Package 'DIAlignR'

April 28, 2024

Type Package

Title Dynamic Programming Based Alignment of MS2 Chromatograms

Version 2.11.0

Description
To obtain unbiased proteome coverage from a biological sample, mass-spectrometer is operated in Data Independent Acquisition (DIA) mode. Alignment of these DIA runs establishes consistency and less missing values in complete data-matrix. This package implements dynamic programming with affine gap penalty based approach for pair-wise alignment of analytes. A hybrid approach of global alignment (through MS2 features) and local alignment (with MS2 chromatograms) is implemented in this tool.

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.2.1

biocViews MassSpectrometry, Metabolomics, Proteomics, Alignment, Software

Depends methods, stats, R (>= 4.0)

Imports zoo (>= 1.8-3), data.table, magrittr, dplyr, tidyr, rlang, mzR (>= 2.18), signal, bit64, reticulate, ggplot2, RSQLite, DBI, ape, phangorn, pracma, RMSNumpress, Rcpp

Suggests knitr, akima, lattice, scales, gridExtra, latticeExtra, rmarkdown, BiocStyle, BiocParallel, testthat (>= 2.1.0)

VignetteBuilder knitr

BugReports https://github.com/shubham1637/DIAlignR/issues

LinkingTo Rcpp, RcppEigen

SystemRequirements C++14

git_url https://git.bioconductor.org/packages/DIAlignR

git_branch devel

git_last_commit 61c355a

git_last_commit_date 2023-10-24

Repository Bioconductor 3.19

Date/Publication 2024-04-28

Author Shubham Gupta [aut, cre] (<https://orcid.org/0000-0003-3500-8152>),
Hannes Rost [aut] (<https://orcid.org/0000-0003-0990-7488>),
Justin Sing [aut]

Maintainer Shubham Gupta <shubham.1637@gmail.com>

Contents

`addFlankToLeft` *Add signal to the left of XIC*

Description

This function copies flanking chromatogram from XIC and paste it to the left of newXIC.

Usage

```
addFlankToLeft(flankSeq, XIC, newXIC)
```

Arguments

`flankSeq`	(logical) must be TRUE at the front of the vector.
`XIC`	(data-frame) first column is time, second column is intensity.
`newXIC`	(data-frame) first column is time, second column is intensity.

Value

(dataframe) has two columns:

`time`	(numeric)
`intensity`	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXIC, addFlankToRight

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
       4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom2 <- data.frame(time = c(3013.4, 3016, 3020), intensity = c(1.2, 3.4, 5.6))
flankSeq <- as.logical(c(1,1,0,0,0,0,0,0,0,0,0,0,1,1))
## Not run:
addFlankToLeft(flankSeq, chrom, chrom2)

## End(Not run)
```

---

addFlankToRight *Add signal to the right of XIC*

Description

This function copies flanking chromatogram from XIC and paste it to the right of newXIC.

Usage

```
addFlankToRight(flankSeq, XIC, newXIC)
```

Arguments

flankSeq	(logical) must be TRUE at the tail of the vector.
XIC	(data-frame) first column is time, second column is intensity.
newXIC	(data-frame) first column is time, second column is intensity.

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-07-16

See Also

childXIC, addFlankToLeft

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
       4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom2 <- data.frame(time = c(3013.4, 3016, 3020), intensity = c(1.2, 3.4, 5.6))
flankSeq <- as.logical(c(1,1,0,0,0,0,0,0,0,0,0,0,1,1))
## Not run:
addFlankToRight(flankSeq, chrom, chrom2)

## End(Not run)
```

---

addXIC *Add XIC to pyopenms experiment*

Description

A chromatogram and its repective native ID (transition ID) is added to MSExperiment object.

Usage

```
addXIC(ropenms, expriment, xic, nativeId)
```

Arguments

ropenms	(pyopenms module) get this python module through get_ropenms().
expriment	(python object) an MSExperiment() created using ropenms.
xic	(data-frame) must have two numeric columns.
nativeId	(integer) transition ID of the xic.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-06-06

Examples

```
## Not run:
ropenms <- get_ropenms(condaEnv = "envName", useConda=TRUE)
expriment <- ropenms$MSExperiment()
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR)
xic <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]][["4618"]][[1]]
addXIC(ropenms, expriment, xic, 34L)
chroms <- expriment$getChromatograms()
reticulate::py_to_r(chroms[[0]]$getNativeID())
reticulate::py_to_r(chroms[[0]]$get_peaks())

## End(Not run)
```

AffineAlignObj-class *An S4 object for class AffineAlignObj*

Description

s is a point-wise similarity matrix between signalA and signalB. Intermediate matrices M,A,B are calculated from s for affine-alignment. Each cell of the Traceback matrix has coordinate of its parent cell. path matrix is a binary matrix with ones indicating path of maximum cumulative score. GapOpen and GapExten are gap-opening and gap-extension penalties used by affine alignment algorithm. indexA_aligned and indexB_aligned are aligned indices of signalA and SignalB. The cumulative alignment score is in score vector.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

doAffineAlignmentCpp

`AffineAlignObjLight-class` *An S4 object for class AffineAlignObjLight It only contains aligned indices.*

Description

indexA_aligned and indexB_aligned are aligned indices of signalA and SignalB. The cumulative alignment score is in score vector.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

`doAffineAlignmentCpp`

`AffineAlignObjMedium-class` *An S4 object for class AffineAlignObjMedium. It only contains similarity matrix and aligned indices.*

Description

s is a point-wise similarity matrix between signalA and signalB. path matrix is a binary matrix with ones indicating path of maximum cumulative score. GapOpen and GapExten are gap-opening and gap-extension penalties used by affine alignment algorithm. indexA_aligned and indexB_aligned are aligned indices of signalA and SignalB. The cumulative alignment score is in score vector.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

`doAffineAlignmentCpp`

`alignChromatogramsCpp` *Aligns MS2 extracted-ion chromatograms(XICs) pair.*

Description

Aligns MS2 extracted-ion chromatograms(XICs) pair.

Usage

```
alignChromatogramsCpp(
  l1,
  l2,
  alignType,
  tA,
  tB,
  normalization,
  simType,
  B1p = 0,
  B2p = 0,
  noBeef = 0L,
  goFactor = 0.125,
  geFactor = 40,
  cosAngleThresh = 0.3,
  OverlapAlignment = TRUE,
  dotProdThresh = 0.96,
  gapQuantile = 0.5,
  kerLen = 9L,
  hardConstrain = FALSE,
  samples4gradient = 100,
  objType = "heavy"
)
```

Arguments

`l1`	(list) A list of numeric vectors. l1 and l2 should have same length.
`l2`	(list) A list of numeric vectors. l1 and l2 should have same length.
`alignType`	(char) A character string. Available alignment methods are "global", "local" and "hybrid".
`tA`	(numeric) A numeric vector. This vector has equally spaced timepoints of XIC A.
`tB`	(numeric) A numeric vector. This vector has equally spaced timepoints of XIC B.
`normalization`	(char) A character string. Normalization must be selected from (L2, mean or none).

simType	(char) A character string. Similarity type must be selected from (dotProductMasked, dotProduct, cosineAngle, cosine2Angle, euclideanDist, covariance, correlation, crossCorrelation). Mask = s > quantile(s, dotProdThresh) AllowDotProd= [Mask × cosine2Angle + (1 - Mask)] > cosAngleThresh s_new= s × AllowDotProd
B1p	(numeric) Timepoint mapped by global fit for tA[1].
B2p	(numeric) Timepoint mapped by global fit for tA[length(tA)].
noBeef	(integer) It defines the distance from the global fit, upto which no penalization is performed. The window length = 2*noBeef.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) if false; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
objType	(char) A character string. Must be either light, medium or heavy.

Value

affineAlignObj (S4class) A S4class object from C++ AffineAlignObj struct.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 Date: 2019-03-08

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR
data(oswFiles_DIAlignR, package="DIAlignR")
```

```
oswFiles <- oswFiles_DIAlignR
XICs.ref <- XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XICs[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
tVec.ref <- XICs.ref[[1]][["time"]] # Extracting time component
tVec.eXp <- XICs.eXp[[1]][["time"]] # Extracting time component
B1p <- 4964.752
B2p <- 5565.462
noBeef <- 77.82315/3.414
l1 <- lapply(XICs.ref, `[[`, 2)
l2 <- lapply(XICs.eXp, `[[`, 2)
AlignObj <- alignChromatogramsCpp(l1, l2, alignType = "hybrid", tA = tVec.ref, tB = tVec.eXp,
 normalization = "mean", simType = "dotProductMasked", B1p = B1p, B2p = B2p, noBeef = noBeef,
 goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3, OverlapAlignment = TRUE,
 dotProdThresh = 0.96, gapQuantile = 0.5, hardConstrain = FALSE, samples4gradient = 100,
 objType = "light")
```

`alignedXIC` *Create an aligned chromatogram*

Description

Modifies chromatogram to have the same length as indices. Imputes missing values with appropriate method. Time and intensity for the flanking missing indices are set as NA.

Usage

```
alignedXIC(
  XIC,
  indices,
  method = "spline",
  polyOrd = 4,
  kernelLen = 9,
  splineMethod = "fmm"
)
```

Arguments

`XIC`	(data-frame) first column is time, second column is intensity.
`indices`	(integer) vector of monotonically increasing integers.
`method`	(string) must be either "spline", "sgolay" or "linear".
`polyOrd`	(integer) must be less than kernelLen.
`kernelLen`	(integer) must be an odd integer.
`splineMethod`	(string) must be either "fmm" or "natural".

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXIC, imputeChromatogram

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(alignObj_DIAlignR, package="DIAlignR")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
alignedIndices <- cbind(alignObj_DIAlignR@indexA_aligned, alignObj_DIAlignR@indexB_aligned)
colnames(alignedIndices) <- c("indexAligned.ref", "indexAligned.eXp")
alignedIndices[, 1:2][alignedIndices[, 1:2] == 0] <- NA_integer_
## Not run:
plot(alignedXIC(XICs.ref[[1]], alignedIndices[,"indexAligned.ref"]), type = "l")

## End(Not run)
```

---

alignmentStats *Prints alignment summary*

Description

Prints alignment summary

Usage

```
alignmentStats(finalTbl, params)
```

Arguments

finalTbl	(dataframe) an output of writeTables function.
params	(list) must have following elements: alignedFDR2 and unalignedFDR.

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-07-15

See Also

paramsDIAlignR, writeTables

AlignObj-class *An S4 object for class AlignObj*

Description

s is a point-wise similarity matrix between signalA and signalB. Intermediate matrices M is calculated from s for alignment. Each cell of the Traceback matrix has coordinate of its parent cell. path matrix is a binary matrix with ones indicating path of maximum cumulative score. GapOpen and GapExten are gap-opening and gap-extension penalties used by alignment algorithm. They must be the same. indexA_aligned and indexB_aligned are aligned indices of signalA and SignalB. The cumulative alignment score is in score vector.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

doAlignmentCpp

---

alignObj_DIAlignR *Alignment object of a peptide.*

Description

Aligned XICs of peptide (ID = 4618) 14299_QFNNTDIVLLEDFQK/3 across two SWATH runs:

run1 : hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML
run2 : hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML

Usage

```
alignObj_DIAlignR
```

Format

A S4 object of 16 slots:

s similarity score matrix.

M Match or Mismatch matrix, residues of A and B are aligned without a gap. M(i,j) = Best score upto (i,j) given Ai is aligned to Bj.

A Insert in sequence A, residue in A is aligned to gap in B. A(i,j) is the best score given that Ai is aligned to a gap in B.

B Insert in sequence B, residue in B is aligned to gap in A. B(i,j) is the best score given that Bj is aligned to a gap in A.

Traceback Traceback matrices store source matrix name and direction as matrices are filled with dynamic programming.

path Path matrix would represent alignment path through similarity matrix as binary-hot encoding.

signalA_len Number of data-points in signal A.

signalB_len Number of data-points in signal B.

GapOpen Penalty for Gap opening. For n consecutive gaps: Penalty = GapOpen + (n-1)*GapExten.

GapExten Penalty for Gap extension. For n consecutive gaps: Penalty = GapOpen + (n-1)*GapExten.

FreeEndGaps True for Overlap alignment.

indexA_aligned Aligned signalA indices after affine alignment.

indexB_aligned Aligned signalB indices after affine alignment.

score Cumulative score along the aligned path.

simScore_forw Not needed, will be removed.

nGaps Total number of gaps in the alignment path.

Source

C++ code is exaplained at DIAlign namespace. File test_GenerateData.R has source code to generate the example data.

---

alignTargetedRuns *Outputs intensities for each analyte from aligned Targeted-MS runs*

Description

This function expects osw and xics directories at dataPath. It first reads osw files and fetches chromatogram indices for each analyte. It then align XICs of its reference XICs. Best peak, which has lowest m-score, about the aligned retention time is picked for quantification.

Usage

```
alignTargetedRuns(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  scoreFile = NULL,
  runs = NULL,
  peps = NULL,
  refRun = NULL,
  applyFun = lapply,
  saveAlignedPeaks = FALSE
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
scoreFile	(string) path to the peptide score file, needed when oswMerged is FALSE.
runs	(string) names of xics file without extension.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
refRun	(string) reference for alignment. If no run is provided, m-score is used to select reference run.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.
saveAlignedPeaks	(logical) Save a mapping table to track aligned feature ids against reference feature id

Value

An output table with following columns: precursor, run, intensity, RT, leftWidth, rightWidth, peak_group_rank, m_score, alignment_rank, peptide_id, sequence, charge, group_label.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

References

Gupta S, Ahadi S, Zhou W, Röst H. "DIAlignR Provides Precise Retention Time Alignment Across Distant Runs in DIA and Targeted Proteomics." Mol Cell Proteomics. 2019 Apr;18(4):806-817. doi: https://doi.org/10.1074/mcp.TIR118.001132 Epub 2019 Jan 31.

See Also

getRunNames, getFeatures, setAlignmentRank, getMultipeptide

Examples

```
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
dataPath <- system.file("extdata", package = "DIAlignR")
BiocParallel::register(BiocParallel::MulticoreParam(workers = 4, progressbar = TRUE))
alignTargetedRuns(dataPath, outFile = "testDIAlignR", params = params, applyFun = BiocParallel::bplapply)
```

---

alignToMaster *Align descendants to master*

Description

During traverse-down, parent runs are aligned to the master/child run. This function performs the alignment by already saved aligned parent-child time vectors. For the aligned peaks, alignment_rank is set to 1 in multipeptide environment.

Usage

```
alignToMaster(
  ref,
  eXp,
  alignedVecs,
  refRun,
  adaptiveRT,
  multipeptide,
  prec2chromIndex,
  mzPntrs,
  fileInfo,
  precursors,
  params,
  applyFun = lapply
)
```

Arguments

ref	(string) name of the descendant run. Must be in the rownames of fileInfo.
eXp	(string) name of one of the parent run. Must be in the rownames of fileInfo.
alignedVecs	(list of dataframes) Each dataframe contains aligned parents time-vectors and resulting child/master time vector for a precursor. This is the second element of getChildXICs output.
refRun	(integer) must be of the same length as of precursors. 1: reference is runA, 2: reference is runB.

multipeptide	(environment) contains multiple data-frames that are collection of features associated with analytes. This is an output of getMultipeptide.
prec2chromIndex	(list) a list of dataframes having following columns: transition_group_id: it is PRECURSOR.ID from osw file. chromatogramIndex: index of chromatogram in mzML file.
mzPntrs	(list) a list of mzRpwiz.
fileInfo	(data-frame) output of getRunNames.
precursors	(data-frame) atleast two columns transition_group_id and transition_ids are required.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Details

refRun is flipped if eXp is runB instead of runA.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

traverseUp, traverseDown, setAlignmentRank

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
mzPntrs <- list2env(getMZMLpointers(fileInfo))
features <- list2env(getFeatures(fileInfo, maxFdrQuery = 0.05, runType = "DIA_Proteomics"))
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, runType = params[["runType"]],
 context = "experiment-wide", maxPeptideFdr = params[["maxPeptideFdr"]])
precursors <- dplyr::arrange(precursors, .data$peptide_id, .data$transition_group_id)
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, oswMerged = TRUE, params[["runType"]], params[["context"]
peptideScores <- lapply(peptideIDs, function(pep) dplyr::filter(peptideScores, .data$peptide_id == pep))
names(peptideScores) <- as.character(peptideIDs)
prec2chromIndex <- list2env(getChromatogramIndices(fileInfo, precursors, mzPntrs))
multipeptide <- getMultipeptide(precursors, features)
prec2chromIndex <- list2env(getChromatogramIndices(fileInfo, precursors, mzPntrs))
```

```
adaptiveRTs <- new.env()
refRuns <- new.env()
tree <- ape::reorder.phylo(ape::read.tree(text = "(run1:7,run2:2)master1;"), "postorder")
## Not run:
ropenms <- get_ropenms(condaEnv = "envName", useConda=TRUE)
multipeptide <- traverseUp(tree, dataPath, fileInfo, features, mzPntrs, prec2chromIndex, precursors, params,
 adaptiveRTs, refRuns, multipeptide, peptideScores, ropenms)
multipeptide <- getMultipeptide(precursors, features)
alignedVecs <- readRDS(file = file.path(dataPath, "master1_av.rds"))
adaptiveRT <- (adaptiveRTs[["run1_run2"]] + adaptiveRTs[["run2_run1"]])/2
multipeptide[["14383"]]$alignment_rank[multipeptide[["14383"]]$run == "master1"] <- 1L
multipeptide <- alignToMaster(ref = "master1", eXp = "run1", alignedVecs, refRuns[["master1"]][,1], adaptiveRT,
 multipeptide, prec2chromIndex, mzPntrs, fileInfo, precursors, params)
# Cleanup
for(run in names(mzPntrs)) DBI::dbDisconnect(mzPntrs[[run]])
file.remove(file.path(dataPath, "master1_av.rds"))
file.remove(file.path(dataPath, "xics", "master1.chrom.sqMass"))

## End(Not run)
```

`alignToRef` *Aligns an analyte from an experiment to the reference run*

Description

df contains unaligned features for an analyte across multiple runs. This function aligns eXp run to ref run and updates corresponding features.

Usage

```
alignToRef(
  eXp,
  ref,
  refIdx,
  fileInfo,
  XICs,
  XICs.ref,
  params,
  df,
  globalFits,
  RSE,
  feature_alignment_map = NULL
)
```

Arguments

`eXp`	(string) name of the run to be aligned to reference run. Must be in the rownames of fileInfo.

ref	(string) name of the reference run. Must be in the rownames of fileInfo.
refIdx	(integer) index of the reference feature in df.
fileInfo	(data-frame) output of getRunNames.
XICs	(list of dataframes) fragment-ion chromatograms of the analytes for all runs.
XICs.ref	(list of dataframes) fragment-ion chromatograms of the analyte_chr from the reference run.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
df	(dataframe) a collection of features related to the peptide
globalFits	(list) each element is either of class lm or loess. This is an output of getGlobalFits.
RSE	(list) Each element represents Residual Standard Error of corresponding fit in globalFits.
feature_alignment_mapping	(data.table) contains experiment feature ids mapped to corresponding reference feature id per analyte. This is an output of getRefExpFeatureMap.

Value

invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignTargetedRuns, perBatch, setAlignmentRank, getMultipeptide, getRefExpFeatureMap

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
```

---

alignToRefMST *Aligns an analyte for an edge of MST*

Description

df contains unaligned features for an analyte across multiple runs. This function aligns eXp run to ref run and updates corresponding features.

Usage

```
alignToRefMST(iNet, net, fileInfo, XICs, params, analytes, df, globalFits, RSE)
```

Arguments

iNet	(integer) the index of edge to be aligned in the net.
net	(matrix) each row represents an edge of MST.
fileInfo	(data-frame) output of getRunNames.
XICs	(list of dataframes) fragment-ion chromatograms of the analytes for all runs.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
analytes	(string) precursor IDs of the requested peptide.
df	(dataframe) a collection of features related to analytes.
globalFits	(list) each element is either of class lm or loess. This is an output of getGlobalFits.
RSE	(list) Each element represents Residual Standard Error of corresponding fit in globalFits.

Value

invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

mstAlignRuns, MSTperBatch, setAlignmentRank, getMultipeptide

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
```

---

alignToRoot4 *Step 4 for progressive alignment*

Description

This is needed when leaves of the tree are directly aligned to the root

Usage

```
alignToRoot4(
  dataPath,
  params,
  outFile = "DIAlignR",
  oswMerged = TRUE,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
outFile	(string) name of the output file.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

progAlignRuns

---

analytesFromFeatures *Outputs analytes below FDR*

Description

Provides all found analytes or only common analytes that have m-score less than analyteFDR.

Usage

```
analytesFromFeatures(features, analyteFDR = 1, commonAnalytes = TRUE)
```

Arguments

features	(list of data-frames) contains features and their properties identified in each run.
analyteFDR	(numeric) only analytes that have m-score less than this, will be included in the output.
commonAnalytes	(logical) TRUE: intersect across all runs, FASLE: union across all runs.

Value

a vector of integer.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getFeatures

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
## Not run:
commonAnalytes <- analytesFromFeatures(features, analyteFDR = 0.01, commonAnalytes = TRUE)

## End(Not run)
```

---

approxFill *Fill missing values using linear interpolation*

Description

Fill missing values using linear interpolation

Usage

```
approxFill(chrom)
```

Arguments

chrom	(data-frames) first column is time, second column is intensity.

