------------

---

**Description**

readSpe2Peps

**Usage**

```
readSpe2Peps(workingPath)
```

**Arguments**

workingPath     a full path with .Spe2Pep.txt files in it

**Value**

the PSMs dataframes in lists

**Examples**

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
```

```
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
list.files(sip_dir, full.names = TRUE)
psm <- readSpe2Peps(sip_dir)
psm <- psm[[1]]$PSM
```

---

```
residue_peak_calculator_DIY
```

*Simple residue peak calculator of user defined isotopic distribution of one residue*

---

### Description

Simple residue peak calculator of user defined isotopic distribution of one residue

### Usage

```
residue_peak_calculator_DIY(residue, Atom, Prob)
```

### Arguments

residue	residue name
Atom	isotopes of "C13", "N15", "H2", "O18", "S34"
Prob	its SIP abundance (0.0~1.0)

### Value

A DataFrame with two columns: "Mass" containing the mass of each isotope and "Prob" containing the probability of each isotope.

### Examples

```
df <- residue_peak_calculator_DIY("A", "C13", 0.2)
```

---

```
scoreIntensity
```

*scoreIntensity*

---

### Description

```
scoreIntensity
```

### Usage

```
scoreIntensity(observed, realIntensity, expectedIntensity, Atom, Prob)
```

**Arguments**

observed	this peak is observed or not
realIntensity	real intensity in MS2 scan
expectedIntensity	expected intensity
Atom	"C13" or "N15"
Prob	its SIP abundance (0.0~1.0)

**Value**

a score of this intensity match

**Examples**

```
scoreIntensity(TRUE, 1200.0, 1180.0, "C13", 0.02)
```

---

scoreIntensityByCE     *scoreIntensityByCrossEntropy*

---

**Description**

scoreIntensityByCrossEntropy

**Usage**

```
scoreIntensityByCE(expectedIntensity, observedIntensity)
```

**Arguments**

expectedIntensity	expected intensity
observedIntensity	observed intensity in MS2 scan

**Value**

numeric, a score of this intensity match

**Examples**

```
scoreIntensityByCE(c(10.0, 20.0, 30.0), c(9.5, 21.0, 28.0))
```

---

 scorePSM

*scorePSM*


---

**Description**

scorePSM

**Usage**

```
scorePSM(realMZ, realIntensity, realCharge, parentCharge, pepSeq, Atom, Prob)
```

**Arguments**

realMZ	mz vector in MS2 scan
realIntensity	intensity vector in MS2 scan
realCharge	charge vector in MS2 scan
parentCharge	int parent charge of MS2 scan
pepSeq	a string of peptide
Atom	"C13" or "N15"
Prob	its SIP abundance (0.0~1.0)

**Value**

a score of this PSM

**Examples**

```
demo_file <- system.file("extdata", "107728.FT2", package = "Aerith")
scan1 <- readOneScanMS2(ftFile = demo_file, 107728)
score <- scorePSM(scan1$peaks$mz,
  scan1$peaks$intensity, scan1$peaks$charge, 2,
  "[HSQVFSTAEDNQSAVTIHVLQGER]", "C13", 0.0107)
```

---

 scorePSMsimple

*scorePSMsimple* Score a PSM without isotopic envelope shape modeling
 

---

**Description**

scorePSMsimple Score a PSM without isotopic envelope shape modeling

**Usage**

```
scorePSMsimple(
  realMZ,
  realIntensity,
  realCharge,
  parentCharge,
  pepSeq,
  Atom,
  Prob
)
```

**Arguments**

realMZ	mz vector in MS2 scan
realIntensity	intensity vector in MS2 scan
realCharge	charge vector in MS2 scan
parentCharge	precursor charge, 2 for example
pepSeq	a string of peptide
Atom	"C13" or "N15"
Prob	its SIP abundance (0.0~1.0)

**Value**

a score of this PSM

**Examples**

