Package 'DIAlignR'

November 30, 2020

Type Package

Title Dynamic Programming Based Alignment of MS2 Chromatograms

Version 1.2.0

Author Shubham Gupta <shubham.1637@gmail.com>, Hannes Rost <hannes.rost@utoronto.ca>

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

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 7.1.0

biocViews MassSpectrometry, Metabolomics, Proteomics, Alignment, Software

Depends methods, stats

Imports zoo (>= 1.8-3), dplyr, tidyr, rlang, mzR (>= 2.18), signal, ggplot2, scales, gridExtra, RSQLite, DBI, Rcpp

Suggests knitr, lattice, latticeExtra, rmarkdown, BiocStyle, testthat (>= 2.1.0)

VignetteBuilder knitr

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

LinkingTo Rcpp

SystemRequirements C++11

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

git_branch RELEASE_3_12

git_last_commit 25b9e1a

git_last_commit_date 2020-10-27

Date/Publication 2020-11-29

R topics documented:

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

 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,
  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). 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.
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 License: (c) Author (2019) + MIT 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[["run1"]][["14299_QFNNTDIVLLEDFQK/3"]]
XICs.eXp <- XICs[["run2"]][["14299_QFNNTDIVLLEDFQK/3"]]
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")
```

`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`

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

Description

This function expects osw and mzml 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.csv",
  oswMerged = TRUE,
  runs = NULL,
  runType = "DIA_Proteomics",
  maxFdrQuery = 0.05,
  XICfilter = "sgolay",
  polyOrd = 4,
  kernelLen = 9,
  globalAlignment = "loess",
  globalAlignmentFdr = 0.01,
  globalAlignmentSpan = 0.1,
  RSEdistFactor = 3.5,
  normalization = "mean",
  simMeasure = "dotProductMasked",
  alignType = "hybrid",
  goFactor = 0.125,
  geFactor = 40,
  cosAngleThresh = 0.3,
  OverlapAlignment = TRUE,
  dotProdThresh = 0.96,
  gapQuantile = 0.5,
  hardConstrain = FALSE,
  samples4gradient = 100,
  analyteFDR = 0.01,
  unalignedFDR = 0.01,
  alignedFDR = 0.05,
  baselineType = "base_to_base",
  integrationType = "intensity_sum",
  fitEMG = FALSE,
  recalIntensity = FALSE,
  fillMissing = TRUE,
  smoothPeakArea = FALSE
)
```

Arguments

`dataPath`	(string) path to mzml and osw directory.
`outFile`	(string) name of the output file.

oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.
runs	(A vector of string) names of mzml file without extension.
runType	(string) must be one of the strings "DIA_proteomics", "DIA_Metabolomics".
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.
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 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	(character) must be selected from "mean", "l2".
simMeasure	(string) must be selected from dotProduct, cosineAngle, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.
alignType	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.
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.
hardConstrain	(logical) if FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) modulates penalization of masked indices.
analyteFDR	(numeric) defines the upper limit of FDR on a precursor to be considered for multipeptide.
unalignedFDR	(numeric) must be between 0 and maxFdrQuery. Features below unalignedFDR are considered for quantification even without the RT alignment.
alignedFDR	(numeric) must be between unalignedFDR and 1. Features below alignedFDR are considered for quantification after the alignment.
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".

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

`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.

`smoothPeakArea` (logical) FALSE: raw chromatograms will be used for quantification. TRUE: smoothed chromatograms will be used for quantification.

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

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

```
dataPath <- system.file("extdata", package = "DIAlignR")
alignTargetedRuns(dataPath, outFile = "testDIAlignR.csv", oswMerged = TRUE)
```

---

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)
```

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.

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[["run1"]][["14299_QFNNTDIVLLEDFQK/3"]]
l1 <- lapply(XICs, `[[`, 1)
l2 <- lapply(XICs, `[[`, 2)
areaIntegrator(l1, l2, left = 5203.7, right = 5268.5, "intensity_sum", "base_to_base", FALSE)
# 224.9373
```

---

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

---

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

---

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

`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)
```

`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

`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 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)
```

`getAlignedIndices` *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

```
getAlignedIndices(
  XICs.ref,
  XICs.eXp,
  globalFit,
  alignType,
  adaptiveRT,
  normalization,
  simMeasure,
  goFactor,
  geFactor,
  cosAngleThresh,
  OverlapAlignment,
  dotProdThresh,
  gapQuantile,
  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, 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.
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")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["run1"]][["14299_QFNNTDIVLLEDFQK/3"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["run2"]][["14299_QFNNTDIVLLEDFQK/3"]]
globalFit <- getGlobalAlignment(oswFiles_DIAlignR, ref = "run2", eXp = "run0",
 maxFdrGlobal = 0.05, spanvalue = 0.1)
adaptiveRT <- 77.82315 #3.5*globalFit$s
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, hardConstrain = FALSE,
  samples4gradient = 100)
```