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-05-21

See Also

na.approx, approx

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
 4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom$y[c(1,8, 14)] <- NA
## Not run:
approxFill(chrom)

## End(Not run)
```

`areaIntegrator` *Calculates area between signal-boundaries.*

Description

This function sums all the intensities between left-index and right-index.

Usage

```
areaIntegrator(
  l1,
  l2,
  left,
  right,
  integrationType,
  baselineType,
  fitEMG,
  baseSubtraction,
  kernelLen = 0L,
  polyOrd = 3L
)
```

Arguments

`l1`	(list) A list of time vectors.
`l2`	(list) A list of intensity vectors.
`left`	(numeric) left boundary of the peak.
`right`	(numeric) right boundary of the peak.
`integrationType`	(string) method to ompute the area of a peak contained in XICs. Must be from "intensity_sum", "trapezoid", "simpson".
`baselineType`	(string) method to estimate the background of a peak contained in XICs. Must be from "base_to_base", "vertical_division_min", "vertical_division_max".
`fitEMG`	(logical) enable/disable exponentially modified gaussian peak model fitting.

baseSubtraction
: (logical) TRUE: remove background from peak signal using estimated noise levels.

kernelLen
: (integer) length of filter. Must be an odd number.

polyOrd
: (integer) TRUE: remove background from peak signal using estimated noise levels.

Value

area (numeric).

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR", package = "DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
l1 <- lapply(XICs, `[[`, 1) # time
l2 <- lapply(XICs, `[[`, 2) # intensity
areaIntegrator(l1, l2, left = 5203.7, right = 5268.5, "intensity_sum", "base_to_base", FALSE, TRUE)
# 66.10481 69.39996 46.53095 16.34266 13.13564 13.42331
areaIntegrator(l1, l2, left = 5203.7, right = 5268.5, kernelLen = 9L, "intensity_sum", "base_to_base", FALSE, TRUE)
# 65.01449 71.74432 52.73518 23.84420 17.61869 16.48190
```

---

as.list,AffineAlignObj-method

Converts instances of class AffineAlignObj into list

Description

Converts instances of class AffineAlignObj into list

Usage

```
## S4 method for signature 'AffineAlignObj'
as.list(x)
```

Arguments

x
: An object of class AffineAlignObj.

Value

list

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2020-03-31

---

as.list,AffineAlignObjLight-method

Converts instances of class AffineAlignObjLight into list

Description

Converts instances of class AffineAlignObjLight into list

Usage

```
## S4 method for signature 'AffineAlignObjLight'
as.list(x)
```

Arguments

x	An object of class AffineAlignObjLight

Value

list

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2020-03-31

---

as.list,AffineAlignObjMedium-method

Converts instances of class AffineAlignObjMedium into list

Description

Converts instances of class AffineAlignObjMedium into list

Usage

```
## S4 method for signature 'AffineAlignObjMedium'
as.list(x)
```

Arguments

x	An object of class AffineAlignObjMedium

Value

list

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

---

as.list,AlignObj-method

Converts instances of class AlignObj into list

Description

Converts instances of class AlignObj into list

Usage

```
## S4 method for signature 'AlignObj'
as.list(x)
```

Arguments

x	An object of class AlignObj

Value

list

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

`blobXICs` *Format XICs to blob*

Description

Format XICs to blob

Usage

```
blobXICs(XICs, nativeIds, lossy = TRUE)
```

Arguments

`XICs`	(list) a list of data-frames. Each data frame has elution time and intensity of fragment-ion XIC.
`lossy`	(logical) if TRUE, time and intensity are lossy-compressed.
`nativeId`	(integer) transition ID of the xic.

Details

DATA_TYPE is one of 0 = mz, 1 = intensity, 2 = rt
COMPRESSION is one of 0 = no, 1 = zlib, 2 = np-linear, 3 = np-slof, 4 = np-pic, 5 = np-linear + zlib, 6 = np-slof + zlib, 7 = np-pic + zlib

Value

(data.frame)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR)
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]][["4618"]]
nativeIds <- 27706:27711
## Not run:
blobXICs(XICs, nativeIds)

## End(Not run)
```

`calculateIntensity` *Calculates area of a peak in XIC group*

Description

Retention time from reference run is mapped to experiment run using AlignObj.

Usage

```
calculateIntensity(XICs, left, right, params)
```

Arguments

`XICs`	(list) list of extracted ion chromatograms of a precursor.
`left`	(numeric) left boundary of the peak.
`right`	(numeric) right boundary of the peak.
`params`	(list) parameters are entered as list. Output of the `paramsDIAlignR` function.

Value

area (numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`areaIntegrator`, `setAlignmentRank`

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
## Not run:
calculateIntensity(XICs, 5220, 5261, integrationType = "intensity_sum",
 baselineType = "base_to_base", fitEMG = FALSE)

## End(Not run)
```

checkOverlap *Overlap of two time ranges*

Description

Overlap of two time ranges

Usage

```
checkOverlap(x, y)
```

Arguments

x	(numeric) must have only two values and x[2] > x[1].
y	(numeric) must have only two values and y[2] > y[1].

Value

(logical) TRUE: both time ranges overlap. FALSE: both time ranges do not overlap.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getChildFeature

Examples

```
## Not run:
checkOverlap(c(9.1, 13.1), c(2.1, 3.1))
checkOverlap(c(1.1, 3.1), c(3.2, 7.1))

## End(Not run)
```

checkParams *Checks all the alignment parameters*

Description

Parameters are defined in the paramsDIAlignR function. If parameters are found incongruous, the program will terminate.

Usage

```
checkParams(params)
```

Arguments

params	(list) parameters are entered as list. Output of the paramsDIAlignR function.

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

paramsDIAlignR

Examples

```
params <- paramsDIAlignR()
## Not run:
checkParams(params)

## End(Not run)
```

childXIC *Get a child chromatogram from parents*

Description

Internal gaps in reference chromatogram and corresponding indices in experiment chromatogram are discarded while merging both chromatogram to obtain a child chromatogram.

Usage

```
childXIC(
  XIC.ref,
  XIC.eXp,
  alignedIndices,
  method = "spline",
  polyOrd = 4,
  kernelLen = 9,
  splineMethod = "fmm",
  wRef = 0.5,
  mergeStrategy = "avg",
  keepFlanks = TRUE
)
```

Arguments

XIC.ref	(data-frame) extracted ion chromatogram from reference run. Must not contain missing values.
XIC.eXp	(data-frame) extracted ion chromatogram from experiment run. Must not contain missing values.
alignedIndices	(data-frame) must have two columns "indexAligned.ref" and "indexAligned.eXp".
method	(string) must be either "spline", "sgolay" or "linear".
polyOrd	(integer) must be less than kernelLen.
kernelLen	(integer) must be an odd integer.
splineMethod	(string) must be either "fmm" or "natural".
wRef	(numeric) Weight of the reference XIC. Must be between 0 and 1.
mergeStrategy	(string) must be either ref, avg, refStart or refEnd.
keepFlanks	(logical) TRUE: Flanking chromatogram is not removed.

Value

(list) the first element is chromatogram. The second element is aligned parent time-vectors.

time-merging

There are three startegies for calculating time to keep sampling rate equal to parents:
- Use reference time.
- Average time (Non-gap region will have parent's sampling rate).
- Start/end with the average then fixed sampling rate.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

mergeXIC, alignedXIC, childXICs

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(alignObj_DIAlignR, package="DIAlignR")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
alignedIndices <- cbind(alignObj_DIAlignR@indexA_aligned, alignObj_DIAlignR@indexB_aligned)
colnames(alignedIndices) <- c("indexAligned.ref", "indexAligned.eXp")
alignedIndices[, 1:2][alignedIndices[, 1:2] == 0] <- NA_integer_
## Not run:
plot(childXIC(XICs.ref[[1]], XICs.eXp[[1]], alignedIndices)[[1]], type = 'l')

## End(Not run)
```

---

childXICs *Get child chromatograms from parents*

Description

Get child chromatograms from parents

Usage

```
childXICs(
  XICs.ref,
  XICs.eXp,
  alignedIndices,
  method = "spline",
  polyOrd = 4,
  kernelLen = 9,
  splineMethod = "fmm",
```

```
  wRef = 0.5,
  mergeStrategy = "avg",
  keepFlanks = TRUE
)
```

Arguments

XICs.ref	(list of data-frames) extracted ion chromatograms from reference run.
XICs.eXp	(list of data-frames) extracted ion chromatograms from experiment run.
alignedIndices	(data-frame) must have two columns "indexAligned.ref" and "indexAligned.eXp".
method	(string) must be either "spline", "sgolay" or "linear".
polyOrd	(integer) must be less than kernelLen.
kernelLen	(integer) must be an odd integer.
splineMethod	(string) must be either "fmm" or "natural".
wRef	(numeric) Weight of the reference XIC. Must be between 0 and 1.
mergeStrategy	(string) must be either ref, avg, refStart or refEnd.
keepFlanks	(logical) TRUE: Flanking chromatogram is not removed.

Value

(list) the first element is a list of chromatograms. The second element is aligned parent time-vectors.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXIC, mergeXIC

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(alignObj_DIAlignR, package="DIAlignR")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
alignedIndices <- cbind(alignObj_DIAlignR@indexA_aligned, alignObj_DIAlignR@indexB_aligned)
colnames(alignedIndices) <- c("indexAligned.ref", "indexAligned.eXp")
alignedIndices[, 1:2][alignedIndices[, 1:2] == 0] <- NA_integer_
newXICs <- childXICs(XICs.ref, XICs.eXp, alignedIndices)[[1]]
plotXICgroup(XICs.ref)
plotXICgroup(newXICs)
```

chromatogramIdAsInteger *Coerce chromatogram ids as integer*

Description

chromatogramHeader has 10 columns. The two important columns are: "chromatogramId" which has fragment-ion ID that matches with transition ID in osw file. "chromatogramIndex" that lists indices of chromatograms in mzML file.

Usage

```
chromatogramIdAsInteger(chromatogramHeader)
```

Arguments

chromatogramHeader	(dataframe)

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

constrainSimCpp *Constrain similarity matrix with a mask*

Description

Constrain similarity matrix with a mask

Usage

```
constrainSimCpp(sim, MASK, samples4gradient = 100)
```

Arguments

sim	(matrix) A numeric matrix. Input similarity matrix.
MASK	(matrix) A numeric matrix. Masked indices have non-zero values.
samples4gradient	(numeric) This paarameter modulates penalization of masked indices.

Value

s_new (matrix) A constrained similarity matrix.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
sim <- matrix(c(-2, 10, -2, -2, -2, -2, 10, -2, 10, -2, -2, -2, -2, -2, -2, 10, 10, -2,-2, -2),
 4, 5, byrow = FALSE)
MASK <- matrix(c(0.000, 0.000, 0.707, 1.414, 0.000, 0.000, 0.000, 0.707, 0.707, 0.000,
0.000, 0.000, 1.414, 0.707, 0, 0, 2.121, 1.414, 0, 0), 4, 5, byrow = FALSE)
constrainSimCpp(sim, MASK, 10)
matrix(c(-2, 10, -3.414, -4.828, -2, -2, 10, -3.414, 8.586, -2, -2, -2, -4.828,
-3.414, -2, 10, 5.758, -4.828, -2, -2), 4, 5, byrow = FALSE)
```

---

createMZML *Create an mzML file*

Description

Writes an mzML file having chromatograms and their native IDs.

Usage

```
createMZML(ropenms, filename, XICs, transitionIDs)
```

Arguments

ropenms	(pyopenms module) get this python module through get_ropenms().
filename	(string) name of the mzML file to be written. Extension should be .chrom.mzML.
XICs	(list of list of data-frames) list of extracted ion chromatograms of all precursors.
transitionIDs	(list of integer) length must be the same as of XICs.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-06-06

See Also

get_ropenms, addXIC

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
filename <- paste0(dataPath, "/xics/hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML")
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR)
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]]
nativeIds <- list(27706:27711)
## Not run:
ropenms <- get_ropenms(condaEnv = "envName")
createMZML(ropenms, "testfile.chrom.mzML", XICs, nativeIds)
mzR::chromatogramHeader(mzR::openMSfile("testfile.chrom.mzML", backend = "pwiz"))
file.remove("testfile.chrom.mzML")

## End(Not run)
```

---

createSqMass *Create an sqMass file*

Description

Writes a sqMass file having chromatograms and their native IDs.

Usage

```
createSqMass(filename, XICs, transitionIDs, lossy)
```

Arguments

filename	(string) name of the mzML file to be written. Extension should be .chrom.sqMass.
XICs	(list of list of data-frames) list of extracted ion chromatograms of all precursors.
transitionIDs	(list of integer) length must be the same as of XICs.
lossy	(logical) if TRUE, time and intensity are lossy-compressed.

Details

- compression is one of 0 = no, 1 = zlib, 2 = np-linear, 3 = np-slof, 4 = np-pic, 5 = np-linear + zlib, 6 = np-slof + zlib, 7 = np-pic + zlib
- data_type is one of 0 = mz, 1 = int, 2 = rt
- data contains the raw (blob) data for a single data array

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

createMZML, blobXICs

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR)
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]]
XICs <- list(XICs[[1]], XICs[[1]])
nativeIds <- list(27706:27711, 1:6)
sqName <- "testfile.chrom.sqMass"
## Not run:
createSqMass(sqName, XICs, nativeIds, TRUE)
con <- DBI::dbConnect(RSQLite::SQLite(), dbname = sqName)
XIC_group <- extractXIC_group2(con, 0:5)
DBI::dbDisconnect(con)
file.remove(sqName)

## End(Not run)
```

DIAlignR *DIAlignR*

Description

This package implements dynamic programming with affine gap penalty to find a highest-scoring scoring path. A hybrid approach of global alignment through MS2 features and local alignment with MS2 chromatograms is implemented in this tool.

Author(s)

Shubham Gupta, Hannes Rost

`dialignrLoess` *Modified loess for condition handling*

Description

Modified loess for condition handling

Usage

```
dialignrLoess(RUNS_RT, spanvalue)
```

Arguments

`RUNS_RT`	(data-frame) must have three calumns: transition_group_id, RT.eXp, and RT.ref.
`spanvalue`	(numeric) spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

Value

An object of class "loess" or "lm".

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getLinearfit`, `getLOESSfit`

Examples

```
df <- data.frame("transition_group_id" = 1:10, "RT.eXp" = 2:11, "RT.ref" = 10:19)
## Not run:
dialignrLoess(df, 0.1)
dialignrLoess(df[1:4,], 0.1)

## End(Not run)
```

doAffineAlignmentCpp *Perform affine global and overlap alignment on a similarity matrix*

Description

Perform affine global and overlap alignment on a similarity matrix

Usage

```
doAffineAlignmentCpp(sim, go, ge, OverlapAlignment)
```

Arguments

sim	(NumericMatrix) A numeric matrix with similarity values of two sequences or signals.
go	(numeric) Penalty for introducing first gap in alignment.
ge	(numeric) Penalty for introducing subsequent gaps in alignment.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.

Value

affineAlignObj (S4class) An object from C++ class of AffineAlignObj.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
# Get sequence similarity of two DNA strings
Match=10; MisMatch=-2
seq1 = "GCAT"; seq2 = "CAGTG"
s <- getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
objAffine_Global <- doAffineAlignmentCpp(s, 22, 7, FALSE)
slot(objAffine_Global, "score") # -2  -4  -6  4 -18
objAffine_Olap <- doAffineAlignmentCpp(s, 22, 7, TRUE)
slot(objAffine_Olap, "score") # 0 10 20 18 18 18

Match=10; MisMatch=-2
seq1 = "CAT"; seq2 = "CAGTG"
s <- getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
objAffine_Global <- doAffineAlignmentCpp(s, 22, 7, FALSE)
slot(objAffine_Global, "score") # 10  20  -2  -9 -11
objAffine_Olap <- doAffineAlignmentCpp(s, 22, 7, TRUE)
slot(objAffine_Olap, "score") # 10 20 18 18 18
```

```
Match=10; MisMatch=-2
seq1 = "CA"; seq2 = "AG"
s <- getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
objAffine_Global <- doAffineAlignmentCpp(s, 22, 7, FALSE)
slot(objAffine_Global, "simScore_forw") # -4
```

`doAlignmentCpp` *Perform non-affine global and overlap alignment on a similarity matrix*

Description

Perform non-affine global and overlap alignment on a similarity matrix

Usage

```
doAlignmentCpp(sim, gap, OverlapAlignment)
```

Arguments

`sim`	(NumericMatrix) A numeric matrix with similarity values of two sequences or signals.
`gap`	(double) Penalty for introducing gaps in alignment.
`OverlapAlignment`	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.

Value

AlignObj (S4class) An object from C++ class of AlignObj.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
# Get sequence similarity of two DNA strings
Match=10; MisMatch=-2
seq1 = "GCAT"; seq2 = "CAGTG"
s <- getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
obj_Global <- doAlignmentCpp(s, 22, FALSE)
slot(obj_Global, "score") # -2 -4 -6 4 -18
obj_Olap <- doAlignmentCpp(s, 22, TRUE)
slot(obj_Olap, "score") # 0 10 20 18 18 18

Match=1; MisMatch=-1
```

```
seq1 = "TTTC"; seq2 = "TGC"
s <- getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
obj_Global <- doAlignmentCpp(s, 2, FALSE)
slot(obj_Global, "optionalPaths")
matrix(data = c(1,1,1,1,1,1,1,1,1,2,1,2,1,3,3,1,1,3,6,3), nrow = 5, ncol =4, byrow = TRUE)
slot(obj_Global, "M_forw")
matrix(data = c(0,-2,-4,-6,-2,-7,-22,-45,-4,-20,-72,-184,-6,-41,-178,-547,-8,-72,-366,-1274),
 nrow = 5, ncol =4, byrow = TRUE)
```

`extractXIC_group` *Extract XICs of chromIndices*

Description

Extracts XICs using mz object. Each chromatogram represents a transition of precursor.

Usage

```
extractXIC_group(mz, chromIndices)
```

Arguments

`mz`	(mzRpwiz object)
`chromIndices`	(vector of Integers) Indices of chromatograms to be extracted.

Value

A list of data-frames. Each data frame has elution time and intensity of fragment-ion XIC.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
mzmlName<-paste0(dataPath,"/xics/hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML")
mz <- mzR::openMSfile(mzmlName, backend = "pwiz")
chromIndices <- c(37L, 38L, 39L, 40L, 41L, 42L)
## Not run:
XIC_group <- extractXIC_group(mz, chromIndices)

## End(Not run)
```

`extractXIC_group2` *Extract XICs of chromIndices*

Description

DATA_TYPE is one of 0 = mz, 1 = intensity, 2 = rt Extracts XICs using connection to sqMass file Each chromatogram represents a transition of precursor.

Usage

```
extractXIC_group2(con, chromIndices)
```

Arguments

`chromIndices`	(vector of Integers) Indices of chromatograms to be extracted.
`mz`	(SQLiteConnection object)

Value

A list of data-frames. Each data frame has elution time and intensity of fragment-ion XIC.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
sqName <- paste0(dataPath,"/xics/hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.sqMass")
chromIndices <- c(36L, 37L, 38L, 39L, 40L, 41L)
## Not run:
con <- DBI::dbConnect(RSQLite::SQLite(), dbname = sqName)
XIC_group <- extractXIC_group2(con, chromIndices)
DBI::dbDisconnect(con)

## End(Not run)
```

`fetchAnalytesInfo` *Fetch features of analytes*

Description

Get a data-frame of analytes' transition_group_ids, their OpenSwath features, chromatogram indices and associated FDR-scores.

Usage

```
fetchAnalytesInfo(
  oswName,
  maxFdrQuery,
  oswMerged,
  analytes,
  filename,
  runType = "DIA_Proteomics",
  analyteInGroupLabel = FALSE
)
```

Arguments

`oswName`	(char) path to the osw file.
`maxFdrQuery`	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
`oswMerged`	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
`analytes`	(vector of strings) transition_group_ids for which features are to be extracted. analyteInGroupLabel must be set according the pattern used here.
`filename`	(data-frame) Should be from the RUN.FILENAME column from osw files.
`runType`	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
`analyteInGroupLabel`	(logical) TRUE for getting analytes as PRECURSOR.GROUP_LABEL from osw file. FALSE for fetching analytes as PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.

Value

(data-frames) Data-frame has following columns:

`transition_group_id`	(string) it is either fetched from PRECURSOR.GROUP_LABEL or a combination of PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.
`filename`	(string) as mentioned in RUN table of osw files.

RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
delta_rt	(numeric) as in FEATURE.DELTA_RT of osw files.
assay_RT	(numeric) library retention time as in PRECURSOR.LIBRARY_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.
leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.
transition_id	(integer) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
filenames <- getRunNames(dataPath = dataPath)
oswName <- paste0(dataPath,"/osw/merged.osw")
## Not run:
analytesInfo <- fetchAnalytesInfo(oswName, maxFdrQuery = 0.05, oswMerged = TRUE,
 analytes = c("19051_KLIVTSEGC[160]FK/2"), filename = filenames$filename[2],
  runType = "DIA_Proteomics", analyteInGroupLabel = TRUE)
analytesInfo <- fetchAnalytesInfo(oswName, maxFdrQuery = 0.05, oswMerged = TRUE,
 analytes = c("IHFLSPVRPFTLTPGDEEESFIQLITPVR_3"), filename = filenames$filename[3],
  runType = "DIA_Proteomics", analyteInGroupLabel = FALSE)

## End(Not run)
```

---

fetchFeaturesFromRun *Get features from a feature file.*

Description

Get a data-frame of OpenSwath features that contains retention time, intensities, boundaries etc.

Usage

```
fetchFeaturesFromRun(
  filename,
  runID,
  maxFdrQuery = 1,
  maxIPFFdrQuery = 1,
  runType = "DIA_Proteomics"
)
```

Arguments

filename	(string) Path to the feature file.
runID	(string) id in RUN.ID column of the feature file.
maxFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
maxIPFFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_IPF.QVALUE less than itself. (For PTM IPF use)
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

(data-frames) Data-frame has following columns:

transition_group_id	(integer) a unique id for each precursor.
RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.
leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, getFeatures, getFeaturesQuery

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
## Not run:
featuresInfo <- fetchFeaturesFromRun(fileInfo$featureFile[1], fileInfo$spectraFileID[1],
 maxFdrQuery = 0.05)
dim(featuresInfo) # 211  8

## End(Not run)
```

`fetchPeptidesInfo` *Get scores of all peptides*

Description

Return a scores, pvalues, and qvalues for all peptides from the osw file.

Usage

```
fetchPeptidesInfo(oswName, runType, context)
```

Arguments

`oswName`	(char) path to the osw file.
`runType`	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
`context`	(string) Context used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".

Value

(dataframe) with following columns:

`peptide_id`	(integer) a unique id for each precursor.
`run`	(character) as in SCORE_PEPTIDE.RUN_ID of osw files.
`score`	(numeric) as in SCORE_PEPTIDE.SCORE of osw files.
`pvalue`	(numeric) as in SCORE_PEPTIDE.PVALUE of osw files.
`qvalue`	(numeric) as in SCORE_PEPTIDE.QVALUE of osw files.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getPeptideQuery, getPeptideScores

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
oswName <- fileInfo[["featureFile"]][1]
## Not run:
precursorsInfo <- fetchPeptidesInfo(fileInfo, runType = "DIA_Proteomics", context = "experiment-wide")

## End(Not run)
```

---

fetchPeptidesInfo2 *Get scores of all peptides*

Description

Return a scores, pvalues, and qvalues for all peptides from the osw file.