```
demo_file <- system.file("extdata", "107728.FT2", package = "Aerith")
scan1 <- readOneScanMS2(ftFile = demo_file, scanNumber = 107728)
score <- scorePSMsimple(
  scan1$peaks$mz,
  scan1$peaks$intensity,
  scan1$peaks$charge,
  2,
  "[HSQVFSTAEDNQSAVTIHVLQGER]",
  "C13",
  0.0107
)
```

---

summaryPSMsipPCT

*Summarize SIP percent for PSMs*


---

**Description**

This function reads a PSM file and computes summary statistics for the SIP percent values, including the total count, average, median, Median Absolute Deviation (MAD), standard deviation, estimated FDR, the number of labeled PSMs, and the median SIP percent for the labeled PSMs.

**Usage**

```
summaryPSMsipPCT(psmPath, SIPthreshold = 5, chargeThreshold = 3)
```

**Arguments**

psmPath            A character string specifying the path to the PSM file.  
 SIPthreshold     Numeric value representing the SIP threshold (default is 5).  
 chargeThreshold   Numeric value representing the parent charge threshold (default is 3).

**Value**

A data frame containing summary statistics of the SIP percent values.

**Examples**

```
demo_file <- system.file("extdata", "demo.psm.txt", package = "Aerith")
summaryStats <- summaryPSMsipPCT(demo_file)
print(summaryStats)
```

---

writeAllScanMS1            *write all MS1 scans has charge*

---

**Description**

write all MS1 scans has charge

**Usage**

```
writeAllScanMS1(header, scansList, ftFile)
```

**Arguments**

header            a list of FT file header  
 scansList        a list of scans for output  
 ftFile            a ft1 file's output path

**Value**

TRUE if the file was written successfully, FALSE otherwise

## Examples

```
rds <- system.file("extdata", "demo.FT1.rds", package = "Aerith")
demo_file <- tempfile(fileext = ".FT1")
writeLines(readRDS(rds), demo_file)
header <- readFTheader(demo_file)
ft1 <- readAllScanMS1(demo_file)
tmp <- tempdir()
writeAllScanMS1(header, ft1[1:10], file.path(tmp, "demo10.FT1"))
list.files(tmp, pattern = "demo10.FT1", full.names = TRUE)
```

---

writeAllScanMS2	<i>write all MS2 scans has charge</i>
-----------------	---------------------------------------

---

## Description

write all MS2 scans has charge

## Usage

```
writeAllScanMS2(header, scansList, ftFile)
```

## Arguments

header	a list of FT file header
scansList	a list of scans for output
ftFile	a ft2 file's output path

## Value

TRUE if the file was written successfully, FALSE otherwise

## Examples

```
demo_file <- system.file("extdata", "demo.FT2", package = "Aerith")
header <- readFTheader(demo_file)
ft2 <- readAllScanMS2(demo_file)
tmp <- tempdir()
writeAllScanMS2(header, ft2[1:10], file.path(tmp, "demo10.FT2"))
list.files(tmp, pattern = "demo10.FT2", full.names = TRUE)
```

---

```
writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel
    writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel read each
    scan's top PSMs from multiple .Spe2PepFile.txt files of each .FT2 file
    and write them to one tsv file
```

---

## Description

writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel read each scan's top PSMs from multiple .Spe2PepFile.txt files of each .FT2 file and write them to one tsv file

## Usage

```
writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel(
  workingPath,
  topN = 5L,
  fileName = "a.tsv"
)
```

## Arguments

workingPath	a full path with .Spe2PepFile.txt files in it
topN	store top N PSMs of each scan of one .FT2 file
fileName	the output path

## Value

nothing but write a tsv of top N PSMs

## Examples