---

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,
  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, 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.
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")
XICs.ref <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["run1"]][["14299_QFNNTDIVLLEDFQK/3"]]
XICs.eXp <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["run2"]][["14299_QFNNTDIVLLEDFQK/3"]]
globalFit <- getGlobalAlignment(oswFiles_DIAlignR, ref = "run1", eXp = "run2",
 maxFdrGlobal = 0.05, spanvalue = 0.1)
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, hardConstrain = FALSE, samples4gradient = 100, objType = "light")
```

---

getAlignObjs *AlignObj for analytes between a pair of runs*

Description

This function expects osw and mzml 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,
  runType = "DIA_Proteomics",
  maxFdrQuery = 0.05,
  analyteFDR = 0.01,
```

```
  XICfilter = "sgolay",
  polyOrd = 4,
  kernelLen = 9,
  globalAlignment = "loess",
  globalAlignmentFdr = 0.01,
  globalAlignmentSpan = 0.1,
  RSEdistFactor = 3.5,
  normalization = "mean",
  simMeasure = "dotProductMasked",
  alignType = "hybrid",
  goFactor = 0.125,
  geFactor = 40,
  cosAngleThresh = 0.3,
  OverlapAlignment = TRUE,
  dotProdThresh = 0.96,
  gapQuantile = 0.5,
  hardConstrain = FALSE,
  samples4gradient = 100,
  objType = "light"
)
```

Arguments

analytes	(vector of integers) transition_group_ids for which features are to be extracted.
runs	(A vector of string) Names of mzml file without extension.
dataPath	(char) Path to mzml and osw directory.
refRun	(string) reference for alignment. If no run is provided, m-score is used to select reference run.
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_Metabolomics".
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) only analytes that have m-score less than this, will be included in the output.
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 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 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	(character) must be selected from "mean", "l2".
simMeasure	(string) must be selected from dotProduct, cosineAngle, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation.

alignType	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.
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.
hardConstrain	(logical) if FALSE; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	(numeric) modulates penalization of masked indices.
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")
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)
```

---

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).
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 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
```

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(mzml).

Usage

```
getChromatogramIndices(fileInfo, precursors, mzPntrs)
```

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.

Value

(list) A list of dataframes 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

chromatogramIdAsInteger, mapPrecursorToChromIndices

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
precursors <- getPrecursors(fileInfo, oswMerged = TRUE)
mzPntrs <- getMZMLpointers(fileInfo)
prec2chromIndex <- getChromatogramIndices(fileInfo, precursors, mzPntrs)
rm(mzPntrs)
```

`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
)
```

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). 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.

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, runType = "DIA_proteomics")
```

Arguments

`fileInfo`	(data-frame) Output of DIAlignR::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) 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
: (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,fetchPrecursorsInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
## Not run:
features <- getFeatures(fileInfo, maxFdrQuery = 1.00, runType = "DIA_proteomics")
dim(features[[2]]) # 227  7

## End(Not run)
```

---

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, B1p, B2p, noBeef = 50L, hardConstrain = FALSE)
```

Arguments

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.
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.
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(3353.2, 3356.6, 3360.0, 3363.5)
tB <- c(3325.9, 3329.3, 3332.7, 3336.1)
B1p <- 3325.751; B2p <- 3336.119
noBeef <- 1
mask <- getGlobalAlignMaskCpp(tA, tB, B1p, B2p, noBeef, FALSE)
round(mask, 3)
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, 0.000, 0.000), 4, 4, byrow = FALSE)
```

---

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

See Also

getFeatures

Examples

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

---

getMultipeptide *Get multipeptides*

Description

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

Usage

```
getMultipeptide(precursors, features)
```

Arguments

precursors	(data-frames) Contains precursors and associated transition IDs.
features	(list of data-frames) Contains features and their properties identified in each run.

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

License: (c) Author (2020) + GPL-3 Date: 2020-04-08

See Also

getPrecursors,getFeatures

Examples

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

---

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

Examples

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

---

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	(string) This must be one of the strings "DIA_proteomics", "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)
```

---

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")
```

Arguments

fileInfo	(data-frame) Output of DIAlignR::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_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

See Also

getRunNames, fetchPrecursorsInfo

Examples

```
dataPath <- system.file("extdata", package = "DIAlignR")
fileInfo <- DIAlignR::getRunNames(dataPath = dataPath)
## Not run:
precursorsInfo <- getPrecursors(fileInfo, oswMerged = TRUE, runType = "DIA_proteomics")
dim(precursorsInfo) # 322  6

## End(Not run)
```

---

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)
```

Arguments

dataPath	(char) Path to mzml and osw directory.
oswMerged	(logical) TRUE for experiment-wide FDR and FALSE for run-specific FDR by pyprophet.

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 (2019) + GPL-3 Date: 2019-12-14

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 License: (c) Author (2019) + MIT 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)
```

`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
)
```

Arguments

`analytes`	(integer) a vector of precursor IDs.
`runs`	(vector of string) names of mzML files without extension.
`dataPath`	(string) Path to mzml 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.

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)
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)
```

`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)
```

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.

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.

`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

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 mzml file without extension.
`dataPath`	(string) path to mzml 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

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")
```

---

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

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")
```

`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[["run0"]][["14299_QFNNTDIVLLEDFQK/3"]][[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 = NULL,
  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

License: (c) Author (2020) + GPL3 Date: 2020-02-21

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[["run0"]][["14299_QFNNTDIVLLEDFQK/3"]]
## Not run:
newXICs <- smoothXICs(XICs, type = "sgolay", samplingTime = 3.42, kernelLen = 9,
 polyOrd = 3)

## 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

License: (c) Author (2020) + GPL3 Date: 2020-04-01

Examples

```
data("XIC_QFNNTDIVLLEDFQK_3_DIAlignR")
XICs <- XIC_QFNNTDIVLLEDFQK_3_DIAlignR[["run0"]][["14299_QFNNTDIVLLEDFQK/3"]]
## Not run:
newXICs <- smoothXICs(XICs, len = 0.5)

## 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

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)
```

`XIC_QFNNTDIVLLEDFQK_3_DIAlignR` *Extracted-ion chromatograms (XICs) of a peptide*

Description

XICs of peptide QFNNTDIVLLEDFQK/3 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