Usage

```
fetchPeptidesInfo2(oswName, runType, context, runID)
```

Arguments

oswName	(char) path to the osw file.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
context	(string) Context used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".

Value

(dataframe) with following columns:

peptide_id	(integer) a unique id for each precursor.
run	(character) as in SCORE_PEPTIDE.RUN_ID of osw files.
score	(numeric) as in SCORE_PEPTIDE.SCORE of osw files.
pvalue	(numeric) as in SCORE_PEPTIDE.PVALUE of osw files.
qvalue	(numeric) as in SCORE_PEPTIDE.QVALUE of osw files.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getPeptideQuery, getPeptideScores

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
oswName <- fileInfo[["featureFile"]][1]
## Not run:
precursorsInfo <- fetchPeptidesInfo(fileInfo, runType = "DIA_Proteomics", context = "experiment-wide")

## End(Not run)
```

---

fetchPrecursorsInfo *Get precursors from a feature file*

Description

Get a data-frame of analytes' transition_group_id, transition_ids, peptide_id and amino-acid sequences.

Usage

```
fetchPrecursorsInfo(
  filename,
  runType = "DIA_Proteomics",
  selectIDs = NULL,
  context = "global",
  maxPeptideFdr = 0.05,
  level = "Peptide",
  useIdentifying = FALSE
)
```

Arguments

filename	(string) Should be from the RUN.FILENAME column from osw files.
runType	(string) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
selectIDs	(integer) a vector of integers.
context	(string) Context used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".
maxPeptideFdr	(numeric) A numeric value between 0 and 1. It is used to filter peptides from osw file which have SCORE_PEPTIDE.QVALUE less than itself.
level	(string) Apply maxPeptideFDR on Protein as well if specified as "Protein". Default: "Peptide".
useIdentifying	(logical) Set TRUE to use identifying transitions in alignment. (DEFAULT: FALSE)

Value

(data-frames) Data-frame has following columns:

`transition_group_id`	(integer) a unique id for each precursor.
`transition_id`	(list) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.
`peptide_id`	(integer) a unique id for each peptide. A peptide can have multiple precursors.
`sequence`	(string) amino-acid sequence of the precursor with possible modifications.
`charge`	(integer) charge on the precursor.
`group_label`	(string) TODO Figure it out.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getRunNames`, `getPrecursors`, `getPrecursorsQuery`

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
filename <- paste0(dataPath,"/osw/merged.osw")
## Not run:
precursorsInfo <- fetchPrecursorsInfo(filename, runType = "DIA_Proteomics", context = "experiment-wide")
dim(precursorsInfo) # 234  6

## End(Not run)
```

---

fetchTransitionsFromRun

Get transitions from a feature file

Description

Get a data-frame of OpenSwath features that contains retention time, transition intensities, boundaries etc.

Usage

```
fetchTransitionsFromRun(
  filename,
  runID,
  maxFdrQuery = 1,
  runType = "DIA_Proteomics"
)
```

Arguments

filename	(string) Path to the feature file.
runID	(string) id in RUN.ID column of the feature file.
maxFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".

Value

(data-frames) Data-frame has following columns:

transition_group_id	(integer) a unique id for each precursor.
RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
intensity	(list) of peak intensities as in FEATURE_TRANSITION.AREA_INTENSITY of osw files.
leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, getTransitions, getTransitionsQuery

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
## Not run:
transitionsInfo <- fetchTransitionsFromRun(fileInfo$featureFile[1], fileInfo$spectraFileID[1],
 maxFdrQuery = 0.05)
dim(transitionsInfo) # 211  8

## End(Not run)
```

`filenamesFromMZML` *Get mzML filenames from the directory.*

Description

Reads all mzML names avaialble in the directory.

Usage

```
filenamesFromMZML(dataPath, chromFile)
```

Arguments

`dataPath`	(char) Path to xics and osw directory.

Value

A dataframe with two columns:

`runName`	(string) contain respective mzML names without extension.
`chromatogramFile`	(string) Path to the chromatogram file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
## Not run:
filenamesFromMZML(dataPath)

## End(Not run)
```

`filenamesFromOSW` *Get mzML filenames from osw RUN table.*

Description

Get mzML filenames from osw RUN table.

Usage

```
filenamesFromOSW(dataPath, pattern)
```

Arguments

`dataPath`	(char) path to xics and osw directory.
`pattern`	(char) must be either *.osw or *merged.osw .

Value

A dataframe with three columns:

`spectraFile`	(string) as mentioned in RUN table of osw files.
`spectraFileID`	(string) ID in RUN table of osw files.
`featureFile`	(string) Path to the feature file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
## Not run:
filenamesFromOSW(dataPath, "*.osw")
filenamesFromOSW(dataPath, "*merged.osw")

## End(Not run)
```

getAlignedFigs *Plot aligned XICs group for a specific peptide.*

Description

AlignObj is the output from getAlignObjs fucntion. This function prepares ggplot objects from AlignObj.

Usage

```
getAlignedFigs(
  AlignObj,
  XICs.ref,
  XICs.eXp,
  refPeakLabel,
  annotatePeak = FALSE
)
```

Arguments

AlignObj	(S4 object)
XICs.ref	(list) List of extracted ion chromatograms (dataframe) from reference run. The dataframe has two columns: first column is for time and second column indicates intensity.
XICs.eXp	(list) List of extracted ion chromatograms (dataframe) from experiment run.The dataframe has two columns: first column is for time and second column indicates intensity.
refPeakLabel	(numeric vector) It contains peak apex, left width and right width.
annotatePeak	(logical) TRUE: Peak boundaries and apex will be highlighted.

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
AlignObjOutput <- getAlignObjs(analytes = 4618L, runs, dataPath = dataPath)
AlignObj <- AlignObjOutput[[2]][["4618"]][[1]][["AlignObj"]]
XICs.ref <- AlignObjOutput[[2]][["4618"]][[1]][["ref"]]
XICs.eXp <- AlignObjOutput[[2]][["4618"]][[1]][["eXp"]]
refPeakLabel <- AlignObjOutput[[2]][["4618"]][[1]][["peak"]]
## Not run:
getAlignedFigs(AlignObj, XICs.ref, XICs.eXp, refPeakLabel)

## End(Not run)
```

---

getAlignedIndices *Get aligned indices.*

Description

This function aligns XICs of reference and experiment runs. It produces aligned indices between refernce run and experiment run.

Usage

```
getAlignedIndices(
  XICs.ref,
  XICs.eXp,
  globalFit,
  alignType,
  adaptiveRT,
  normalization,
  simMeasure,
  goFactor,
  geFactor,
  cosAngleThresh,
  OverlapAlignment,
  dotProdThresh,
  gapQuantile,
  kerLen,
  hardConstrain,
  samples4gradient,
  objType = "light"
)
```

Arguments

XICs.ref	List of extracted ion chromatograms from reference run.

XICs.eXp	List of extracted ion chromatograms from experiment run.
globalFit	Linear or loess fit object between reference and experiment run.
alignType	Available alignment methods are "global", "local" and "hybrid".
adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(character) Must be selected from "mean", "l2".
simMeasure	(string) Must be selected from dotProduct, cosineAngle, crossCorrelation, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) If FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
objType	(char) Must be selected from light, medium and heavy.

Value

(data-frame) Aligned indices of reference and experiment runs. Gaps are introduced as NA.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignChromatogramsCpp, getAlignObj

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(oswFiles_DIAlignR, package="DIAlignR")
run1 <- "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"
run2 <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run1]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run2]][["4618"]]
globalFit <- getGlobalAlignment(oswFiles_DIAlignR, ref = "run1", eXp = "run2",
  fitType = "loess", maxFdrGlobal = 0.05, spanvalue = 0.1)
adaptiveRT <- 77.82315 #3.5*globalFit$s
## Not run:
getAlignedIndices(XICs.ref, XICs.eXp, globalFit, alignType = "hybrid",
 adaptiveRT = adaptiveRT, normalization = "mean",
  simMeasure = "dotProductMasked", goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3,
 OverlapAlignment = TRUE, dotProdThresh = 0.96, gapQuantile = 0.5, kerLen = 9L, hardConstrain = FALSE,
  samples4gradient = 100)

## End(Not run)
```

`getAlignedTimes` *Get aligned Retention times.*

Description

This function aligns XICs of reference and experiment runs. It produces aligned retention times between refernce run and experiment run.

Usage

```
getAlignedTimes(
  XICs.ref,
  XICs.eXp,
  globalFit,
  alignType,
  adaptiveRT,
  normalization,
  simMeasure,
  goFactor,
  geFactor,
  cosAngleThresh,
  OverlapAlignment,
  dotProdThresh,
  gapQuantile,
  kerLen,
  hardConstrain,
  samples4gradient,
  objType = "light"
)
```

Arguments

XICs.ref	List of extracted ion chromatograms from reference run.
XICs.eXp	List of extracted ion chromatograms from experiment run.
globalFit	Linear or loess fit object between reference and experiment run.
alignType	Available alignment methods are "global", "local" and "hybrid".
adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(character) Must be selected from "mean", "l2".
simMeasure	(string) Must be selected from dotProduct, cosineAngle, crossCorrelation, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) If FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
objType	(char) Must be selected from light, medium and heavy.

Value

(list) the first element corresponds to the aligned reference time, the second element is the aligned experiment time.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignChromatogramsCpp, getAlignObj

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(oswFiles_DIAlignR, package="DIAlignR")
run1 <- "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"
run2 <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run1]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run2]][["4618"]]
RUNS_RT <- getRTdf(oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
globalFit <- loess(RT.eXp ~ RT.ref, data = RUNS_RT, span = 0.1, control=loess.control(surface="direct"))
adaptiveRT <- 77.82315 #3.5*globalFit$s
getAlignedTimes(XICs.ref, XICs.eXp, globalFit, alignType = "hybrid",
 adaptiveRT = adaptiveRT, normalization = "mean",
  simMeasure = "dotProductMasked", goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3,
 OverlapAlignment = TRUE, dotProdThresh = 0.96, gapQuantile = 0.5, kerLen = 9L, hardConstrain = FALSE,
  samples4gradient = 100)
```

---

getAlignedTimesCpp *Get aligned indices from MS2 extracted-ion chromatograms(XICs) pair.*

Description

Get aligned indices from MS2 extracted-ion chromatograms(XICs) pair.

Usage

```
getAlignedTimesCpp(
  l1,
  l2,
  kernelLen,
  polyOrd,
  alignType,
  adaptiveRT,
  normalization,
  simType,
  Bp,
  goFactor = 0.125,
  geFactor = 40,
  cosAngleThresh = 0.3,
  OverlapAlignment = TRUE,
  dotProdThresh = 0.96,
  gapQuantile = 0.5,
  kerLen = 9L,
  hardConstrain = FALSE,
  samples4gradient = 100
)
```

Arguments

l1	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
l2	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
kernelLen	(integer) length of filter. Must be an odd number.
polyOrd	(integer) TRUE: remove background from peak signal using estimated noise levels.
alignType	(char) A character string. Available alignment methods are "global", "local" and "hybrid".
adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(char) A character string. Normalization must be selected from (L2, mean or none).
simType	(char) A character string. Similarity type must be selected from (dotProductMasked, dotProduct, cosineAngle, cosine2Angle, euclideanDist, covariance, correlation, crossCorrelation). Mask = s > quantile(s, dotProdThresh) AllowDotProd= [Mask × cosine2Angle + (1 - Mask)] > cosAngleThresh s_new= s × AllowDotProd
Bp	(numeric) Timepoint mapped by global fit for tA.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) if false; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.

Value

NumericMatrix Aligned indices of l1 and l2.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR
XICs.ref <- lapply(XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
XICs.eXp <- lapply(XICs[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
Bp <- seq(4964.752, 5565.462, length.out = nrow(XICs.ref[[1]]))
time <- getAlignedTimesCpp(XICs.ref, XICs.eXp, 11, 4, alignType = "hybrid", adaptiveRT = 77.82315,
 normalization = "mean", simType = "dotProductMasked", Bp = Bp,
 goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3, OverlapAlignment = TRUE,
 dotProdThresh = 0.96, gapQuantile = 0.5, hardConstrain = FALSE, samples4gradient = 100)
```

---

getAlignedTimesFast *Get aligned Retention times.*

Description

This function aligns XICs of reference and experiment runs. It produces aligned retention times between refernce run and experiment run.

Usage

```
getAlignedTimesFast(XICs.ref, XICs.eXp, globalFit, adaptiveRT, params)
```

Arguments

XICs.ref	List of extracted ion chromatograms from reference run.
XICs.eXp	List of extracted ion chromatograms from experiment run.
globalFit	Linear or loess fit object between reference and experiment run.
adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.

Value

(matrix) the first column corresponds to the aligned reference time, the second column is the aligned experiment time.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2021) + GPL-3 Date: 2021-01-02

See Also

alignChromatogramsCpp, getAlignObj

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(oswFiles_DIAlignR, package="DIAlignR")
run1 <- "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"
run2 <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
XICs.ref <- lapply(XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run1]][["4618"]], as.matrix)
XICs.eXp <- lapply(XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run2]][["4618"]], as.matrix)
params <- paramsDIAlignR()
params[["globalAlignment"]] <- "linear"
globalFit <- getGlobalAlignment(oswFiles_DIAlignR, ref = "run2", eXp = "run0",
 fitType = params[["globalAlignment"]], maxFdrGlobal = 0.05, spanvalue = 0.1)
adaptiveRT <- 77.82315 #3.5*getRSE(globalFit, params[["globalAlignment"]])
globalFit <- coef(globalFit)
getAlignedTimesFast(XICs.ref, XICs.eXp, globalFit, adaptiveRT, params)
```

---

getAlignObj *Outputs AlignObj from an alignment of two XIC-groups*

Description

Outputs AlignObj from an alignment of two XIC-groups

Usage

```
getAlignObj(
  XICs.ref,
  XICs.eXp,
  globalFit,
  alignType,
  adaptiveRT,
  normalization,
  simType,
  goFactor,
  geFactor,
  cosAngleThresh,
  OverlapAlignment,
  dotProdThresh,
  gapQuantile,
  kerLen,
  hardConstrain,
  samples4gradient,
  objType = "light"
)
```

Arguments

XICs.ref	List of extracted ion chromatograms from reference run.
XICs.eXp	List of extracted ion chromatograms from experiment run.
globalFit	Linear or loess fit object between reference and experiment run.
alignType	Available alignment methods are "global", "local" and "hybrid".
adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(character) Must be selected from "mean", "l2".
simType	(string) Must be selected from dotProduct, cosineAngle, crossCorrelation, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) If FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
objType	(char) Must be selected from light, medium and heavy.

Value

A S4 object. Three most-important slots are:

indexA_aligned	(integer) aligned indices of reference run.
indexB_aligned	(integer) aligned indices of experiment run.
score	(numeric) cumulative score of alignment.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignChromatogramsCpp

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(oswFiles_DIAlignR, package="DIAlignR")
run1 <- "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"
run2 <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run1]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run2]][["4618"]]
RUNS_RT <- getRTdf(oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
globalFit <- loess(RT.eXp ~ RT.ref, data = RUNS_RT, span = 0.1, control=loess.control(surface="direct"))
AlignObj <- getAlignObj(XICs.ref, XICs.eXp, globalFit, alignType = "hybrid", adaptiveRT = 77.82315,
 normalization = "mean", simType = "dotProductMasked", goFactor = 0.125,
  geFactor = 40, cosAngleThresh = 0.3, OverlapAlignment = TRUE, dotProdThresh = 0.96,
 gapQuantile = 0.5, kerLen = 9L, hardConstrain = FALSE, samples4gradient = 100, objType = "light")
```

---

getAlignObjs *AlignObj for analytes between a pair of runs*

Description

This function expects osw and xics directories at dataPath. It first reads osw files and fetches chromatogram indices for each requested analyte. It then align XICs of each analyte to its reference XICs. AlignObj is returned which contains aligned indices and cumulative score along the alignment path.

Usage

```
getAlignObjs(
  analytes,
  runs,
  dataPath = ".",
  refRun = NULL,
  oswMerged = TRUE,
  params = paramsDIAlignR(),
  objType = "light"
)
```

Arguments

analytes	(vector of integers) transition_group_ids for which features are to be extracted.
runs	(string) names of xics file without extension.
dataPath	(string) path to xics and osw directory.
refRun	(string) reference for alignment. If no run is provided, m-score is used to select reference run.

oswMerged	(logical) TRUE if merged file from pyprophet is used.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
objType	(char) Must be selected from light, medium and heavy.

Value

A list of fileInfo and AlignObjs. Each AlignObj is an S4 object. Three most-important slots are:

indexA_aligned	(integer) aligned indices of reference run.
indexB_aligned	(integer) aligned indices of experiment run.
score	(numeric) cumulative score of alignment.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

References

Gupta S, Ahadi S, Zhou W, Röst H. "DIAlignR Provides Precise Retention Time Alignment Across Distant Runs in DIA and Targeted Proteomics." Mol Cell Proteomics. 2019 Apr;18(4):806-817. doi: https://doi.org/10.1074/mcp.TIR118.001132 Epub 2019 Jan 31.

See Also

plotAlignedAnalytes, getRunNames, getFeatures, getXICs4AlignObj, getAlignObj

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
runs <- c("hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt",
 "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
analytes <- c(32L, 898L, 2474L)
AlignObjOutput <- getAlignObjs(analytes, runs, dataPath = dataPath)
plotAlignedAnalytes(AlignObjOutput)
```

`getAnalytesQuery` *Generate SQL query to fetch limited information from osw files.*

Description

Generate SQL query to fetch limited information from osw files.

Usage

```
getAnalytesQuery(
  maxFdrQuery,
  oswMerged = TRUE,
  filename = NULL,
  runType = "DIA_Proteomics",
  analyteInGroupLabel = FALSE
)
```

Arguments

`maxFdrQuery`	(numeric) value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
`oswMerged`	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
`filename`	(string) as mentioned in RUN table of osw files..
`runType`	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
`analyteInGroupLabel`	(logical) TRUE for getting analytes as PRECURSOR.GROUP_LABEL from osw file.

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

`getOswAnalytes`

getBaseGapPenaltyCpp *Calculates gap penalty for dynamic programming based alignment.*

Description

This function outputs base gap-penalty depending on SimType used. In case of getting base gap-penalty from similarity matrix distribution, gapQuantile will be used to pick the value.

Usage

```
getBaseGapPenaltyCpp(sim, SimType, gapQuantile = 0.5)
```

Arguments

sim	(matrix) A numeric matrix. Input similarity matrix.
SimType	(char) A character string. Similarity type must be selected from (dotProductMasked, dotProduct, cosineAngle, cosine2Angle, euclideanDist, covariance, correlation, crossCorrelation).
gapQuantile	(numeric) Must be between 0 and 1.

Value

baseGapPenalty (numeric).

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
sim <- matrix(c(-12, 1.0, 12, -2.3, -2, -2, 1.07, -2, 1.80, 2, 22, 42, -2, -1.5, -2, 10), 4, 4,
 byrow = FALSE)
getBaseGapPenaltyCpp(sim, "dotProductMasked", 0.5) # -0.25
```

getChildFeature *Transform features to child time-domain*

Description

This function transforms the peaks' times to child run's time-domain. The feature intensity is calculated with appropriate method stated in params. Internal missing values are not allowed in timeParent.

Usage

```
getChildFeature(XICs, alignedVec, df.ref, df.eXp, i.ref, i.eXp, params)
```

Arguments

XICs	(list of data-frames) extracted ion chromatograms from the child run.
alignedVec	(data-frame) aligned parent time-vectors. Must have five columns
df.ref	(data-frame) contains reference-run features to be transformed. It has a format of getFeatures output.
df.eXp	(data-frame) contains experiment-run features to be transformed. It has a format of getFeatures output.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.

Value

(data-frame) this has a format of getFeatures output.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

trfrParentFeature, getNodeRun

Examples

```
data(masterXICs_DIAlignR, package="DIAlignR")
newXICs <- masterXICs_DIAlignR
params <- paramsDIAlignR()
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
df.ref <- features$run1[features$run1$transition_group_id == 4618L, ]
df.eXp <- features$run2[features$run2$transition_group_id == 4618L, ]
## Not run:
getChildFeature(newXICs[[1]], newXICs[[2]], df.ref, df.eXp, params)

## End(Not run)
```

getChildXICpp *Get child chromatogram from two parent chromatogram*

Description

Get child chromatogram from two parent chromatogram

Usage

```
getChildXICpp(
  l1,
  l2,
  kernelLen,
  polyOrd,
  alignType,
  adaptiveRT,
  normalization,
  simType,
  Bp,
  goFactor = 0.125,
  geFactor = 40,
  cosAngleThresh = 0.3,
  OverlapAlignment = TRUE,
  dotProdThresh = 0.96,
  gapQuantile = 0.5,
  kerLen = 9L,
  hardConstrain = FALSE,
  samples4gradient = 100,
  wRef = 0.5,
  splineMethod = "natural",
  mergeStrategy = "avg",
  keepFlanks = TRUE
)
```

Arguments

l1	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
l2	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
kernelLen	(integer) length of filter. Must be an odd number.
polyOrd	(integer) TRUE: remove background from peak signal using estimated noise levels.
alignType	(char) A character string. Available alignment methods are "global", "local" and "hybrid".

adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(char) A character string. Normalization must be selected from (L2, mean or none).
simType	(char) A character string. Similarity type must be selected from (dotProductMasked, dotProduct, cosineAngle, cosine2Angle, euclideanDist, covariance, correlation, crossCorrelation). Mask = s > quantile(s, dotProdThresh) AllowDotProd= [Mask × cosine2Angle + (1 - Mask)] > cosAngleThresh s_new= s × AllowDotProd
Bp	(numeric) Timepoint mapped by global fit for tA.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) if false; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
wRef	(numeric) Weight of the reference XIC. Must be between 0 and 1.
splineMethod	(string) must be either "fmm" or "natural".
mergeStrategy	(string) must be either ref, avg, refStart or refEnd.
keepFlanks	(logical) TRUE: Flanking chromatogram is not removed.

Value

(List) of chromatograms and their aligned time vectors.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 Date: 2021-01-08

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR
XICs.ref <- lapply(XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
XICs.eXp <- lapply(XICs[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
Bp <- seq(4964.752, 5565.462, length.out = nrow(XICs.ref[[1]]))
chrom <- getChildXICpp(XICs.ref, XICs.eXp, 11L, 4L, alignType = "hybrid", adaptiveRT = 77.82315,
 normalization = "mean", simType = "dotProductMasked", Bp = Bp,
 goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3, OverlapAlignment = TRUE,
 dotProdThresh = 0.96, gapQuantile = 0.5, hardConstrain = FALSE, samples4gradient = 100,
 wRef = 0.5, keepFlanks= TRUE)
```

---

getChildXICs *Develop child XICs for precursors*

Description

This function performs the chromatogram alignment of all precursors across runA and runB. Aligned chromatograms are merged into a child chromatogram. Aligned time vector and resulting child time vector for each precursor is also returned.