```
tmp <- tempdir()
sip_dir <- file.path(tmp, "sip")
dir.create(sip_dir)
demo_file <- system.file("extdata", "demo_target.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000target.Spe2Pep.txt"))
demo_file <- system.file("extdata", "demo_decoy.Spe2Pep.txt", package = "Aerith")
file.copy(demo_file, file.path(sip_dir, "Pan_052322_X13.SIP_C13_050_000decoy.Spe2Pep.txt"))
writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel(sip_dir, 3, file.path(sip_dir, "top3.tsv"))
list.files(sip_dir, full.names = TRUE)
print(file.info(file.path(sip_dir, "top3.tsv")))
```

# Index

- \* **internal**
  - .onLoad, 4
  - .onUnload, 5
  - Aerith-package, 4
- .onLoad, 4
- .onUnload, 5
- AAspectra-class, 5
- Aerith (Aerith-package), 4
- aerith (Aerith-package), 4
- Aerith-package, 4
- annotatePrecursor, 6
- annotatePSM, 7
  
- BYion\_peak\_calculator\_DIY, 8
  
- cal\_isotope\_numbers, 11
- cal\_isotope\_numbers\_SIP, 12
- cal\_isotope\_peaks\_fft, 12
- calBYAtomCountAndBaseMass, 9
- calPepAtomCount, 10
- calPepNeutronMass, 10
- calPepPrecursorMass, 11
  
- data.frame, 5
- denoiseOneMS2ScanHasCharge, 13
  
- extractPSMfeatures, 14
- extractPSMfeaturesTargetAndDecoy, 15
- extractPSMfeaturesTargetAndDecoytoPercolatorP1, 16
  
- generateCFGs, 17
- generateOneCFG, 18
- getFilterThreshold, 19
- getFilterThresholdTopPSMs, 19
- getFilterThresholdTopPSMsSpe2Pep, 20
- getMZ, 21
- getPrecursorSpectra, 22
- getPrecursorSpectra(), 6
- getRealScan, 22
- getRealScanFromList, 23
- getRealScans, 23
- getRealScansWithCharges, 24
- getRealScanWithCharge, 25
- getRetentionTimeAndPrecursorInfo, 25
- getSipBYionSpectra, 26
- getSipBYionSpectra(), 6
- getSipPrecursorSpectra, 27
- getSipPrecursorSpectra(), 6
- getTIC, 28
- getUnfilteredPeptides, 28
- getUnfilteredPSMs, 29
  
- plot, AApectra, missing-method, 6, 29
- plot, AApectra-method
  - (plot, AApectra, missing-method), 29
- plotFilteredPCTIntensitySummary, 30
- plotMolecularFFTisotopes, 31
- plotMolecularIsotopes, 32
- plotPrecursorAnnotation, 33
- plotPrecursorMzFrequency, 35
- plotProSipPct, 35
- plotPSMannotation, 36
- plotPSMs, 37
- plotPSMsipPCT, 38
- plotRealScan, 39
- plotScanFrequency, 40
- plotScanFrequencyMS2, 41
- plotScoreDistribution, 41
- plotSipBYionLabel, 42
- plotSIPfilteredPCTIntensityBySample, 42
- plotTIC, 43
- precursor\_peak\_calculator, 44
- precursor\_peak\_calculator\_DIY, 44
- precursor\_peak\_calculator\_DIY\_averagine, 45
  
- rankyfyfy, 46

readAllScanMS1, [46](#)  
readAllScanMS2, [47](#)  
readFilesScansTopPSMs, [47](#)  
readFilesScansTopPSMsFromOneFT2, [48](#)  
readFTheader, [49](#)  
readMgf, [49](#)  
readMzm1MS1, [50](#)  
readMzm1MS2, [50](#)  
readOneScanMS1, [51](#)  
readOneScanMS2, [52](#)  
readPepXMLtable, [52](#)  
readPSMtsv, [53](#)  
readScansMS1, [53](#)  
readScansMS1Vector, [54](#)  
readScansMS2, [55](#)  
readScansMS2Vector, [55](#)  
readSip, [56](#)  
readSips, [57](#)  
readSpe2Pep, [57](#)  
readSpe2PepFilesScansTopPSMs, [58](#)  
readSpe2PepFilesScansTopPSMsFromEachFT2Parallel,  
[59](#)  
readSpe2PepFilesScansTopPSMsFromEachFT2TargetAndDecoyParallel,  
[59](#)  
readSpe2PepFilesScansTopPSMsFromOneFT2,  
[60](#)  
readSpe2Peps, [61](#)  
residue\_peak\_calculator\_DIY, [62](#)  
  
scoreIntensity, [62](#)  
scoreIntensityByCE, [63](#)  
scorePSM, [64](#)  
scorePSMsimple, [64](#)  
summaryPSMsipPCT, [65](#)  
  
writeAllScanMS1, [66](#)  
writeAllScanMS2, [67](#)  
writeSpe2PepFilesScansTopPSMsFromEachFT2Parallel,  
[68](#)