Usage

```
getChildXICs(
  runA,
  runB,
  fileInfo,
  features,
  mzPntrs,
  precursors,
  prec2chromIndex,
  refRun,
  peptideScores,
  params,
  applyFun = lapply
)
```

Arguments

runA	(string) name of a run to be merged with runB. Must be in the rownames of fileInfo.
runB	(string) name of a run to be merged with runA. Must be in the rownames of fileInfo.
fileInfo	(data-frame) output of getRunNames.
features	(list of data-frames) contains features and their properties identified in each run.
mzPntrs	(list) a list of mzRpwiz.

precursors	(data-frame) atleast two columns transition_group_id and transition_ids are required.
prec2chromIndex	(list) a list of dataframes having following columns: transition_group_id: it is PRECURSOR.ID from osw file. chromatogramIndex: index of chromatogram in mzML file.
refRun	(integer) must be of the same length as of precursors. 1: reference is runA, 2: reference is runB.
peptideScores	(list of data-frames) each dataframe has scores of a peptide across all runs.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list) has three elements. The first element has child XICs for all the precursors. The second element has corresponding aligned time vectors. Third element contains Residual Standard Errors (RSE) of global fits amongst runA and runB.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXICs, getNodeRun

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
mzPntrs <- getMZMLpointers(fileInfo)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, runType = "DIA_Proteomics",
 context = "experiment-wide", maxPeptideFdr = 0.05)
precursors <- dplyr::arrange(precursors, .data$peptide_id, .data$transition_group_id)
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, oswMerged = TRUE, params[["runType"]], params[["context"]
peptideScores <- lapply(peptideIDs, function(pep) dplyr::filter(peptideScores, .data$peptide_id == pep))
names(peptideScores) <- as.character(peptideIDs)
prec2chromIndex <- getChromatogramIndices(fileInfo, precursors, mzPntrs)
var2 <- as.character(sapply(peptideIDs, function(p) precursors$transition_group_id[which(precursors$peptide_id =
refRun <- data.frame(rep(1L, length(peptideIDs)), var2)
mergedXICs <- getChildXICs(runA = "run1", runB = "run2", fileInfo, features, mzPntrs,
  precursors, prec2chromIndex, refRun, peptideScores, params)
for(con in mzPntrs) DBI::dbDisconnect(con)
```

getChromatogramIndices *Get chromatogram indices of precursors.*

Description

This function reads the header of chromatogram files. It then fetches chromatogram indices by matching transition_id(osw) with chromatogramID(xics).

Usage

```
getChromatogramIndices(fileInfo, precursors, mzPntrs, applyFun = lapply)
```

Arguments

fileInfo	(data-frame) Output of getRunNames function.
precursors	(data-frame) Atleast two columns transition_group_id and transition_ids are required.
mzPntrs	A list of mzRpwiz.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list) A list of dataframes having following columns:

transition_group_id
: (string) it is PRECURSOR.ID from osw file.

chromatogramIndex
: (integer) index of chromatogram in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

chromatogramIdAsInteger, mapPrecursorToChromIndices

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, context = "experiment-wide")
mzPntrs <- getMZMLpointers(fileInfo)
prec2chromIndex <- getChromatogramIndices(fileInfo, precursors, mzPntrs)
for(mz in mzPntrs) DBI::dbDisconnect(mz)
```

`getChromSimMatCpp` *Calculates similarity matrix of two fragment-ion chromatogram groups or extracted-ion chromatograms(XICs)*

Description

Calculates similarity matrix of two fragment-ion chromatogram groups or extracted-ion chromatograms(XICs)

Usage

```
getChromSimMatCpp(
  l1,
  l2,
  normalization,
  simType,
  cosAngleThresh = 0.3,
  dotProdThresh = 0.96,
  kerLen = 9L
)
```

Arguments

`l1`	(list) A list of vectors. Length should be same as of l2.
`l2`	(list) A list of vectors. Length should be same as of l1.
`normalization`	(char) A character string. Normalization must be selected from (L2, mean or none).
`simType`	(char) A character string. Similarity type must be selected from (dotProductMasked, dotProduct, cosineAngle, cosine2Angle, euclideanDist, covariance, correlation, crossCorrelation). Mask = s > quantile(s, dotProdThresh) AllowDotProd= [Mask × cosine2Angle + (1 - Mask)] > cosAngleThresh s_new= s × AllowDotProd
`cosAngleThresh`	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
`dotProdThresh`	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
`kerLen`	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.

Value

s (matrix) Numeric similarity matrix. Rows and columns expresses seq1 and seq2, respectively.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 Date: 2019-03-05

Examples

```
# Get similarity matrix of dummy chromatograms
r1 <- list(c(1.0,3.0,2.0,4.0), c(0.0,0.0,0.0,1.0), c(4.0,4.0,4.0,5.0))
r2 <- list(c(1.4,2.0,1.5,4.0), c(0.0,0.5,0.0,0.0), c(2.0,3.0,4.0,0.9))
round(getChromSimMatCpp(r1, r2, "L2", "dotProductMasked"), 3)
matrix(c(0.125, 0.162, 0.144, 0.208, 0.186, 0.240,
0.213, 0.313, 0.233, 0.273, 0.253, 0.346, 0.101, 0.208, 0.154, 0.273), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "L2", "dotProduct"), 3)
matrix(c(0.125, 0.162, 0.144, 0.208, 0.186,0.240, 0.213, 0.313, 0.233,
0.273, 0.253, 0.346, 0.101, 0.208, 0.154, 0.273), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "L2", "cosineAngle"), 3)
matrix(c(0.934, 0.999, 0.989, 0.986, 0.933, 0.989,
 0.983, 0.996, 0.994, 0.960, 0.995, 0.939, 0.450,
 0.761, 0.633, 0.772), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "L2", "cosine2Angle"), 3)
matrix(c(0.744, 0.998, 0.957, 0.944, 0.740, 0.956, 0.932,
0.985, 0.974, 0.842, 0.978, 0.764, -0.596, 0.158,
-0.200, 0.190), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "mean", "euclideanDist"), 3)
matrix(c(0.608, 0.614, 0.680, 0.434, 0.530, 0.742,
0.659, 0.641, 0.520, 0.541, 0.563, 0.511, 0.298,
0.375, 0.334, 0.355), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "L2", "covariance"), 3)
matrix(c(0.025, 0.028, 0.027, 0.028, 0.032, 0.034,
0.033, 0.034, 0.055, 0.051, 0.053, 0.051,
-0.004, 0.028, 0.012, 0.028), 4, 4, byrow = FALSE)

round(getChromSimMatCpp(r1, r2, "L2", "correlation"), 3)
matrix(c(0.874, 0.999, 0.974, 0.999, 0.923, 0.986, 0.993,
0.986, 0.991, 0.911, 0.990, 0.911, -0.065, 0.477,
0.214, 0.477), 4, 4, byrow = FALSE)
```

---

getFeatures *Get features from all feature files*

Description

Get a list of data-frame of OpenSwath features that contains retention time, intensities, boundaries etc.

Usage

```
getFeatures(
  fileInfo,
```

```
  maxFdrQuery = 0.05,
  maxIPFFdrQuery = 0.05,
  runType = "DIA_Proteomics",
  applyFun = lapply
)
```

Arguments

fileInfo	(data-frame) output of getRunNames function.
maxFdrQuery	(numeric) a numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
maxIPFFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_IPF.QVALUE less than itself. (For PTM IPF use)
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list of dataframes) each dataframe has following columns:

transition_group_id	(integer) a unique id for each precursor.
RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.
leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id. (If using 'DIA_IPF' runType, this will represent IPF's QVALUE)
ms2_m_score	(numeric) MS2 q-value of each feature associated with transition_group_id. (Will only be present if using 'DIA_IPF' runType)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, fetchPrecursorsInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
dim(features[[2]]) # 938  8
```

---

getFeaturesQuery *Get features from a SQLite file*

Description

Query is generated to identify features below a FDR cut-off from a run.

Usage

```
getFeaturesQuery(runType = "DIA_Proteomics")
```

Arguments

runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2020-04-07

See Also

fetchFeaturesFromRun

`getGlobalAlignMaskCpp` *Outputs a mask for constraining similarity matrix*

Description

This function takes in timeVectors from both runs, global-fit mapped values of end-points of first time vector and sample-length of window of no constraining. Outside of window, all elements of matrix are either equally weighted or weighted proportional to distance from window-boundry.

Usage

```
getGlobalAlignMaskCpp(tA, tB, tBp, noBeef = 50L, hardConstrain = FALSE)
```

Arguments

`tA`	(numeric) This vector has equally spaced timepoints of XIC A.
`tB`	(numeric) This vector has equally spaced timepoints of XIC B.
`tBp`	(numeric) mapping of tA to run B using some global fit.
`noBeef`	(integer) It defines the distance from the global fit, upto which no penalization is performed. The window length = 2*noBeef.
`hardConstrain`	(logical) if false; indices farther from noBeef distance are filled with distance from linear fit line.

Value

mask (matrix) A numeric matrix.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2019) + MIT Date: 2019-03-08

Examples

```
tA <- c(1707.6, 1711, 1714.5, 1717.9, 1721.3, 1724.7, 1728.1, 1731.5, 1734.9, 1738.4)
tB <- c(1765.7, 1769.1, 1772.5, 1775.9, 1779.3, 1782.7, 1786.2, 1789.6, 1793, 1796.4)
tBp <- c(1786.9, 1790.35, 1793.9, 1797.36, 1800.81, 1804.26, 1807.71, 1811.17, 1814.62, 1818.17)
noBeef <- 1
mask <- getGlobalAlignMaskCpp(tA, tB, tBp, noBeef, FALSE)
round(mask, 3)
matrix(c( 5.215,4.218,4.221,2.225,1.228,0,0,0,0.788, 1.785,
6.226,5.230,4.233,3.236,2.239,1.243,0,0,0, 0.774), nrow = 2, ncol = 10, byrow = FALSE)
#image(mask) # A is on x-axis, B is on y-axis
```

`getGlobalAlignment` *Calculates global alignment between RT of two runs*

Description

This function selects features from oswFiles which has m-score < maxFdrLoess. It fits linear/loess regression on these feature. Retention-time mapping is established from reference to experiment run.

Usage

```
getGlobalAlignment(
  oswFiles,
  ref,
  eXp,
  fitType = "linear",
  maxFdrGlobal = 0.01,
  spanvalue = 0.1
)
```

Arguments

`oswFiles`	(list of data-frames) it is output from getFeatures function.
`ref`	(string) Must be a combination of "run" and an iteger e.g. "run2".
`eXp`	(string) Must be a combination of "run" and an iteger e.g. "run2".
`fitType`	(string) Must be from "loess" or "linear".
`maxFdrGlobal`	(numeric) A numeric value between 0 and 1. Features should have m-score lower than this value for participation in global fit.
`spanvalue`	(numeric) Spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

Value

An object of class "loess".

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-14

See Also

`getFeatures`

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
fit <- getGlobalAlignment(oswFiles = oswFiles_DIAlignR, ref = "run1", eXp = "run2",
 fitType = "linear", maxFdrGlobal = 0.05, spanvalue = 0.1)
```

`getGlobalFits` *Calculates all global alignment needed in refRun*

Description

Calculates all global alignment needed in refRun

Usage

```
getGlobalFits(
  refRun,
  features,
  fileInfo,
  globalAlignment,
  globalAlignmentFdr,
  globalAlignmentSpan,
  applyFun = lapply
)
```

Arguments

`refRun`	(data-frame) Output of getRefRun function. Must have two columsn : transition_group_id and run.
`features`	(list of data-frames) it is output from getFeatures function.
`fileInfo`	(data-frame) Output of getRunNames function.
`globalAlignment`	(string) Must be from "loess" or "linear".
`globalAlignmentFdr`	(numeric) A numeric value between 0 and 1. Features should have m-score lower than this value for participation in global fit.
`globalAlignmentSpan`	(numeric) Spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

Value

(list) Each element is either of class lm or loess.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRefRun, getFeatures, getGlobalAlignment

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath, oswMerged = TRUE)
features <- getFeatures(fileInfo, maxFdrQuery = 0.05)
precursors <- getPrecursors(fileInfo, TRUE, "DIA_Proteomics", "experiment-wide", 0.01)
precursors <- dplyr::arrange(precursors, .data$peptide_id, .data$transition_group_id)
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, TRUE, "DIA_Proteomics", "experiment-wide")
peptideScores <- lapply(peptideIDs, function(pep) dplyr::filter(peptideScores, .data$peptide_id == pep))
names(peptideScores) <- as.character(peptideIDs)
## Not run:
refRun <- getRefRun(peptideScores)
fits <- getGlobalFits(refRun, features, fileInfo, "linear", 0.05, 0.1)

## End(Not run)
```

---

getLinearfit *Calculates linear fit between RT of two runs*

Description

This function uses output of getRTdf that selects features from oswFiles which has m-score < maxFdrLoess. It fits Linear model on these feature. Loess mapping is established from reference to experiment run.

Usage

```
getLinearfit(RUNS_RT)
```

Arguments

RUNS_RT	(data-frame) must have three calumns: transition_group_id, RT.eXp, and RT.ref.

Value

An object of class "lm".

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getLOESSfit, getFeatures, getRTdf

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
## Not run:
RUNS_RT <- getRTdf(oswFiles = oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
lm.fit <- getLinearfit(RUNS_RT)

## End(Not run)
```

---

getLOESSfit *Calculates LOESS fit between RT of two runs*

Description

This function uses output of getRTdf that selects features from oswFiles which has m-score < maxFdrLoess. It fits LOESS on these feature. Loess mapping is established from reference to experiment run.

Usage

```
getLOESSfit(RUNS_RT, spanvalue = 0.1)
```

Arguments

RUNS_RT	(data-frame) must have three calumns: transition_group_id, RT.eXp, and RT.ref.
spanvalue	(numeric) Spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

Value

An object of class "loess".

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getLinearfit, getFeatures, getRTdf

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
## Not run:
RUNS_RT <- getRTdf(oswFiles = oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
Loess.fit <- getLOESSfit(RUNS_RT, spanvalue = 0.1)

## End(Not run)
```

`getMappedRT` *Get mapping of reference RT on experiment run.*

Description

This function aligns XICs of reference and experiment runs. Using alignment, it maps retention time from refernce run on experiment run.

Usage

```
getMappedRT(
  refRT,
  XICs.ref,
  XICs.eXp,
  globalFit,
  alignType,
  adaptiveRT,
  normalization,
  simMeasure,
  goFactor,
  geFactor,
  cosAngleThresh,
  OverlapAlignment,
  dotProdThresh,
  gapQuantile,
  kerLen,
  hardConstrain,
  samples4gradient,
  objType = "light"
)
```

Arguments

`refRT`	Peak's retention-time in reference run.
`XICs.ref`	List of extracted ion chromatograms from reference run.
`XICs.eXp`	List of extracted ion chromatograms from experiment run.
`globalFit`	Linear or loess fit object between reference and experiment run.
`alignType`	Available alignment methods are "global", "local" and "hybrid".

adaptiveRT	(numeric) Similarity matrix is not penalized within adaptive RT.
normalization	(character) Must be selected from "mean", "l2".
simMeasure	(string) Must be selected from dotProduct, cosineAngle, crossCorrelation, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.
goFactor	(numeric) Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	(numeric) Penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) In simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) An input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) In simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) If FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) This parameter modulates penalization of masked indices.
objType	(char) Must be selected from light, medium and heavy.

Value

(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignChromatogramsCpp

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
data(oswFiles_DIAlignR, package="DIAlignR")
run1 <- "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"
run2 <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run1]][["4618"]]
```

```
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[[run2]][["4618"]]
RUNS_RT <- getRTdf(oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
globalFit <- loess(RT.eXp ~ RT.ref, data = RUNS_RT, span = 0.1, control=loess.control(surface="direct"))
adaptiveRT <- 77.82315 #3.5*globalFit$s
## Not run:
getMappedRT(refRT = 5238.35, XICs.ref, XICs.eXp, globalFit, alignType = "hybrid",
 adaptiveRT = adaptiveRT, normalization = "mean",
  simMeasure = "dotProductMasked", goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3,
 OverlapAlignment = TRUE, dotProdThresh = 0.96, gapQuantile = 0.5, kerLen = 9L, hardConstrain = FALSE,
  samples4gradient = 100)

## End(Not run)
```

---

getMST *Create a MST from distance matrix*

Description

Builds a minimum spanning tree from the distance.

Usage

```
getMST(distMat)
```

Arguments

distMat	(dist) a pairwise distance matrix.

Value

(matrix) array of tree-edges.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

mst, traverseMST

Examples

```
m <- matrix(c(0,13,21,22, 13,0,12,13, 21,12,0,13, 22,13,13,0), byrow = TRUE,
            ncol = 4, dimnames = list(c("run1", "run2", "run3", "run4"),
                                      c("run1", "run2", "run3", "run4")))
distMat <- as.dist(m, diag = FALSE, upper = FALSE)
## Not run:
x <- as.data.frame(getMST(distMat))
weights <- reshape2::melt(as.matrix(distMat))
weights$Var <- paste(weights$Var1, weights$Var2, sep = "_")
x$weight <- weights[match(paste(x[,1], x[,2], sep = "_"), weights$Var), "value"]
g1 <- igraph::graph_from_data_frame(x, directed = FALSE)
plot(g1, edge.label = igraph::E(g1)$weight)

## End(Not run)
```

---

getMultipeptide *Get multipeptides*

Description

Each element of the multipeptide is a collection of features associated with a peptide.

Usage

```
getMultipeptide(
  precursors,
  features,
  runType = "DIA_Proteomics",
  applyFun = lapply,
  masters = NULL
)
```

Arguments

precursors	(data-frames) Contains precursors and associated transition IDs.
features	(list of data-frames) Contains features and their properties identified in each run.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
applyFun	(function) value must be either lapply or BiocParallel::bplapply.
masters	(characters) names of extra runs.

Value

(list) of dataframes having following columns:

transition_group_id	(integer) a unique id for each precursor.
run	(string) run identifier.

RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.
leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.
alignment_rank	(integer) rank of each feature post-alignment.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getPrecursors, getFeatures

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath, oswMerged = TRUE)
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, context = "experiment-wide")
features <- getFeatures(fileInfo, maxFdrQuery = 0.05)
multipeptide <- getMultipeptide(precursors, features)
multipeptide[["9861"]]
```

---

getMZMLpointers *Get pointers to each mzML file.*

Description

Returns instantiated mzRpwiz object associated to mzML file.

Usage

```
getMZMLpointers(fileInfo)
```

Arguments

fileInfo	(data-frame) Output of DIAlignR::getRunNames function

Value

(A list of mzRpwiz)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-13

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
mzPntrs <- getMZMLpointers(fileInfo)
```

---

getNativeIDs *Fetch NativeIDs*

Description

Get transition(native) IDs for the peptides.

Usage

```
getNativeIDs(oswIn, peps, params = paramsDIAlignR(), oswMerged = TRUE)
```

Arguments

oswIn	(string) path to the osw feature file.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.

Value

(integer) a vector of transition IDs.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2022) + GPL-3 Date: 2022-04-19

See Also

reduceXICs

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
oswIn <- file.path(dataPath, "osw", "merged.osw")
peps <- c(3L, 11L) # No transitions for 3L.
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
getNativeIDs(oswIn, peps, params) # 106468 106469 106470 106471 106472 106473
```

`getNodeIDs` *Get node IDs from tree*

Description

Get node IDs from tree

Usage

```
getNodeIDs(tree)
```

Arguments

`tree`	(phylo) a phylogenetic tree.

Value

(integer) a vector with names being node IDs.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getTree`

Examples

```
m <- matrix(c(0,1,2,3, 1,0,1.5,1.5, 2,1.5,0,1, 3,1.5,1,0), byrow = TRUE,
            ncol = 4, dimnames = list(c("run1", "run2", "run3", "run4"),
                                      c("run1", "run2", "run3", "run4")))
distMat <- as.dist(m)
## Not run:
tree <- getTree(distMat)
getNodeIDs(tree)

## End(Not run)
```

getNodeRun *Create a child run from two parent runs*

Description

Get merged features and merged chromatograms from parent runs. Chromatograms are written on the disk at dataPath/xics. For each precursor aligned parent time-vectors and corresponding child time-vector are also calculated and written as *_av.rda at dataPath.

Usage

```
getNodeRun(
  runA,
  runB,
  mergeName,
  dataPath,
  fileInfo,
  features,
  mzPntrs,
  prec2chromIndex,
  precursors,
  params,
  adaptiveRTs,
  refRuns,
  multipeptide,
  peptideScores,
  ropenms,
  applyFun = lapply
)
```

Arguments

runA	(string) name of a run to be merged with runB. Must be in the rownames of fileInfo.
runB	(string) name of a run to be merged with runA. Must be in the rownames of fileInfo.
mergeName	(string) name of the node that is generated with merging of runA and runB.
dataPath	(string) path to xics and osw directory.
fileInfo	(data-frame) output of getRunNames.
features	(list of data-frames) contains features and their properties identified in each run.
mzPntrs	(list) a list of mzRpwiz.
prec2chromIndex	(list) a list of dataframes having following columns: transition_group_id: it is PRECURSOR.ID from osw file. chromatogramIndex: index of chromatogram in mzML file.

precursors	(data-frame) atleast two columns transition_group_id and transition_ids are required.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
adaptiveRTs	(environment) an empty environment used to store data for downstream analysis.
refRuns	(environment) an empty environment used to store data for downstream analysis.
multipeptide	(environment) contains multiple data-frames that are collection of features associated with analytes. This is an output of getMultipeptide.
peptideScores	(list of data-frames) each dataframe has scores of a peptide across all runs.
ropenms	(pyopenms module) get this python module through get_ropenms. Required only for chrom.mzML files.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXICs, getChildXICs, traverseUp

Examples

```
library(data.table)
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
fileInfo <- getRunNames(dataPath = dataPath)
mzPntrs <- list2env(getMZMLpointers(fileInfo))
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, runType = params[["runType"]],
 context = "experiment-wide", maxPeptideFdr = params[["maxPeptideFdr"]])
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, oswMerged = TRUE, params[["runType"]], params[["context"]
masters <- paste("master", 1:(nrow(fileInfo)-1), sep = "")
peptideScores <- lapply(peptideIDs, function(pep) {x <- peptideScores[.(pep)][,-c(1L)]
 x <- rbindlist(list(x, data.table("run" = masters, "score" = NA_real_, "pvalue" = NA_real_,
    "qvalue" = NA_real_)), use.names=TRUE)
  setkeyv(x, "run"); x})
names(peptideScores) <- as.character(peptideIDs)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
## Not run:
masterFeatures <- dummyFeatures(precursors, nrow(fileInfo)-1, 1L)
features <- do.call(c, list(features, masterFeatures))
multipeptide <- getMultipeptide(precursors, features, numMerge = 0L, startIdx = 1L)
```

```
prec2chromIndex <- getChromatogramIndices(fileInfo, precursors, mzPntrs)
masterChromIndex <- dummyChromIndex(precursors, nrow(fileInfo)-1, 1L)
prec2chromIndex <- do.call(c, list(prec2chromIndex, masterChromIndex))
mergeName <- "master1"
adaptiveRTs <- new.env()
refRuns <- new.env()
getNodeRun(runA = "run2", runB = "run0", mergeName = mergeName, dataPath = ".", fileInfo, features,
 mzPntrs, prec2chromIndex, precursors, params, adaptiveRTs, refRuns, multipeptide, peptideScores, ropenms = NULL)
file.remove(file.path(".", "xics", paste0(mergeName, ".chrom.sqMass")))
file.remove(list.files(".", pattern = "*_av.rds", full.names = TRUE))

## End(Not run)
for(run in names(mzPntrs)) DBI::dbDisconnect(mzPntrs[[run]])
```

`getOswAnalytes` *Fetch analytes from OSW file*

Description

Get a data-frame of analytes, their chromatogram indices and associated FDR-scores.

Usage

```
getOswAnalytes(
  fileInfo,
  oswMerged = TRUE,
  analyteInGroupLabel = FALSE,
  maxFdrQuery = 0.05,
  runType = "DIA_Proteomics"
)
```

Arguments

`oswMerged`	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
`analyteInGroupLabel`	(logical) TRUE for getting analytes as PRECURSOR.GROUP_LABEL from osw file. FALSE for fetching analytes as PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.
`maxFdrQuery`	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
`runType`	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
`dataPath`	(char) path to xics and osw directory.
`filenames`	(data-frame) column "filename" contains RUN table from osw files. column "runs" contain respective mzML names without extension. To get filenames use `getRunNames` function.

Value

(A list of data-frames) Each data-frame has following columns:

`transition_group_id`
: (string) it is either fetched from PRECURSOR.GROUP_LABEL or a combination of PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.

`filename`
: (string) as mentioned in RUN table of osw files.

`peak_group_rank`
: (integer) rank of each feature associated with transition_group_id.

`m_score`
: (numeric) q-value of each feature associated with transition_group_id.

`transition_id`
: (integer) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getRunNames`

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
filenames <- getRunNames(dataPath = dataPath)
## Not run:
oswFiles <- getOswAnalytes(dataPath = dataPath, filenames = filenames,
 analyteInGroupLabel = TRUE)
oswFiles[["run0"]][1,]
oswFiles <- getOswAnalytes(dataPath = dataPath, filenames = filenames,
 analyteInGroupLabel = FALSE)
oswFiles[["run0"]][1,]

## End(Not run)
```

---

getOswFiles *Get list of peptides and their chromatogram indices.*

Description

This function reads all osw and xics files in the directories at dataPath. It selects analytes which has associated features with m-score < maxFdrQuery. For these analytes it fetches chromatogram indices by matching transition_id(osw) with chromatogramID(xics).

Usage

```
getOswFiles(
  fileInfo,
  mzPntrs,
  maxFdrQuery = 0.05,
  analyteFDR = 0.01,
  oswMerged = TRUE,
  analytes = NULL,
  runType = "DIA_Proteomics",
  analyteInGroupLabel = FALSE
)
```

Arguments

mzPntrs	A list of mzRpwiz. FALSE for fetching analytes as PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.
maxFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
analyteFDR	(numeric) Not used.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
analytes	(string) analyte is as PRECURSOR.GROUP_LABEL or as PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
analyteInGroupLabel	(logical) TRUE for getting analytes as PRECURSOR.GROUP_LABEL from osw file.
dataPath	(char) path to xics and osw directory.
filenames	(data-frame) column "filename" contains RUN table from osw files. column "runs" contain respective mzML names without extension. To get filenames use DIAlignR::getRunNames function.

Value

(data-frames) Data-frame has following columns:

transition_group_id	(string) it is either fetched from PRECURSOR.GROUP_LABEL or a combination of PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.
RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
delta_rt	(numeric) as in FEATURE.DELTA_RT of osw files.
assay_RT	(numeric) library retention time as in PRECURSOR.LIBRARY_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.

leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.
chromatogramIndex	(integer) Index of chromatogram in mzML file.
transition_ids	(integer) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
filenames <- getRunNames(dataPath = dataPath)
## Not run:
mzPntrs <- getMZMLpointers(filenames)
oswFiles <- getOswFiles(filenames, mzPntrs, analyteInGroupLabel = TRUE)
rm(mzPntrs)

## End(Not run)
```

---

getPeptideQuery *Get peptide scores*

Description

For each peptide, its score, pvalue and qvalues are fetched across all runs.

Usage

```
getPeptideQuery(runType = "DIA_Proteomics")
```

Arguments

runType	(char) This must be one of the strings "DIA_proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-07-01

See Also

getPeptideScores

---

getPeptideQuery2 *Get peptide scores*

Description

For each peptide, its score, pvalue and qvalues are fetched across all runs.

Usage

```
getPeptideQuery2(runType = "DIA_Proteomics")
```

Arguments

runType	(char) This must be one of the strings "DIA_proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-11-18

See Also

getPeptideScores

getPeptideScores *Get scores of peptide*

Description

Get a list of dataframes that contains peptide scores, pvalues, and qvalues across all runs.

Usage

```
getPeptideScores(
  fileInfo,
  peptides,
  oswMerged = TRUE,
  runType = "DIA_Proteomics",
  context = "global"
)
```

Arguments

fileInfo	(data-frame) Output of getRunNames function.
peptides	(integer) Ids of peptides for which scores are required.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
context	(string) Context used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".

Value

(list of dataframes) dataframe has following columns:

peptide_id	(integer) a unique id for each precursor.
run	(character) as in SCORE_PEPTIDE.RUN_ID of osw files.
score	(numeric) as in SCORE_PEPTIDE.SCORE of osw files.
pvalue	(numeric) as in SCORE_PEPTIDE.PVALUE of osw files.
qvalue	(numeric) as in SCORE_PEPTIDE.QVALUE of osw files.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, fetchPeptidesInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
precursorsInfo <- getPrecursors(fileInfo, oswMerged = TRUE, runType = "DIA_Proteomics",
context = "experiment-wide", maxPeptideFdr = 0.05)
peptidesInfo <- getPeptideScores(fileInfo, unique(precursorsInfo$peptide_id))
dim(peptidesInfo) # 684 5
```

---

getPrecursorByID *Find precursors given their IDs*

Description

Get a data-frame of analytes' transition_group_id, transition_ids, peptide_id and amino-acid sequences.

Usage

```
getPrecursorByID(
  analytes,
  fileInfo,
  oswMerged = TRUE,
  runType = "DIA_Proteomics"
)
```

Arguments

analytes	(integer) a vector of integers.
fileInfo	(data-frame) Output of getRunNames function.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

(data-frames) A data-frame having following columns:

transition_group_id	(integer) a unique id for each precursor.
transition_id	(list) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.
peptide_id	(integer) a unique id for each peptide. A peptide can have multiple precursors.
sequence	(string) amino-acid sequence of the precursor with possible modifications.
charge	(integer) charge on the precursor.
group_label	(string) TODO Figure it out.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2019-04-06

See Also

getRunNames, fetchPrecursorsInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath, oswMerged = TRUE)
precursors <- getPrecursorByID(c(32L, 2474L), fileInfo, oswMerged = TRUE)
```

---

getPrecursorIndices *Get MS1 chromatogram indices of precursors.*

Description

This function reads the header of chromatogram files. It then fetches chromatogram indices by matching transition_group_id(osw) with chromatogramID(xics).

Usage

```
getPrecursorIndices(
  fileInfo,
  precursors,
  mzPntrs,
  ions = ".*",
  applyFun = lapply
)
```

Arguments

fileInfo	(data-frame) Output of getRunNames function.
precursors	(data-frame) Atleast two columns transition_group_id and transition_ids are required.
mzPntrs	A list of mzRpwiz.
ions	(character) Strings to specify ions. eg. i0, i1.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list) A list of dataframes having following columns:

`transition_group_id`
: (string) it is PRECURSOR.ID from osw file.

`chromatogramIndex`
: (integer) index of MS1 chromatogram in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`chromatogramIdAsInteger`, `mapPrecursorToChromIndices`

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, context = "experiment-wide")
mzPntrs <- getMZMLpointers(fileInfo)
prec2chromIndex <- getPrecursorIndices(fileInfo, precursors, mzPntrs)
for(mz in mzPntrs) DBI::dbDisconnect(mz)
```

---

getPrecursors *Get precursors from all feature files*

Description

Get a data-frame of analytes' transition_group_id, transition_ids, peptide_id and amino-acid sequences.

Usage

```
getPrecursors(
  fileInfo,
  oswMerged = TRUE,
  runType = "DIA_Proteomics",
  context = "global",
  maxPeptideFdr = 0.05,
  level = "Peptide",
  useIdentifying = FALSE
)
```

Arguments

fileInfo	(data-frame) Output of getRunNames function.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
context	(string) Context used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".
maxPeptideFdr	(numeric) A numeric value between 0 and 1. It is used to filter peptides from osw file which have SCORE_PEPTIDE.QVALUE less than itself.
level	(string) Apply maxPeptideFDR on Protein as well if specified as "Protein". Default: "Peptide".
useIdentifying	(logical) Set TRUE to use identifying transitions in alignment. (DEFAULT: FALSE)

Value

(data-frames) A data-frame having following columns:

transition_group_id	(integer) a unique id for each precursor.
peptide_id	(integer) a unique id for each peptide. A peptide can have multiple precursors.
sequence	(string) amino-acid sequence of the precursor with possible modifications.
charge	(integer) charge on the precursor.
group_label	(string) TODO Figure it out.
transition_ids	(list) fragment-ion ID associated with transition_group_id. This is matched with chromatogram ID in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, fetchPrecursorsInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
precursorsInfo <- getPrecursors(fileInfo, oswMerged = TRUE, runType = "DIA_Proteomics",
context = "experiment-wide", maxPeptideFdr = 0.05)
dim(precursorsInfo) # 234  6
```

getPrecursorsQuery *Get precursor Info*

Description

For each precursor in the table respective transition ids are fetched. Order of transition is kept same as the order of their intensities in `getTransitionsQuery`.

Usage

```
getPrecursorsQuery(runType = "DIA_Proteomics", level = "Peptide")
```

Arguments

`runType`	(string) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
`level`	(string) Apply maxPeptideFDR on Protein as well if specified as "Protein". Default: "Peptide".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`fetchPrecursorsInfo`

getPrecursorsQueryID *Get precursor Info*

Description

For each precursor in the table respective transition ids are fetched. Order of transition is kept same as the order of their intensities in `getTransitionsQuery`.

Usage

```
getPrecursorsQueryID(analytes, runType = "DIA_Proteomics")
```

Arguments

analytes	(integer) A vector of integer that is searched in PRECURSOR.ID.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-04-04

See Also

fetchPrecursorsInfo

---

getPrecursorSubset *Prints messages if a certain number of analytes are aligned*

Description

Prints messages if a certain number of analytes are aligned

Usage

```
getPrecursorSubset(precursors, params)
```

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-11-19

getQuery *Generate SQL query to fetch information from osw files.*

Description

Generate SQL query to fetch information from osw files.

Usage

```
getQuery(
  maxFdrQuery,
  oswMerged = TRUE,
  analytes = NULL,
  filename = NULL,
  runType = "DIA_Proteomics",
  analyteInGroupLabel = FALSE,
  identifying = FALSE
)
```

Arguments

maxFdrQuery	(numeric) value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
analytes	(vector of strings) transition_group_ids for which features are to be extracted. analyteInGroupLabel must be set according the pattern used here.
filename	(string) as mentioned in RUN table of osw files.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
analyteInGroupLabel	(logical) TRUE for getting analytes as PRECURSOR.GROUP_LABEL from osw file.
identifying	(logical) TRUE for the extraction of identifying transtions. (Default: FALSE)

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

getRefExpFeatureMap *Alignment Feature Mapping Table*

Description

Construct an alignment feature mapping table for star align method, to map aligned features in experiment runs to the reference run

Usage

```
getRefExpFeatureMap(precursors, features, applyFun = lapply)
```

Arguments

precursors	(data-frames) Contains precursors and associated transition IDs.
features	(list of data-frames) Contains features and their properties identified in each run.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list) of dataframes per precursor peptide id having following columns:

reference_feature_id
: (integer) id of reference feature.

experiment_feature_id
: (integer) id of experiment feature aligned to reference feature.

Author(s)

Justin Sing, <justinc.sing@mail.utoronto.ca>

ORCID: 0000-0003-0386-0092

See Also

getPrecursors, getFeatures

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath, oswMerged = TRUE)
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, context = "experiment-wide")
features <- getFeatures(fileInfo, maxFdrQuery = 0.05)
multiFeatureAlignmentMap <- getRefExpFeatureMap(precursors, features)
multiFeatureAlignmentMap[["9861"]]
```

getRefRun *Fetch the reference run for each peptide*

Description

Provides the reference run based on lowest p-value.

Usage

```
getRefRun(peptideScores, applyFun = lapply)
```

Arguments

peptideScores	(list of data-frames) each dataframe has scores of a peptide across all runs.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(dataframe) has two columns:

peptide_id	(integer) a unique id for each peptide.
run	(string) run identifier.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getPeptideScores

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
precursorsInfo <- getPrecursors(fileInfo, oswMerged = TRUE, runType = "DIA_Proteomics",
                                context = "experiment-wide", maxPeptideFdr = 0.05)
peptideIDs <- unique(precursorsInfo$peptide_id)
peptidesInfo <- getPeptideScores(fileInfo, peptideIDs)
peptidesInfo <- lapply(peptideIDs, function(pep) peptidesInfo [.(pep),])
names(peptidesInfo) <- as.character(peptideIDs)
getRefRun(peptidesInfo)
```

getRSE *Calculates Residual Standard Error of the fit*

Description

Calculates Residual Standard Error of the fit

Usage

```
getRSE(fit, globalAlignment)
```

Arguments

fit	(lm or loess) Linear or loess fit object between reference and experiment run.

Value

(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getLOESSfit, getLinearfit, getGlobalAlignment

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
## Not run:
Loess.fit <- getGlobalAlignment(oswFiles = oswFiles_DIAlignR, ref = "run1", eXp = "run2",
maxFdrGlobal = 0.05, spanvalue = 0.1, fit = "loess")
getRSE(Loess.fit, "loess")

## End(Not run)
```

getRTdf *Calculates global alignment between RT of two runs*

Description

This function selects features from oswFiles which has m-score < maxFdrLoess. It fits linear/loess regression on these feature. Retention-time mapping is established from reference to experiment run.

Usage

```
getRTdf(features, ref, eXp, maxFdrGlobal)
```

Arguments

features	(list of data-frames) it is output from getFeatures function.
ref	(string) Must be a combination of "run" and an iteger e.g. "run2".
eXp	(string) Must be a combination of "run" and an iteger e.g. "run2".
maxFdrGlobal	(numeric) A numeric value between 0 and 1. Features should have m-score lower than this value for participation in global fit.

Value

A data-frame

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getGlobalAlignment

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
df <- getRTdf(features = oswFiles_DIAlignR, ref = "run1", eXp = "run2", maxFdrGlobal = 0.05)
```

getRunNames *Get names of all runs*

Description

Fetches all osw files, then, keeps only those runs which has corresponding mzML files. mzML file names must match with RUN.FILENAME columns of osw files.

Usage

```
getRunNames(dataPath, oswMerged = TRUE, params = paramsDIAlignR())
```

Arguments

dataPath	(char) Path to xics and osw directory.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.

Value

(dataframe) it has five columns:

spectraFile	(string) as mentioned in RUN table of osw files.
runName	(string) contain respective mzML names without extension.
spectraFileID	(string) ID in RUN table of osw files.
featureFile	(string) Path to the feature file.
chromatogramFile	(string) Path to the chromatogram file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
getRunNames(dataPath = dataPath, oswMerged = TRUE)
```

`getSeqSimMatCpp` *Calculates similarity matrix for two sequences*

Description

Calculates similarity matrix for two sequences

Usage

```
getSeqSimMatCpp(seq1, seq2, match, misMatch)
```

Arguments

`seq1`	(char) A single string.
`seq2`	(char) A single string.
`match`	(double) Score for character match.
`misMatch`	(double) score for character mismatch.

Value

s (matrix) Numeric similarity matrix. Rows and columns expresses seq1 and seq2, respectively.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 Date: 2019-03-05

Examples

```
# Get sequence similarity of two DNA strings
Match=10; MisMatch=-2
seq1 = "GCAT"; seq2 = "CAGTG"
getSeqSimMatCpp(seq1, seq2, Match, MisMatch)
matrix(c(-2, 10, -2, -2, -2, -2, 10, -2, 10, -2, -2, -2, -2, -2, -2, 10, 10, -2, -2, -2),
 4, 5, byrow = FALSE)
```

`getTransitions` *Get transitions from all feature files*

Description

Get a list of data-frame of OpenSwath features that contains retention time, intensities, boundaries etc.

Usage

```
getTransitions(
  fileInfo,
  maxFdrQuery = 0.05,
  runType = "DIA_Proteomics",
  applyFun = lapply
)
```

Arguments

`fileInfo`	(data-frame) output of `getRunNames` function.
`maxFdrQuery`	(numeric) a numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
`runType`	(char) yhis must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
`applyFun`	(function) value must be either lapply or BiocParallel::bplapply.

Value

(list of dataframes) each dataframe has following columns:

`transition_group_id`	(integer) a unique id for each precursor.
`RT`	(numeric) retention time as in FEATURE.EXP_RT of osw files.
`intensity`	(list) of peak intensities as in FEATURE_TRANSITION.AREA_INTENSITY of osw files.
`leftWidth`	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
`rightWidth`	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
`peak_group_rank`	(integer) rank of each feature associated with transition_group_id.
`m_score`	(numeric) q-value of each feature associated with transition_group_id.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getRunNames, fetchTransitionsFromRun

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath)
transitions <- getTransitions(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
dim(transitions[[2]]) # 938  8
```

---

getTransitionsQuery *Get transitions from a SQLite file*

Description

Query is generated to identify features and their transitions below a FDR cut-off from a run. Order of transition intensity is kept same as the order of their Ids in getPrecursorsQuery.

Usage

```
getTransitionsQuery(runType = "DIA_Proteomics")
```

Arguments

runType	(char) This must be one of the strings "DIA_proteomics", "DIA_IPF", "DIA_Metabolomics".

Value

SQL query to be searched.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-11-15

See Also

fetchTransitionsFromRun, getPrecursorsQueryID, getPrecursorsQuery

getTree *Create a phylogenetic tree*

Description

Builds a phylogenetic tree from the distance matrix using UPGMA algorithm.

Usage

```
getTree(distMat, method = "average", prefix = "master")
```

Arguments

distMat	(dist) a pairwise distance matrix.

Value

(phylo) a tree.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

upgma, getNodeIDs

Examples

```
m <- matrix(c(0,1,2,3, 1,0,1.5,1.5, 2,1.5,0,1, 3,1.5,1,0), byrow = TRUE,
            ncol = 4, dimnames = list(c("run1", "run2", "run3", "run4"),
                                      c("run1", "run2", "run3", "run4")))
distMat <- as.dist(m, diag = FALSE, upper = FALSE)
## Not run:
tree <- getTree(distMat)

## End(Not run)
tree <- ape::read.tree(text = "(run1:9,(run2:7,run0:2)master2:5)master1;")
plot(tree, type = "phylogram", show.node.label = TRUE)
ape::axisPhylo(1)
plot(tree, type = "unrooted", show.node.label = TRUE)
ape::edgelabels(tree$edge.length)
tree <- ape::nj(distMat) # Neighbor-Joining tree
plot(tree, type = "unrooted", show.node.label = TRUE)
ape::edgelabels(tree$edge.length)
```

getXICs *Get XICs of all analytes*

Description

For all the analytes requested in runs, it first creates oswFiles, then, fetches chromatogram indices from oswFiles and extract chromatograms from mzML files.

Usage

```
getXICs(
  analytes,
  runs,
  dataPath = ".",
  maxFdrQuery = 1,
  runType = "DIA_Proteomics",
  oswMerged = TRUE,
  params = paramsDIAlignR()
)
```

Arguments

analytes	(integer) a vector of precursor IDs.
runs	(vector of string) names of mzML files without extension.
dataPath	(string) Path to xics and osw directory.
maxFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.

Value

A list of list. Each list contains XIC-group for that run. XIC-group is a list of dataframe that has elution time and intensity of fragment-ion XIC.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getXICs4AlignObj, getRunNames, analytesFromFeatures

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
analytes <- c(32L, 898L, 2474L)
XICs <- getXICs(analytes, runs = runs, dataPath = dataPath)
```

`getXICs4AlignObj` *Extract XICs of analytes*

Description

For all the analytes requested, it fetches chromatogram indices from prec2chromIndex and extracts chromatograms using mzPntrs.

Usage

```
getXICs4AlignObj(mzPntrs, fileInfo, runs, prec2chromIndex, analytes)
```

Arguments

`mzPntrs`	a list of mzRpwiz.
`fileInfo`	(data-frame) output of getRunNames().
`runs`	(vector of string) names of mzML files without extension.
`prec2chromIndex`	(list of data-frames) output of getChromatogramIndices(). Each dataframe has two columns: transition_group_id and chromatogramIndex.
`analytes`	(integer) a vector of precursor IDs.

Value

A list of list of data-frames. Each data frame has elution time and intensity of fragment-ion XIC.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getChromatogramIndices`, `getRunNames`

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
          "hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt")
analytes <- c(32L, 898L, 2474L)
params <- paramsDIAlignR()
params[["chromFile"]] <- "mzML"
fileInfo <- getRunNames(dataPath = dataPath)
fileInfo <- updateFileInfo(fileInfo, runs)
precursors <- getPrecursorByID(analytes,fileInfo)
mzPntrs <- getMZMLpointers(fileInfo)
prec2chromIndex <- getChromatogramIndices(fileInfo, precursors, mzPntrs)
XICs <- getXICs4AlignObj(mzPntrs, fileInfo, runs, prec2chromIndex, analytes)
rm(mzPntrs)
```

`get_ropenms` *Get ropenms handle*

Description

Python path can also be set using Sys.setenv(RETICULATE_PYTHON = pythonPath). Also, remove .Rhistory file to avoid conflict with previously used python version.

Usage

```
get_ropenms(pythonPath = NULL, condaEnv = NULL, useConda = TRUE)
```

Arguments

`pythonPath`	(string) path of the python program that has pyopenms module.
`condaEnv`	(string) name of the conda environment that has pyopenms module.
`useConda`	(logical) TRUE: Use conda environment. FALSE: Use python through pythonPath.

Value

(pyopenms module)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
## Not run:
  ropenms <- get_ropenms(condaEnv = "envName", useConda=TRUE)

## End(Not run)
```

`imputeChromatogram` *Fill missing intensities in a chromatogram*

Description

Fill missing intensities in a chromatogram

Usage

```
imputeChromatogram(
  chromatogram,
  method = "spline",
  polyOrd = 4,
  kernelLen = 9,
  splineMethod = "fmm"
)
```

Arguments

`chromatogram`	(data-frames) first column is time, second column is intensity.
`method`	(string) must be either "spline", "sgolay" or "linear".
`polyOrd`	(integer) must be less than kernelLen.
`kernelLen`	(integer) must be an odd integer.
`splineMethod`	(string) must be either "fmm" or "natural".

Value

(dataframe) has two columns:

`time`	(numeric)
`intensity`	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

na.approx, na.spline

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
 4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom$y[c(1,8, 14)] <- NA
imputeChromatogram(chrom, "sgolay")
imputeChromatogram(chrom, "spline")
imputeChromatogram(chrom, "linear")
```

---

ipfReassignFDR *Re-Assign FDR Aligned Peaks*

Description

This method will re-assign peaks with either the MS2 FDR if there was not enough confidence (identifying transitions) in the IPF FDR, or it will assign the user defined alignedFDR2. This assumes the reference run has a really good peak that meets the IPF FDR threshold.

Usage

```
ipfReassignFDR(dt, refRuns, fileInfo, params)
```

Arguments

dt	(data.table) Aligned results tables from writeTables
refRuns	(dataframe) A peptide reference run table from getRefRun
fileInfo	(dataframe) A table with run filename information from getRunNames
params	(list) A list of parameters used by DIAlignR generated from paramsDIAlignR

Value

(data.table) Aligned results table with reassigned m-scores

Author(s)

Justin Sing, <justinc.sing@mail.utoronto.ca>

ORCID: 0000-0003-0386-0092

See Also

writeTables, getRefRun, getRunNames, paramsDIAlignR

Examples

```
## Not run:
finalTbl <- ipfReassignFDR(finalTbl, refRuns, fileInfo, params)

## End(Not run)
```

`mapIdxToTime` *Establishes mapping from index to time*

Description

Takes a time vector and index vector of same length. This function create a new time vector given indices specified in `idx`.

Usage

```
mapIdxToTime(timeVec, idx)
```

Arguments

`timeVec`	A numeric vector
`idx`	An integer vector

Value

A mutated time vector

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
timeVec <- c(1.3,5.6,7.8)
idx <- c(NA, NA, 1L, 2L, NA, NA, 3L, NA)
mapIdxToTime(timeVec, idx) # c(NA, NA, 1.3, 5.6, 6.333, 7.067, 7.8, NA)
```

mappedRTfromAlignObj *Map reference run time on the experiment run.*

Description

Retention time from reference run is mapped to experiment run using AlignObj.

Usage

```
mappedRTfromAlignObj(refRT, tVec.ref, tVec.eXp, AlignObj)
```

Arguments

refRT	(numeric) retention time in reference run.
tVec.ref	(numeric) time vector of refernce run.
tVec.eXp	(numeric) time vector of experiment run.
AlignObj	(S4 object) must have indexA_aligned, indexB_aligned and score as slots.

Value

(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
tVec.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]][[1]
tVec.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]][[1
## Not run:
AlignObj <- testAlignObj()
mappedRTfromAlignObj(refRT= 5238.35, tVec.ref, tVec.eXp, AlignObj)

## End(Not run)
```

`mapPrecursorToChromIndices` *Merge precursor and transitions mapping with chromatogram header*

Description

Merges dataframes on transition_ids(OSW) = chromatogramId(mzML).

Usage

```
mapPrecursorToChromIndices(prec2transition, chromHead)
```

Arguments

`prec2transition`	(dataframe) This has two columns: transition_group_id and transition_ids with integer values.
`chromHead`	(dataframe) This has two columns: chromatogramId and chromatogramIndex with integer values.

Value

(dataframe) having following columns:

`transition_group_id`
: (string) it is either fetched from PRECURSOR.GROUP_LABEL or a combination of PEPTIDE.MODIFIED_SEQUENCE and PRECURSOR.CHARGE from osw file.

`chromatogramIndex`
: (integer) Index of chromatogram in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getChromatogramIndices`

masterXICs_DIAlignR *Master fragment-ion chromatograms from two parents*

Description

Created merged XICs of peptide (ID = 4618) 14299_QFNNTDIVLLEDFQK/3 from two SWATH runs:
run1 : hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML
run2 : hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML

Usage

```
masterXICs_DIAlignR
```

Format

The format is similar to the output of childXICs. A list of two elements: First element contains six fragmentation chromatograms. The second element has aligned parent time-vectors and corresponding child time-vector. It has five columns:

indexAligned.ref (integer) aligned indices of reference run.

indexAligned.eXp (integer) aligned indices of experiment run.

tAligned.ref (numeric) aligned time-vector of reference run.

tAligned.eXp (numeric) aligned time-vector of experiment run.

alignedChildTime (numeric) aligned time-vector of master run.

Source

File test_GenerateData.R has source code to generate the example data.

mergeOswAnalytes_ChromHeader *Merge dataframes from OSW and mzML files*

Description

Merges dataframes on transition_id(OSW) = chromatogramId(mzML).

Usage

```
mergeOswAnalytes_ChromHeader(
  oswAnalytes,
  chromHead,
  analyteFDR = 1,
  runType = "DIA_Proteomics"
)
```

Arguments

oswAnalytes	(dataframe) This is an output of getOswFiles.
chromHead	(dataframe) This has two columns: chromatogramId and chromatogramIndex with integer values.
analyteFDR	(numeric) Not used.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getOswFiles

---

mergeXIC *Merge two XICs into one*

Description

Both extracted-ion chromatograms must have same number of rows. Missing values are not allowed. Intensities are weighted-averaged.

Usage

```
mergeXIC(XIC.ref, XIC.eXp, wRef, mergeStrategy)
```

Arguments

XIC.ref	(data-frame) extracted ion chromatogram from reference run. Must not contain missing values.
XIC.eXp	(data-frame) extracted ion chromatogram from experiment run. Must not contain missing values.
wRef	(numeric) Weight of the reference XIC. Must be between 0 and 1.
mergeStrategy	(string) must be either ref, avg, refStart or refEnd.

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

time-merging

There are three startegies for calculating time to keep sampling rate equal to parents:
- Use reference time.
- Average time (Non-gap region will have parent's sampling rate).
- Start/end with the average then fixed sampling rate.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

childXIC, alignedXIC

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["14299_QFNN
## Not run:
plot(mergeXIC(XICs.ref[[1]], XICs.eXp[[1]], wRef = 0.5, mergeStrategy = "ref"), type = "l")

## End(Not run)
```

---

mstAlignRuns *Peptide quantification through MST alignment*

Description

This function expects osw and xics directories at dataPath. It first reads osw files and fetches chromatogram indices for each analyte. To perform alignment, first a guide-tree is built using mst which can also be provided with mstNet parameter. As we traverse from the start node to the other nodes, runs are aligned pairwise.

Usage

```
mstAlignRuns(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  scoreFile = NULL,
  runs = NULL,
  peps = NULL,
  mstNet = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
scoreFile	(string) path to the peptide score file, needed when oswMerged is FALSE.
runs	(string) names of xics file without extension.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
mstNet	(matrix) array of tree-edges. Look up getMST.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2021) + GPL-3 Date: 2021-05-15

See Also

alignTargetedRuns, progAlignRuns, getMST

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
mstAlignRuns(dataPath, params = params, outFile = "DIAlignR")
## Not run:
```

```
y <- strsplit("run0 run1\nrun2 run2", split = '\n')[[1]]
y <- cbind(A = strsplit(y[1], " ")[[1]], B = strsplit(y[2], " ")[[1]])
plot(igraph::graph_from_edgelist(y, directed = FALSE))

## End(Not run)
```

`MSTperBatch` *Aligns an analyte for a batch of peptides*

Description

For the ith analyte in the batch, this function traverse the MST from start node and align runs pairwise. In the process it updates multipeptide with aligned features.

Usage

```
MSTperBatch(
  iBatch,
  nets,
  peptides,
  multipeptide,
  refRuns,
  precursors,
  prec2chromIndex,
  fileInfo,
  mzPntrs,
  params,
  globalFits,
  RSE,
  applyFun = lapply
)
```

Arguments

`nets`	(list) set of trees ordered for traversal obained from `traverseMST`.
`peptides`	(integer) vector of peptide IDs.
`multipeptide`	(list) contains multiple data-frames that are collection of features associated with analytes. This is an output of `getMultipeptide`.
`refRuns`	(data-frame) output of `getRefRun`. Must have two columsn : transition_group_id and run.
`precursors`	(data-frame) atleast two columns transition_group_id and transition_ids are required.
`prec2chromIndex`	(list) a list of dataframes having following columns: transition_group_id: it is PRECURSOR.ID from osw file. chromatogramIndex: index of chromatogram in mzML file.

fileInfo	(data-frame) output of getRunNames.
mzPntrs	(list) a list of mzRpwiz.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
globalFits	(list) each element is either of class lm or loess. This is an output of getGlobalFits.
RSE	(list) Each element represents Residual Standard Error of corresponding fit in globalFits.

Value

invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

mstAlignRuns, alignToRefMST, getAlignedTimesFast, getMultipeptide

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
```

---

mstScript1 *Extract features and generate minimum spanning tree.*

Description

Extract features and generate minimum spanning tree.

Usage

```
mstScript1(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  runs = NULL,
  mstNet = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
runs	(string) names of xics file without extension.
mstNet	(string) minimum spanning tree in string format. See example of mstScript2.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2022) + GPL-3 Date: 2022-04-19

See Also

mstAlignRuns, mstScript2

Examples

```
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
dataPath <- system.file("extdata", package = "DIAlignR")
BiocParallel::register(BiocParallel::MulticoreParam(workers = 4, progressbar = TRUE))
mstScript1(dataPath, outFile = "testDIAlignR", params = params, applyFun = BiocParallel::bplapply)
file.remove(file.path(dataPath, "testDIAlignR_mst1.RData"))
```

---

mstScript2 *Performs alignment using mstScript1 output*

Description

Performs alignment using mstScript1 output

Usage

```
mstScript2(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  scoreFile = NULL,
  peps = NULL,
  mstNet = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
scoreFile	(string) path to the peptide score file, needed when oswMerged is FALSE.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
mstNet	(string) minimum spanning tree in string format. See example of mstScript2.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

mstAlignRuns, mstScript1

Examples

```
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
dataPath <- system.file("extdata", package = "DIAlignR")
BiocParallel::register(BiocParallel::MulticoreParam(workers = 4, progressbar = TRUE))
mstScript1(dataPath, outFile = "testDIAlignR", params = params, applyFun = BiocParallel::bplapply)
mstNet <- "run0 run0\nrun1 run2"
mstScript2(dataPath, outFile = "testDIAlignR", params = params, mstNet = mstNet, applyFun = lapply)
file.remove(file.path(dataPath, "testDIAlignR_mst1.RData"))
file.remove("testDIAlignR.tsv")
```

---

multipeptide_DIAlignR *Analytes information from multipeptide.*

Description

analytes info from three SWATH runs:

run0 : hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt.chrom.mzML
run1 : hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML
run2 : hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML

Usage

```
multipeptide_DIAlignR
```

Format

A list of 199 elements where each element represents a precursor and consists of a dataframe:

transition_group_id ID of each precursor. Same as the name of the list

RT Retention time, in sec

intensity Inensity of associated feature

leftWidth Left width of the peak, in sec

rightWidth Right width of the peak, in sec

peak_group_rank Ranking of associated feature

m_score qvalue of associated feature

run Name of the run, feature is from

alignment_rank Rank of the feature after alignment

Source

Raw files are downloaded from Peptide Atlas. File test_GenerateData.R has source code to generate the example data.

`nrDesc` *Number of descendants*

Description

Get the number of descendants for each node in the tree.

Usage

```
nrDesc(tree)
```

Arguments

`tree`	(phylo) a phylogenetic tree.

Value

(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

See Also

`Ntip`, `getNodeIDs`

Examples

```
m <- matrix(c(0,1,2,3, 1,0,1.5,1.5, 2,1.5,0,1, 3,1.5,1,0), byrow = TRUE,
            ncol = 4, dimnames = list(c("run1", "run2", "run3", "run4"),
                                      c("run1", "run2", "run3", "run4")))
distMat <- as.dist(m, diag = FALSE, upper = FALSE)
## Not run:
tree <- getTree(distMat)
getNodeIDs(tree)
nrDesc(tree)

## End(Not run)
```

---

oswFiles_DIAlignR *Analytes information from osw files*

Description

analytes info from three SWATH runs:

run0 : hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt.chrom.mzML
run1 : hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML
run2 : hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML

Usage

```
oswFiles_DIAlignR
```

Format

A list of three elements where each element consists of a dataframe:

transition_group_id ID of each peptide

RT Retention time, in sec

intensity Inensity of associated feature

leftWidth Left width of the peak, in sec

rightWidth Right width of the peak, in sec

peak_group_rank Ranking of associated feature

m_score qvalue of associated feature

Source

Raw files are downloaded from Peptide Atlas. File test_GenerateData.R has source code to generate the example data.

otherChildXICpp *Get child chromatogram for other precursors using main precursor alignment*

Description

Get child chromatogram for other precursors using main precursor alignment

Usage

```
otherChildXICpp(
  l1,
  l2,
  kernelLen,
  polyOrd,
  mat,
  childTime,
  wRef = 0.5,
  splineMethod = "natural"
)
```

Arguments

l1	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
l2	(list) A list of numeric matrix of two columns. l1 and l2 should have same length.
kernelLen	(integer) length of filter. Must be an odd number.
polyOrd	(integer) TRUE: remove background from peak signal using estimated noise levels.
mat	(matrix) aligned time and child time from the main precursor.
childTime	(numeric) iime vector from the child chromatogram.
wRef	(numeric) Weight of the reference XIC. Must be between 0 and 1.
splineMethod	(string) must be either "fmm" or "natural".

Value

(List) of chromatograms.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2021) + MIT Date: 2021-01-08

Examples

```
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR
XICs.ref <- lapply(XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
XICs.eXp <- lapply(XICs[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]], as.matrix)
Bp <- seq(4964.752, 5565.462, length.out = nrow(XICs.ref[[1]]))
chrom <- getChildXICpp(XICs.ref, XICs.eXp, 11L, 4L, alignType = "hybrid", adaptiveRT = 77.82315,
 normalization = "mean", simType = "dotProductMasked", Bp = Bp,
 goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3, OverlapAlignment = TRUE,
 dotProdThresh = 0.96, gapQuantile = 0.5, hardConstrain = FALSE, samples4gradient = 100,
 wRef = 0.5, keepFlanks= TRUE)
chrom2 <- otherChildXICpp(XICs.ref, XICs.eXp, 11L, 4L, chrom[[2]], chrom[[1]][[1]][,1],
0.5, "natural")
```

---

paramsDIAlignR *Parameters for the alignment functions*

Description

Retention alignment requires OpenSWATH/pyProphet extracted features and chromatograms. This function provides a suite of parameters used for selecting features and manipulating chromatograms. Chromatogram alignment can be performed via reference based or progressively via rooted or unrooted tree. This function provides sensible parameters for these tasks.

Usage

```
paramsDIAlignR()
```

Value

A list of parameters:

runType	(string) must be one of the strings "DIA_Proteomics", "DIA_IPF", "DIA_Metabolomics".
chromFile	(string) must either be "mzML" or "sqMass".
maxFdrQuery	(numeric) a numeric value between 0 and 1. It is used to filter peptides from osw file which have SCORE_MS2.QVALUE less than itself.
maxIPFFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_IPF.QVALUE less than itself. (For PTM IPF use)
maxPeptideFdr	(numeric) a numeric value between 0 and 1. It is used to filter peptides from osw file which have SCORE_PEPTIDE.QVALUE less than itself.
analyteFDR	(numeric) the upper limit of feature FDR to be it considered for building tree.
treeDist	(string) the method used to build distance matrix. Must be either "rsquared", "count" or "RSE".
treeAgg	(string) the method used for agglomeration while performing hierarchical clustering. Must be either "single", "average" or "complete".

`alignToRoot` (logical) if TRUE, align leaves to the root in hierarchical clustering, else use already save aligned vectors.

`prefix` (string) name to be used to define merged runs.

`context` (string) used in pyprophet peptide. Must be either "run-specific", "experiment-wide", or "global".

`unalignedFDR` (numeric) must be between 0 and maxFdrQuery. Features below unalignedFDR are considered for quantification even without the RT alignment.

`alignedFDR1` (numeric) must be between unalignedFDR and alignedFDR2. Features below alignedFDR1 and aligned to the reference are considered for quantification.

`alignedFDR2` (numeric) must be between alignedFDR1 and maxFdrQuery. Features below alignedFDR2 and within certain distance from the aligned time are considered for quantification after the alignment.

`criterion` (integer) strategy to select peak if found overlapping peaks. 1:intensity, 2: RT overlap, 3: mscore, 4: edge distance

`level` (string) apply maxPeptideFDR on Protein as well if specified as "Protein". Default: "Peptide".

`integrationType` (string) method to ompute the area of a peak contained in XICs. Must be from "intensity_sum", "trapezoid", "simpson".

`baseSubtraction` logical TRUE: remove background from peak signal using estimated noise levels.

`baselineType` (string) method to estimate the background of a peak contained in XICs. Must be from "none", "base_to_base", "vertical_division_min", "vertical_division_max".

`fitEMG` (logical) enable/disable exponentially modified gaussian peak model fitting.

`recalIntensity` (logical) recalculate intensity for all analytes.

`fillMissing` (logical) calculate intensity for ananlytes for which features are not found.

`XICfilter` (string) must be either sgolay, boxcar, gaussian, loess or none.

`polyOrd` (integer) order of the polynomial to be fit in the kernel.

`kernelLen` (integer) number of data-points to consider in the kernel.

`globalAlignment` (string) must be either "loess" or "linear".

`globalAlignmentFdr` (numeric) a numeric value between 0 and 1. Features should have m-score lower than this value for participation in LOESS fit.

`globalAlignmentSpan` (numeric) spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

`RSEdistFactor` (numeric) defines how much distance in the unit of rse remains a noBeef zone.

`normalization` (string) must be selected from "mean", "l2".

`simMeasure` (string) must be selected from dotProduct, cosineAngle, crossCorrelation, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.

`alignType` (numeric) available alignment methods are "global", "local" and "hybrid".

goFactor	(numeric) penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty. Should be between 10-1000.
geFactor	(numeric) penalty for introducing subsequent gaps in alignment. This value is multiplied by base gap-penalty.
cosAngleThresh	(numeric) in simType = dotProductMasked mode, angular similarity should be higher than cosAngleThresh otherwise similarity is forced to zero.
OverlapAlignment	(logical) an input for alignment with free end-gaps. False: Global alignment, True: overlap alignment.
dotProdThresh	(numeric) in simType = dotProductMasked mode, values in similarity matrix higher than dotProdThresh quantile are checked for angular similarity.
gapQuantile	(numeric) must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
kerLen	(integer) In simType = crossCorrelation, length of the kernel used to sum similarity score. Must be an odd number.
hardConstrain	(logical) if FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) modulates penalization of masked indices.
fillMethod	(string) must be either "spline", "sgolay" or "linear".
splineMethod	(string) must be either "fmm" or "natural".
mergeTime	(string) must be either "ref", "avg", "refStart" or "refEnd".
keepFlanks	(logical) TRUE: Flanking chromatogram is not removed.
fraction	(integer) indicates which fraction to align.
fractionNum	(integer) Number of fractions to divide the alignment.
lossy	(logical) if TRUE, time and intensity are lossy-compressed in generated sqMass file.
useIdentifying	(logical) Set TRUE to use identifying transitions in alignment. (DEFAULT: FALSE)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

checkParams, alignTargetedRuns

Examples

```
params <- paramsDIAlignR()
```

perBatch *Aligns an analyte across runs*

Description

For the ith analyte in multipeptide, this function aligns all runs to the reference run. The result is a dataframe that contains aligned features corresponding to the analyte across all runs.

Usage

```
perBatch(
  iBatch,
  peptides,
  multipeptide,
  refRuns,
  precursors,
  prec2chromIndex,
  fileInfo,
  mzPntrs,
  params,
  globalFits,
  RSE,
  applyFun = lapply,
  multiFeatureAlignmentMap = NULL
)
```

Arguments

peptides	(integer) vector of peptide IDs.
multipeptide	(list) contains multiple data-frames that are collection of features associated with analytes. This is an output of getMultipeptide.
refRuns	(data-frame) output of getRefRun. Must have two columsn : transition_group_id and run.
precursors	(data-frame) atleast two columns transition_group_id and transition_ids are required.
prec2chromIndex	(list) a list of dataframes having following columns: transition_group_id: it is PRECURSOR.ID from osw file. chromatogramIndex: index of chromatogram in mzML file.
fileInfo	(data-frame) output of getRunNames.
mzPntrs	(list) a list of mzRpwiz.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
globalFits	(list) each element is either of class lm or loess. This is an output of getGlobalFits.
RSE	(list) Each element represents Residual Standard Error of corresponding fit in globalFits.

multiFeatureAlignmentMap
: (list) contains multiple data-frames that are collection of experiment feature ids mapped to corresponding reference feature id per analyte. This is an output of getRefExpFeatureMap.

rownum
: (integer) represnts the index of the multipepetide to be aligned.

Value

invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-07-26

See Also

alignTargetedRuns, alignToRef, getAlignedTimesFast, getMultipeptide

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
```

---

pickNearestFeature *Pick feature closest to reference peak*

Description

It picks a feature that is within adaptiveRT window across eXpRT and has lowest m-score. Feature's m-score also has to be smaller than featureFDR.

Usage

```
pickNearestFeature(eXpRT, analyte, oswFiles, runname, adaptiveRT, featureFDR)
```

Arguments

eXpRT
: (numeric) retention time in experiment run.

analyte
: (integer) vector of precursor IDs.

oswFiles
: (list of data-frames) it is output from getFeatures function.

runname
: (string) must be a combination of "run" and an iteger e.g. "run2".

adaptiveRT
: (numeric) half-width of retention time window. Feature, if found, is picked from within this window.

featureFDR
: (numeric) upper m-score cut-off for a feature to be picked.

Value

(list) Following elements are present in the list:

leftWidth	(numeric) as in FEATURE.LEFT_WIDTH of osw files.
rightWidth	(numeric) as in FEATURE.RIGHT_WIDTH of osw files.
RT	(numeric) retention time as in FEATURE.EXP_RT of osw files.
Intensity	(numeric) peak intensity as in FEATURE_MS2.AREA_INTENSITY of osw files.
peak_group_rank	(integer) rank of each feature associated with transition_group_id.
m_score	(numeric) q-value of each feature associated with transition_group_id.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getFeatures

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
## Not run:
pickNearestFeature(eXpRT = 5237.8, analyte = 4618L, oswFiles = oswFiles_DIAlignR,
 runname = "run2", adaptiveRT = 77.82315, featureFDR = 0.05)

## End(Not run)
```

---

plotAlignedAnalytes *Plot aligned XICs group for a specific peptide. AlignObjOutput is the output from getAlignObjs fucntion.*

Description

Plot aligned XICs group for a specific peptide. AlignObjOutput is the output from getAlignObjs fucntion.

Usage

```
plotAlignedAnalytes(
  AlignObjOutput,
  plotType = "All",
  outFile = "AlignedAnalytes.pdf",
  annotatePeak = FALSE,
  saveFigs = FALSE
)
```

Arguments

AlignObjOutput	(list) list contains fileInfo, AlignObj, raw XICs for reference and experiment, and reference-peak label.
plotType	(string) must be one of the strings "All", "onlyUnaligned" and "onlyAligned".
outFile	(string) name of the output pdf file.
annotatePeak	(logical) TRUE: Peak boundaries and apex will be highlighted.
saveFigs	(logical) TRUE: Figures will be saved in AlignedAnalytes.pdf .

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-13

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
AlignObjOutput <- getAlignObjs(analytes = 4618L, runs, dataPath = dataPath)
plotAlignedAnalytes(AlignObjOutput)
```

---

plotAlignmentPath *Visualize alignment path through similarity matrix*

Description

Plot aligned path through the similarity matrix. Reference run has indices on X-axis, eXp run has them on Y-axis. In getAlignObjs function, objType must be set to medium.

Usage

```
plotAlignmentPath(AlignObjOutput)
```

Arguments

AlignObjOutput	(list) The list contains AlignObj, raw XICs for reference and experiment, and reference-peak label.

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
library(lattice)
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
AlignObjOutput <- getAlignObjs(analytes = 4618L, runs, dataPath = dataPath, objType = "medium")
plotAlignmentPath(AlignObjOutput)
```

---

plotAnalyteXICs *Plot extracted-ion chromatogram.*

Description

Plot extracted-ion chromatogram.

Usage

```
plotAnalyteXICs(
  analyte,
  run,
  dataPath = ".",
  maxFdrQuery = 1,
  XICfilter = "sgolay",
  polyOrd = 4,
  kernelLen = 9,
  runType = "DIA_Proteomics",
  oswMerged = TRUE,
  peakAnnot = NULL,
  Title = NULL
)
```

Arguments

analyte	(integer) an analyte is a PRECURSOR.ID from the osw file.
run	(string) Name of a xics file without extension.
dataPath	(string) path to xics and osw directory.
maxFdrQuery	(numeric) A numeric value between 0 and 1. It is used to filter features from osw file which have SCORE_MS2.QVALUE less than itself.
XICfilter	(string) must be either sgolay, boxcar, gaussian, loess or none.
polyOrd	(integer) order of the polynomial to be fit in the kernel.
kernelLen	(integer) number of data-points to consider in the kernel.
runType	(char) This must be one of the strings "DIA_Proteomics", "DIA_Metabolomics".
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
peakAnnot	(numeric) Peak-apex time.
Title	(logical) TRUE: name of the list will be displayed as title.

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

plotXICgroup

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
run <- "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"
plotAnalyteXICs(analyte = 2474L, run, dataPath = dataPath, oswMerged = TRUE, XICfilter = "none")
plotAnalyteXICs(analyte = 2474L, run, dataPath = dataPath, oswMerged = TRUE, XICfilter = "sgolay")
```

plotSingleAlignedChrom *Plot an aligned XIC-group.*

Description

Plot an aligned XIC-group.

Usage

```
plotSingleAlignedChrom(XIC_group, idx, peakAnnot = NULL)
```

Arguments

XIC_group	(list) It is a list of dataframe which has two columns. First column is for time and second column indicates intensity.
idx	(integer) Indices of aligned chromatograms.
peakAnnot	(numeric) Peak-apex time.

Details

x-axis cannot have the same time-values, therefore, x-axis is indecized.

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

plotXICgroup *Plot Extracted-ion chromatogram group.*

Description

Plot Extracted-ion chromatogram group.

Usage

```
plotXICgroup(XIC_group, peakAnnot = NULL, Title = NULL)
```

Arguments

XIC_group	(list) It is a list of dataframe which has two columns. First column is for time and second column indicates intensity.
peakAnnot	(numeric) Peak-apex time.
Title	(logical) TRUE: name of the list will be displayed as title.

Value

A plot to the current device.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

plotAnalyteXICs

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
 "hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt")
XICs <- getXICs(analytes = 4618L, runs = runs, dataPath = dataPath, oswMerged = TRUE)
plotXICgroup(XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]])
XICs <- smoothXICs(XICs[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]],
  type = "sgolay", kernelLen = 13, polyOrd = 4)
plotXICgroup(XICs, Title = "Precursor 4618 \n
 run hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt")
```

---

populateReferenceExperimentFeatureAlignmentMap

Populate Alignment Feature Mapping Table

Description

Populate an alignment feature mapping table for star align method, to map aligned features in experiment runs to the reference run for a given analyte. Is a data.table generated from of getRefExpFeatureMap.

Usage

```
populateReferenceExperimentFeatureAlignmentMap(
  df,
  feature_alignment_mapping,
  tAligned,
  ref,
  eXp,
  analyte_chr
)
```

Arguments

df	(dataframe) a collection of features related to the peptide
feature_alignment_mapping	(data.table) contains experiment feature ids mapped to corresponding reference feature id per analyte. This is an output of getRefExpFeatureMap.
tAligned	(list) the first element corresponds to the aligned reference time, the second element is the aligned experiment time.
ref	(string) name of the reference run. Must be in the rownames of fileInfo.
eXp	(string) name of the run to be aligned to reference run. Must be in the rownames of fileInfo.
analyte_chr	(string) name of highest quality precursor/transition_group_id

Value

invisible NULL

Author(s)

Justin Sing, <justinc.sing@mail.utoronto.ca>

ORCID: 0000-0003-0386-0092

See Also

getRefExpFeatureMap, alignToRef

Examples

```
data(multipeptide_DIAlignR, package="DIAlignR")
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
params <- paramsDIAlignR()
df <- multipeptide_DIAlignR[["14383"]]
df$alignment_rank[2] <- 1L
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
## Not run:
# Use getAlignedTimes() to get tAligned.
```

```
alignObj <- testAlignObj()
tAligned <- alignedTimes2(alignObj, XICs.ref, XICs.eXp)
setAlignmentRank(df, refIdx = 3L, eXp = "run2", tAligned, XICs.eXp, params, adaptiveRT = 38.66)
# Use getRefExpFeatureMap() to get a list of feature_alignment_mapping tables
feature_alignment_mapping <- data.table::as.data.table(structure(list(
reference_feature_id = structure(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), class = "integer64"),
experiment_feature_id = structure(c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), class = "integer64")),
row.names = c(NA, -15L), class = c("data.table", "data.frame"), sorted = "reference_feature_id"))
populateReferenceExperimentFeatureAlignmentMap(df, feature_alignment_mapping, tAligned, ref = "run0", eXp = "run2

## End(Not run)
```

---

progAlignRuns *Peptide quantification through progressive alignment*

Description

This function expects osw and xics directories at dataPath. It first reads osw files and fetches chromatogram indices for each analyte. To perform alignment, first a crude guide-tree is built which can also be provided with newickTree parameter. As we traverse from the leaf-nodes to the root node, runs are aligned pairwise. The root node is named master1 that has average of all fragment ion chromatograms and identified peak-groups. These features are propagated back to leaf nodes and finally aligned features are written in the output file.

Usage

```
progAlignRuns(
  dataPath,
  params,
  outFile = "DIAlignR",
  ropenms = NULL,
  oswMerged = TRUE,
  scoreFile = NULL,
  runs = NULL,
  peps = NULL,
  newickTree = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
outFile	(string) name of the output file.
ropenms	(pyopenms module) get this python module through get_ropenms. Required only for chrom.mzML files.
oswMerged	(logical) TRUE if merged file from pyprophet is used.

scoreFile	(string) path to the peptide score file, needed when oswMerged is FALSE.
runs	(string) names of xics file without extension.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
newickTree	(string) guidance tree in newick format. Look up getTree.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignTargetedRuns

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
## Not run:
ropenms <- get_ropenms(condaEnv = "envName")
progAlignRuns(dataPath, params = params, outFile = "test3", ropenms = ropenms)
# Removing aligned vectors
file.remove(list.files(dataPath, pattern = "*_av.rds", full.names = TRUE))
# Removing temporarily created master chromatograms
file.remove(list.files(file.path(dataPath, "xics"), pattern = "^master[A-Za-z0-9]+\\.chrom\\.sqMass$", full.name
file.remove(file.path(dataPath, "test3.temp.RData"))
file.remove(file.path(dataPath, "master.merged.osw"))

## End(Not run)
```

---

progComb3 *Step 3 for progressive alignment*

Description

Step 3 for progressive alignment

Usage

```
progComb3(
  dataPath,
  params,
  outFile = "DIAlignR",
  oswMerged = TRUE,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
outFile	(string) name of the output file.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2021) + GPL-3 Date: 2021-03-03

See Also

progAlignRuns

---

progSplit2 *Step 2 for progressive alignment*

Description

Step 2 for progressive alignment

Usage

```
progSplit2(
  dataPath,
  params,
  outFile = "DIAlignR",
  oswMerged = TRUE,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
outFile	(string) name of the output file.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

progAlignRuns

---

progSplit4 *Step 4 for progressive alignment*

Description

Step 4 for progressive alignment

Usage

```
progSplit4(
  dataPath,
  params,
  outFile = "DIAlignR",
  oswMerged = TRUE,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
outFile	(string) name of the output file.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

progAlignRuns

---

progTree1 *Step 1 for progressive alignment*

Description

Step 1 for progressive alignment

Usage

```
progTree1(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  groups = NULL,
  oswMerged = TRUE,
  scoreFile = NULL,
  peps = NULL,
  runs = NULL,
  newickTree = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
groups	(data-frame) contains the run names and their categories/batch id to keep them on the same branch of the tree.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
scoreFile	(string) path to the peptide score file, needed when oswMerged is FALSE.
peps	(integer) ids of peptides to be aligned. If NULL, align all peptides.
runs	(string) names of xics file without extension.
newickTree	(string) guidance tree in newick format. Look up getTree.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2021) + GPL-3 Date: 2021-03-03

See Also

progAlignRuns

---

readMzMLHeader *Get chromatogram header from a mzML file*

Description

Get a table of chromatogram indices and respective transition IDs.

Usage

```
readMzMLHeader(mzmlName)
```

Arguments

mzmlName	(char) path to xics file.

Value

(A data-frame) It has 10 columns. The two important columns are:

chromatogramId	(integer) Fragment-ion ID that matches with transition ID in osw file.
chromatogramIndex	(integer) Index of chromatogram in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2019) + GPL-3 Date: 2019-12-13

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
mzmlName <-paste0(dataPath,"/xics/hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML")
## Not run:
chromHead <- readChromatogramHeader(mzmlName = mzmlName)

## End(Not run)
```

readSqMassHeader *Get chromatogram header from a sqMass file*

Description

Get a table of chromatogram indices and respective transition IDs.

Usage

```
readSqMassHeader(con)
```

Arguments

mzmlName	(char) path to xics file.

Value

(A data-frame) It has 10 columns. The two important columns are:

chromatogramId	(integer) Fragment-ion ID that matches with transition ID in osw file.
chromatogramIndex	(integer) Index of chromatogram in mzML file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
sqName <-paste0(dataPath,"/xics/hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.sqMass")
## Not run:
chromHead <- readChromatogramHeader(sqName)

## End(Not run)
```

`recalculateIntensity` *Calculates area of peaks in peakTable*

Description

For the give peak boundary in peakTable, the function extracts raw chromatograms and recalculate intensities.

Usage

```
recalculateIntensity(
  peakTable,
  dataPath = ".",
  oswMerged = TRUE,
  params = paramsDIAlignR()
)
```

Arguments

`peakTable`	(data-table) usually an output of alignTargetedRuns. Must have these columns: precursor, run, intensity, leftWidth, rightWidth.
`dataPath`	(string) path to xics and osw directory.
`oswMerged`	(logical) TRUE if merged file from pyprophet is used.
`params`	(list) parameters are entered as list. Output of the `paramsDIAlignR` function.

Value

(data-table)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`alignTargetedRuns`, `calculateIntensity`

Examples

```
library(data.table)
peakTable <- data.table(precursor = c(1967L, 1967L, 2474L, 2474L),
                  run = rep(c("hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt",
                  "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"), 2),
                  intensity = c(186.166, 579.832, 47.9525, 3.7413),
                  leftWidth = c(5001.76, 5025.66, 6441.51, 6516.6),
```

```
                    rightWidth = c(5076.86, 5121.25, 6475.65, 6554.2))
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
params$smoothPeakArea <- TRUE
recalculateIntensity(peakTable, dataPath, params = params)
peakTable <- data.table(precursor = c(1967L, 1967L, 2474L, 2474L),
                    run = rep(c("hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt",
                    "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"), 2),
                    intensity = list(NA, NA, NA, NA),
                    leftWidth = c(5001.76, 5025.66, 6441.51, 6516.6),
                    rightWidth = c(5076.86, 5121.25, 6475.65, 6554.2))
params$transitionIntensity <- TRUE
recalculateIntensity(peakTable, dataPath, params = params)
```

`reduceXICs` *Subset an XIC file*

Description

Create a sqMass file that has chromatograms for given native IDs.

Usage

```
reduceXICs(nativeIDs, xicFileIn, xicFileOut)
```

Arguments

`nativeIDs`	(integer) transition IDs to be kept.
`xicFileIn`	(character) name to the current file.
`xicFileOut`	(character) name of the new file.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2022) + GPL-3 Date: 2022-04-19

See Also

`createSqMass`, `getNativeIDs`

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
oswIn <- file.path(dataPath, "osw", "merged.osw")
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
ids <- getNativeIDs(oswIn, 1338L, params)
xicFileIn <- file.path(dataPath, "xics", "hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.sqMass")
reduceXICs(ids, xicFileIn, xicFileOut = "temp.chrom.sqMass")
file.remove("temp.chrom.sqMass")
```

---

script1 *Extract features and generate pairwise alignments.*

Description

Extract features and generate pairwise alignments.

Usage

```
script1(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  runs = NULL,
  applyFun = lapply
)
```

Arguments

dataPath	(string) path to xics and osw directory.
outFile	(string) name of the output file.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
oswMerged	(logical) TRUE if merged file from pyprophet is used.
runs	(string) names of xics file without extension.
applyFun	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignTargetedRuns

Examples

```
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
dataPath <- system.file("extdata", package = "DIAlignR")
BiocParallel::register(BiocParallel::MulticoreParam(workers = 4, progressbar = TRUE))
script1(dataPath, outFile = "testDIAlignR", params = params, applyFun = BiocParallel::bplapply)
file.remove(file.path(dataPath, "testDIAlignR_script1.RData"))
```

`script2` *Performs alignment using script1 output*

Description

Performs alignment using script1 output

Usage

```
script2(
  dataPath,
  outFile = "DIAlignR",
  params = paramsDIAlignR(),
  oswMerged = TRUE,
  scoreFile = NULL,
  peps = NULL,
  refRun = NULL,
  applyFun = lapply
)
```

Arguments

`dataPath`	(string) path to xics and osw directory.
`outFile`	(string) name of the output file.
`params`	(list) parameters are entered as list. Output of the `paramsDIAlignR` function.
`oswMerged`	(logical) TRUE if merged file from pyprophet is used.
`scoreFile`	(string) path to the peptide score file, needed when oswMerged is FALSE.
`peps`	(integer) ids of peptides to be aligned. If NULL, align all peptides.
`refRun`	(string) reference for alignment. If no run is provided, m-score is used to select reference run.
`applyFun`	(function) value must be either lapply or BiocParallel::bplapply.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

alignTargetedRuns

Examples

```
params <- paramsDIAlignR()
params[["context"]] <- "experiment-wide"
dataPath <- system.file("extdata", package = "DIAlignR")
BiocParallel::register(BiocParallel::MulticoreParam(workers = 4, progressbar = TRUE))
script1(dataPath, outFile = "testDIAlignR", params = params, applyFun = BiocParallel::bplapply)
script2(dataPath, outFile = "testDIAlignR", params = params, applyFun = lapply)
file.remove(file.path(dataPath, "testDIAlignR_script1.RData"))
```

---

setAlignmentRank *Set Alignment rank to the aligned feature*

Description

Picks the top feature in run by comparing m-score to unaligned FDR and aligned FDR. If no satisfactory feature is found, peak-integration is carried out by mapping left and right peak boundaries from the reference feature and integrating area under the curve.

Usage

```
setAlignmentRank(df, refIdx, eXp, tAligned, XICs, params, adaptiveRT)
```

Arguments

df	(dataframe) a collection of features related to the peptide
refIdx	(integer) index of the reference feature in df.
eXp	(string) name of the run to be aligned to reference run. Must be in the rownames of fileInfo.
tAligned	(list) the first element corresponds to the aligned reference time, the second element is the aligned experiment time.
XICs	(list of dataframes) fragment-ion chromatograms of the analytes for all runs.
params	(list) parameters are entered as list. Output of the paramsDIAlignR function.
adaptiveRT	(numeric) defines the window around the aligned retention time, within which features with m-score below aligned FDR are considered for quantification.
XICs.eXp	(list) list of extracted ion chromatograms from experiment run.

Value

invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getMultipeptide, calculateIntensity, alignToRef

Examples

```
data(multipeptide_DIAlignR, package="DIAlignR")
data(XIC_QFNNTDIVLLEDFQK_3_DIAlignR, package="DIAlignR")
params <- paramsDIAlignR()
df <- multipeptide_DIAlignR[["14383"]]
df$alignment_rank[2] <- 1L
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
## Not run:
# Use getAlignedTimes() to get tAligned.
alignObj <- testAlignObj()
tAligned <- alignedTimes2(alignObj, XICs.ref, XICs.eXp)
setAlignmentRank(df, refIdx = 3L, eXp = "run2", tAligned, XICs.eXp,
params, adaptiveRT = 38.66)

## End(Not run)
```

---

sgolayCpp *Smooth chromatogram with savitzky-golay filter.*

Description

Smooth chromatogram with savitzky-golay filter.

Usage

```
sgolayCpp(chrom, kernelLen, polyOrd)
```

Arguments

chrom	(matrix) chromatogram containing time and intensity vectors.
kernelLen	(integer) length of filter. Must be an odd number.
polyOrd	(integer) TRUE: remove background from peak signal using estimated noise levels.

Value

(matrix).

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2020) + MIT Date: 2019-12-31

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR", package = "DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt"]][["4618"]]
xic <- sgolayCpp(as.matrix(XICs[[1]]), kernelLen = 11L, polyOrd = 4L)
```

---

sgolayFill *Fill missing values using Savitzky–Golay*

Description

Fill missing values using Savitzky–Golay

Usage

```
sgolayFill(chrom, polyOrd, kernelLen)
```

Arguments

chrom	(data-frame) first column is time (must be equidistant), second column is intensity.
polyOrd	(integer) must be less than kernelLen.
kernelLen	(integer) must be an odd integer.

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-05-21

See Also

polyfit, sgolayfilt

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
 4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom$y[c(1,8)] <- NA
## Not run:
sgolayFill(chrom, polyOrd = 3, kernelLen = 9)

## End(Not run)
```

---

smoothSingleXIC *Smooth chromatogram signal*

Description

Smoothing methods are Savitzky-Golay, Boxcar, Gaussian kernel and LOESS. Savitzky-Golay smoothing is good at preserving peak-shape compared to gaussian and boxcar smoothing. However, it assumes equidistant points that fortunately is the case for DIA data. This requires a quadratic memory to store the fit and slower than other smoothing methods.

Usage

```
smoothSingleXIC(
  chromatogram,
  type,
  samplingTime = NULL,
  kernelLen = NULL,
  polyOrd = NULL
)
```

Arguments

chromatogram	(dataframe) A dataframe of two columns. First column must always be monotonically increasing.
type	(char) must be either sgolay, boxcar, gaussian, loess or none.
samplingTime	(numeric) Time difference between neighboring points.
kernelLen	(integer) Number of data-points to consider in the kernel.
polyOrd	(integer) Order of the polynomial to be fit in the kernel.

Details

Gaussian smoothing uses a gaussian function whose bandwidth is scaled by 0.3706505 to have quartiles at +/- 0.25*bandwidth. The point selection cut-off is also hard at 0.3706505*4*bandwidth.

qnorm(0.75, sd = 0.3706505)

The definition of C_ksmooth can be found using getAnywhere('C_ksmooth') stats:::C_ksmooth

Value

A dataframe with two columns.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

https://terpconnect.umd.edu/~toh/spectrum/Smoothing.html, https://rafalab.github.io/dsbook/smoothing.html, https://github.com/SurajGupta/r-source/blob/master/src/library/stats/src/ksmooth.c

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR")
chrom <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]][["4618"]][[1]]
## Not run:
newChrom <- smoothSingleXIC(chrom, type = "sgolay", samplingTime = 3.42, kernelLen = 9,
 polyOrd = 3)

## End(Not run)
```

---

smoothXICs *Smooth chromatogram signals from a list*

Description

Smoothing methods are Savitzky-Golay, Boxcar, Gaussian kernel and LOESS. Savitzky-Golay smoothing is good at preserving peak-shape compared to gaussian and boxcar smoothing. However, it assumes equidistant points that fortunately is the case for DIA data. This requires a quadratic memory to store the fit and slower than other smoothing methods.

Usage

```
smoothXICs(
  XICs,
  type = "none",
  samplingTime = NULL,
  kernelLen = 9L,
  polyOrd = NULL
)
```

Arguments

XICs	(A list) A list of dataframe that consists of two columns. First column must be monotonically increasing.
type	(char) must be either sgolay, boxcar, gaussian, loess or none.
samplingTime	(numeric) Time difference between neighboring points.
kernelLen	(integer) Number of data-points to consider in the kernel.
polyOrd	(integer) Order of the polynomial to be fit in the kernel.

Value

A list.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

https://terpconnect.umd.edu/~toh/spectrum/Smoothing.html, https://rafalab.github.io/dsbook/smoothing.html

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]][["4618"]]
newXICs <- smoothXICs(XICs, type = "sgolay", samplingTime = 3.42, kernelLen = 9,
 polyOrd = 3)
```

---

splineFill *Fill missing values using spline*

Description

Fill missing values using spline

Usage

```
splineFill(chrom, method = "fmm")
```

Arguments

chrom	(data-frames) first column is time, second column is intensity.
method	(string) must be either "fmm" or "natural".

Value

(dataframe) has two columns:

time	(numeric)
intensity	(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-05-21

See Also

na.spline, spline

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
 4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
chrom <- data.frame(time, y)
chrom$y[c(1,8)] <- NA
## Not run:
splineFill(chrom)

## End(Not run)
```

---

splineFillCpp *Interpolate using spline*

Description

Interpolate using spline

Usage

```
splineFillCpp(x, y, xout)
```

Arguments

x	(numeric) A numeric matrix with similarity values of two sequences or signals.
y	(numeric) Penalty for introducing first gap in alignment.
xout	(numeric) Penalty for introducing subsequent gaps in alignment.

Value

(numeric)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca> ORCID: 0000-0003-3500-8152 License: (c) Author (2021) + MIT Date: 2021-01-06

Examples

```
time <- seq(from = 3003.4, to = 3048, by = 3.4)
y <- c(0.2050595, 0.8850070, 2.2068768, 3.7212677, 5.1652605, 5.8288915, 5.5446804,
       4.5671360, 3.3213154, 1.9485889, 0.9520709, 0.3294218, 0.2009581, 0.1420923)
y[c(1,6)] <- NA_real_
idx <- !is.na(y)
splineFillCpp(time[idx], y[idx], time[!idx])
zoo::na.spline(zoo::zoo(y[idx], time[idx]), xout = time[!idx], method = "natural")
```

---

traverseDown *Traverses down from the root to leaves*

Description

Features of the root node are propagated to all leaves node. Aligned features are set/added in the multipeptide environment.

Usage

```
traverseDown(
  tree,
  dataPath,
  fileInfo,
  multipeptide,
  prec2chromIndex,
  mzPntrs,
  precursors,
  adaptiveRTs,
  refRuns,
  params,
  applyFun = lapply
)
```

Arguments

tree	(phylo) a phylogenetic tree.
dataPath	(string) path to xics and osw directory.
fileInfo	(data-frame) output of getRunNames.
multipeptide	(environment) contains multiple data-frames that are collection of features associated with analytes. This is an output of getMultipeptide.

prec2chromIndex
: (list) a list of dataframes having following columns:
transition_group_id: it is PRECURSOR.ID from osw file.
chromatogramIndex: index of chromatogram in mzML file.

mzPntrs
: (list) a list of mzRpwiz.

precursors
: (data-frame) atleast two columns transition_group_id and transition_ids are required.

adaptiveRTs
: (environment) For each descendant-pair, it contains the window around the aligned retention time, within which features with m-score below aligned FDR are considered for quantification.

refRuns
: (environment) For each descendant-pair, the reference run is indicated by 1 or 2 for all the peptides.

params
: (list) parameters are entered as list. Output of the paramsDIAlignR function.

applyFun
: (function) value must be either lapply or BiocParallel::bplapply.

analytes
: (integer) this vector contains transition_group_id from precursors. It must be of the same length as of multipeptide.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

traverseUp, alignToMaster

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
fileInfo <- getRunNames(dataPath = dataPath)
mzPntrs <- list2env(getMZMLpointers(fileInfo))
features <- list2env(getFeatures(fileInfo, maxFdrQuery = params[["maxFdrQuery"]], runType = params[["runType"]]))
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, runType = params[["runType"]],
 context = "experiment-wide", maxPeptideFdr = params[["maxPeptideFdr"]])
precursors <- dplyr::arrange(precursors, .data$peptide_id, .data$transition_group_id)
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, oswMerged = TRUE, params[["runType"]], params[["context"]
peptideScores <- lapply(peptideIDs, function(pep) dplyr::filter(peptideScores, .data$peptide_id == pep))
names(peptideScores) <- as.character(peptideIDs)
prec2chromIndex <- list2env(getChromatogramIndices(fileInfo, precursors, mzPntrs))
multipeptide <- getMultipeptide(precursors, features)
adaptiveRTs <- new.env()
```

```
refRuns <- new.env()
tree <- ape::read.tree(text = "(run1:9,(run2:7,run0:2)master2:5)master1;")
tree <- ape::reorder.phylo(tree, "postorder")
## Not run:
ropenms <- get_ropenms(condaEnv = "envName", useConda=TRUE)
multipeptide <- traverseUp(tree, dataPath, fileInfo, features, mzPntrs, prec2chromIndex, precursors, params,
 adaptiveRTs, refRuns, multipeptide, peptideScores, ropenms)
multipeptide <- getMultipeptide(precursors, features)
multipeptide <- traverseDown(tree, dataPath, fileInfo, multipeptide, prec2chromIndex, mzPntrs, precursors,
 adaptiveRTs, refRuns, params)
# Cleanup
for(run in names(mzPntrs)) DBI::dbDisconnect(mzPntrs[[run]])
file.remove(list.files(dataPath, pattern = "*_av.rds", full.names = TRUE))
file.remove(list.files(file.path(dataPath, "xics"), pattern = "^master[0-9]+\\.chrom\\.sqMass$", full.names = TRU

## End(Not run)
```

`traverseMST` *Reorder MST*

Description

Reorder MST for traversal from a given node.

Usage

```
traverseMST(net, ref0)
```

Arguments

`net`	(matrix) array of tree-edges.
`ref0`	(character) start node for traversal.

Value

(matrix) array of tree-edges reordered for traversal.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2021) + GPL-3 Date: 2021-05-14

See Also

`mstAlignRuns`, `getMST`

Examples

```
net <- cbind("A" = c("run1","run2","run3"), "B" = c("run2","run3","run4"))
## Not run:
traverseMST(net, "run2")

## End(Not run)
```

---

traverseUp *Traverses up from leaves to the root*

Description

While traversing from leaf to root node, at each node a master run is created. Merged features and merged chromatograms from parent runs are estimated. Chromatograms are written on the disk at dataPath/xics. For each precursor aligned parent time-vectors and corresponding child time-vector are also calculated and written as *_av.rds at dataPath.

Accesors to the new files are added to fileInfo, mzPntrs and prec2chromIndex. Features, reference used for the alignment and adaptiveRTs of global alignments are also added to corresponding environment.

Usage

```
traverseUp(
  tree,
  dataPath,
  fileInfo,
  features,
  mzPntrs,
  prec2chromIndex,
  precursors,
  params,
  adaptiveRTs,
  refRuns,
  multipeptide,
  peptideScores,
  ropenms,
  applyFun = lapply
)
```

Arguments

tree	(phylo) a phylogenetic tree.
dataPath	(string) path to xics and osw directory.
fileInfo	(data-frame) output of getRunNames.
features	(list of data-frames) contains features and their properties identified in each run.
mzPntrs	(list) a list of mzRpwiz.

prec2chromIndex
: (list) a list of dataframes having following columns:
transition_group_id: it is PRECURSOR.ID from osw file.
chromatogramIndex: index of chromatogram in mzML file.

precursors
: (data-frame) atleast two columns transition_group_id and transition_ids are required.

params
: (list) parameters are entered as list. Output of the paramsDIAlignR function.

adaptiveRTs
: (environment) For each descendant-pair, it contains the window around the aligned retention time, within which features with m-score below aligned FDR are considered for quantification.

refRuns
: (environment) For each descendant-pair, the reference run is indicated by 1 or 2 for all the peptides.

multipeptide
: (environment) contains multiple data-frames that are collection of features associated with analytes. This is an output of getMultipeptide.

peptideScores
: (list of data-frames) each dataframe has scores of a peptide across all runs.

ropenms
: (pyopenms module) get this python module through get_ropenms. Required only for chrom.mzML files.

applyFun
: (function) value must be either lapply or BiocParallel::bplapply.

Value

(None)

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getTree, getNodeRun

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
params <- paramsDIAlignR()
fileInfo <- getRunNames(dataPath = dataPath)
mzPntrs <- list2env(getMZMLpointers(fileInfo))
features <- list2env(getFeatures(fileInfo, maxFdrQuery = params[["maxFdrQuery"]], runType = params[["runType"]]))
precursors <- getPrecursors(fileInfo, oswMerged = TRUE, runType = params[["runType"]],
 context = "experiment-wide", maxPeptideFdr = params[["maxPeptideFdr"]])
precursors <- dplyr::arrange(precursors, .data$peptide_id, .data$transition_group_id)
peptideIDs <- unique(precursors$peptide_id)
peptideScores <- getPeptideScores(fileInfo, peptideIDs, oswMerged = TRUE, params[["runType"]], params[["context"]
peptideScores <- lapply(peptideIDs, function(pep) dplyr::filter(peptideScores, .data$peptide_id == pep))
names(peptideScores) <- as.character(peptideIDs)
```

```
peptideScores <- list2env(peptideScores)
multipeptide <- getMultipeptide(precursors, features)
prec2chromIndex <- list2env(getChromatogramIndices(fileInfo, precursors, mzPntrs))
adaptiveRTs <- new.env()
refRuns <- new.env()
tree <- ape::read.tree(text = "(run1:9,(run2:7,run0:2)master2:5)master1;")
tree <- ape::reorder.phylo(tree, "postorder")
## Not run:
ropenms <- get_ropenms(condaEnv = "envName", useConda=TRUE)
multipeptide <- traverseUp(tree, dataPath, fileInfo, features, mzPntrs, prec2chromIndex, precursors, params,
 adaptiveRTs, refRuns, multipeptide, peptideScores, ropenms)
for(run in names(mzPntrs)) DBI::dbDisconnect(mzPntrs[[run]])
# Cleanup
file.remove(list.files(dataPath, pattern = "*_av.rds", full.names = TRUE))
file.remove(list.files(file.path(dataPath, "xics"), pattern = "^master[0-9]+\\.chrom\\.sqMass$", full.names = TRU

## End(Not run)
```

`trfrParentFeature` *Transform features to child time-domain*

Description

This function transforms the peaks' times to child run's time-domain. The feature intensity is calculated with appropriate method stated in params. Internal missing values are not allowed in timeParent.

Usage

```
trfrParentFeature(XICs, timeParent, df, i, params)
```

Arguments

`XICs`	(list of data-frames) extracted ion chromatograms from the child run.
`timeParent`	(data-frame) has two columns: tAligned and alignedChildTime. tAligned is time vector from one of the parent run.
`df`	(data-frame) contains features to be transformed. It has a format of `getFeatures` output.
`params`	(list) parameters are entered as list. Output of the `paramsDIAlignR` function.

Value

(data-frame) this has a format of `getFeatures` output.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

getChildFeature

Examples

```
data(masterXICs_DIAlignR, package="DIAlignR")
newXICs <- masterXICs_DIAlignR
timeParent <- newXICs[[2]][, c("tAligned.ref", "alignedChildTime")]
colnames(timeParent) <- c("tAligned", "alignedChildTime")
params <- paramsDIAlignR()
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_Proteomics")
df <- features$run1[features$run1$transition_group_id == 4618L, ]
## Not run:
trfrParentFeature(newXICs[[1]], timeParent, df, params)

## End(Not run)
```

---

trimXICs *Selects a part of chromatograms*

Description

This function trims chromatograms from the end-points.

Usage

```
trimXICs(XICs, len = 1)
```

Arguments

XICs	(A list) A list of dataframe that consists of two columns. First column must be monotonically increasing.
len	(numeric) must be between 0.1 and 1.

Value

A list.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt"]][["4618"]]
## Not run:
newXICs <- smoothXICs(XICs, len = 0.5)

## End(Not run)
```

`uncompressVec` *Uncompress a Blob object*

Description

compression is one of 0 = no, 1 = zlib, 2 = np-linear, 3 = np-slof, 4 = np-pic, 5 = np-linear + zlib, 6 = np-slof + zlib, 7 = np-pic + zlib

Usage

```
uncompressVec(x, type)
```

Arguments

`x`	(Blob object)
`type`	(integer) must either be 5L or 6L to indicate linear and short logged float compression, respectively.

Value

A numeric vector. Uncompressed form of the Blob.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
sqName <- paste0(dataPath,"/xics/hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.sqMass")
con <- DBI::dbConnect(RSQLite::SQLite(), dbname = sqName)
df1 <- DBI::dbGetQuery(con, "SELECT CHROMATOGRAM_ID, COMPRESSION, DATA_TYPE, DATA FROM DATA WHERE CHROMATOGRAM_ID =
DBI::dbDisconnect(con)
## Not run:
time = uncompressVec(df1[["DATA"]][[1]], df1$COMPRESSION[[1]])
intensity = uncompressVec(df1[["DATA"]][[2]], df1$COMPRESSION[[2]])

## End(Not run)
```

updateFileInfo *Get intersection of runs and fileInfo*

Description

Get intersection of runs and fileInfo

Usage

```
updateFileInfo(fileInfo, runs = NULL)
```

Arguments

fileInfo	(data-frame) output of getRunNames function.
runs	(vector of string) names of mzML files without extension.

Value

(dataframe) it has five columns:

spectraFile	(string) as mentioned in RUN table of osw files.
runName	(string) contain respective mzML names without extension.
spectraFileID	(string) ID in RUN table of osw files.
featureFile	(string) path to the feature file.
chromatogramFile	(string) path to the chromatogram file.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

License: (c) Author (2020) + GPL-3 Date: 2020-04-15

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- getRunNames(dataPath = dataPath, oswMerged = TRUE)
runs <- c("hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt",
          "hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt")
updateFileInfo(fileInfo, runs)
```

`updateOnalignTargetedRuns` *Prints messages if a certain number of analytes are aligned*

Description

Prints messages if a certain number of analytes are aligned

Usage

```
updateOnalignTargetedRuns(i)
```

Value

Invisible NULL

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

`writeOutFeatureAlignmentMap` *Write out alignment map to disk*

Description

Save alignment mapping to disk, either append table to OSW file, or save TSV file(s)

Usage

```
writeOutFeatureAlignmentMap(multiFeatureAlignmentMap, oswMerged, fileInfo)
```

Arguments

`multiFeatureAlignmentMap`	(list) contains multiple data-frames that are collection of experiment feature ids mapped to corresponding reference feature id per analyte. This is an output of `getRefExpFeatureMap`.
`oswMerged`	(logical) TRUE if merged file from pyprophet is used.
`fileInfo`	(data-frame) Output of DIAlignR::getRunNames function

Value

Saves the alignment feature id mapping table.

The mapping table will have the following columns:

ALIGNMENT_GROUP_ID: (int) An interger number that identifies the group of experiments that are aligned per best representative precursor (peptide).

REFERENCE: (logical int) A logical interger, 1 indicates the feature used as the reference, 0 indicates the experiment feature being aligned to reference.

FEATURE_ID: (int64) Feature id derived from OpenSwathWorkflow's peak-group picking annotation, in OSW file.

Writing to disk will be one of two possible outcomes:

1. If fileInfo contains a merged OSW, then the alignment map table will be written to the sqlite database as ALIGNMENT_GROUP_FEATURE_MAPPING.

or

2. If fileInfo does not contain a merged OSW, then the alignment map table will be written to a TSV file.

Author(s)

Justin Sing, <justinc.sing@mail.utoronto.ca>

ORCID: 0000-0003-0386-0092

See Also

getRefExpFeatureMap, getRunNames

Examples

```
data(oswFiles_DIAlignR, , package="DIAlignR")
## Not run:
writeOutFeatureAlignmentMap(multiFeatureAlignmentMap, oswMerged, fileInfo)

## End(Not run)
```

---

writeTables *Writes the output table post-alignment*

Description

Selects all features from multipeptide with alignment rank = 1, and write them onto the disk.

Usage

```
writeTables(fileInfo, multipeptide, precursors)
```

Arguments

`fileInfo`	(data-frame) Output of getRunNames function.
`multipeptide`	(list of data-frames) Each element of the list is collection of features associated with a precursor.
`precursors`	(data-frame) for each transition_group_id, contains peptide sequence and charge.
`filename`	(string) Name of the output file.

Value

An output table with following columns: precursor, run, intensity, RT, leftWidth, rightWidth, peak_group_rank, m_score, alignment_rank, peptide_id, sequence, charge, group_label.

Author(s)

Shubham Gupta, <shubh.gupta@mail.utoronto.ca>

ORCID: 0000-0003-3500-8152

See Also

`getRunNames`, `getMultipeptide`, `getPrecursors`

Examples

```
data(oswFiles_DIAlignR, package="DIAlignR")
## Not run:
writeTables(fileInfo, multipeptide, precursors)

## End(Not run)
```

XIC_QFNNTDIVLLEDFQK_3_DIAlignR

Extracted-ion chromatograms (XICs) of a peptide

Description

XICs of peptide QFNNTDIVLLEDFQK/3 (precursor ID: 4618) from three SWATH runs:

run0 : hroest_K120808_Strep10%PlasmaBiolRepl1_R03_SW_filt.chrom.mzML
run1 : hroest_K120809_Strep0%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML
run2 : hroest_K120809_Strep10%PlasmaBiolRepl2_R04_SW_filt.chrom.mzML

Usage

```
XIC_QFNNTDIVLLEDFQK_3_DIAlignR
```

Format

A list of three elements where each element consists of a list of six data frames. Each data frame has two columns:

time Retention time of ananlyte in the run, in sec

intensity Intensity of signal for the transition

Source

Raw files are downloaded from Peptide Atlas. File test_GenerateData.R has source code to generate the example data.

Index