Package 'DIAlignR'

July 6, 2019

Type Package

Title Dynamic programming based alignment of MS2 chromatograms

Version 1.0.0

Author Shubham Gupta <shubh.gupta@utoronto.ca>, Hannes Rost <hannes.rost@utoronto.ca>

Maintainer Shubham Gupta <shubh.gupta@utoronto.ca>

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.

License BSD

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports methods, zoo (>= 1.8-3), Rcpp

Suggests knitr, rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

SystemRequirements C++11

R topics documented:

AffineAlignObj-class 2
alignChromatogramsCpp 2
AlignObj-class 4
constrainSimCpp 4
DIAlignR 5
doAffineAlignmentCpp 5
doAlignmentCpp 6
getAlignedObj 7
getBaseGapPenaltyCpp 8
getChromSimMatCpp 9

getGlobalAlignMaskCpp 11
getLinearfit 12
getLOESSfit 12
getPepPeakCorp 13
getSeqSimMatCpp 14
mapIdxToTime 15

Index **16**

`AffineAlignObj-class` *An S4 object for class AffineAlignObj*

Description

An S4 object for class AffineAlignObj

`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)
```

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.

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

```
simMeasure <- "dotProductMasked"
run_pair <- c("run1", "run2")
peptide <- peptides[1]
r1 <- lapply(StrepChroms[[run_pair[1]]][[peptide]], `[[`, 2)
r2 <- lapply(StrepChroms[[run_pair[2]]][[peptide]], `[[`, 2)
tRunAVec <- StrepChroms[[run_pair[1]]][[peptide]][[1]][["time"]]
tRunBVec <- StrepChroms[[run_pair[2]]][[peptide]][[1]][["time"]]
```

```
noBeef <- 6
B1p <- predict(Loess.fit, tRunAVec[1]); B2p <- predict(Loess.fit, tRunAVec[length(tRunAVec)])
Alignobj <- alignChromatogramsCpp(r1, r2, "hybrid", tRunAVec, tRunBVec, "mean", simMeasure, B1p, B2p, noBeef)
```

`AlignObj-class` *An S4 object for class AlignObj*

Description

An S4 object for class AlignObj

`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 = F)
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 = F)
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 = F)
```

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 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)
objAffine_Global@score # -2  -4  -6  4 -18
objAffine_Olap <- doAffineAlignmentCpp(s, 22, 7, TRUE)
objAffine_Olap@score # 0 10 20 18 18 18

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

---

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)
obj_Global@score # -2 -4 -6 4 -18
obj_Olap <- doAlignmentCpp(s, 22, TRUE)
obj_Olap@score # 0 10 20 18 18 18
```

`getAlignedObj` *Outputs an aligned object for pairwise correspondence between extracted-ion chromatograms.*

Description

This function calculates three matrices for affine gap alignment using input similarity matrix and affine gap opening and gap closing penalties. An implementation of Needleman-Wunsch alignment and overlap alignment is also provided. All three matrices are clubbed together in an output S4 object.

Usage

```
getAlignedObj(peptide, pairName, XICs.A = NULL, XICs.B = NULL,
  alignType, oswFeatureList = NULL, spanvalue = 0.1,
  normalization = "mean", simMeasure = "dotProductMasked",
  goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3,
  OverlapAlignment = TRUE, dotProdThresh = 0.96, gapQuantile = 0.5,
  hardConstrain = FALSE, samples4gradient = 100, expRSE = 8,
  samplingTime = 3.4, RSEdistFactor = 3.5)
```

Arguments

`peptide`	A string. Peptide for which AlignObj needs to be calculated.
`pairName`	Names of runs joined with underscore. e.g. runA_runB, This will allow alignment of runB to runA.
`XICs.A`	List of extracted ion chromatograms of runA.
`XICs.B`	List of extracted ion chromatograms of runB.
`alignType`	Available alignment methods are "global", "local" and "hybrid".
`oswFeatureList`	List of features detected in each run. For feature detection OpenSWATH, Spectronaut, Skyline etc can be used.
`spanvalue`	A numeric Spanvalue for LOESS fit.
`normalization`	A character string. Must be selected from (mean, l2)
`simMeasure`	A character string. Must be selected from (dotProduct, cosineAngle, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation)

goFactor	A numeric scalar. Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	A numeric scalar. 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	A numeric scalar. Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
hardConstrain	A logical scalar. if false; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	A numeric scalar. This parameter modulates penalization of masked indices.
expRSE	Expected residual standard error(rse) for constraining hybrid alignment, if rse from LOESS fit comes out to be large.
samplingTime	Time difference between two data-points in each chromatogram. For hybrid and local alignment, samples are assumed to be equally time-spaced.
RSEdistFactor	A numeric scalar. This defines how much distance in the unit of rse remains a noBeef zone.

Value

A S4 object

Examples

```
pair <- "run1_run2"
peptide <- "15605_YFMPVHGEYR/3"
hybridObj <- getAlignedObj(peptide, pair, StrepChroms[["run1"]], StrepChroms[["run2"]], "hybrid", oswOutStrep, e
```

---

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

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 = F)
getBaseGapPenaltyCpp(sim, "dotProductMasked", 0.5) # -0.25
```

---

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 License: (c) Author (2019) + MIT 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 = F)

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 = F)

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 = F)

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 = F)

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 = F)

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 = F)

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 = F)
```

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 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 = F)
```

getLinearfit *calculates linear fit between RT of two runs*

Description

This function takes in run-pairs, names of test peptides, output of OpenSWATH which include estimated retention time of peptides.

Usage

```
getLinearfit(pairName, peptides, oswOutput)
```

Arguments

pairName	Names of runs joined with underscore. e.g. runA_runB, This will allow alignment of runB to runA.
peptides	Test peptides' names.
oswOutput	list of list (OpenSWATH output).

Value

An object of class "lm".

getLOESSfit *calculates LOESS fit between RT of two runs*

Description

This function takes in run-pairs, names of test peptides, span paprameter for LOESS function, output of OpenSWATH which include estimated retention time of peptides.

Usage

```
getLOESSfit(pairName, peptides, oswOutput, spanvalue = 0.1)
```

Arguments

pairName	Names of runs joined with underscore. e.g. runA_runB, This will allow alignment of runB to runA.
peptides	Test peptides' names.
oswOutput	list of list (OpenSWATH output).
spanvalue	A numeric Spanvalue for LOESS fit. For targeted proteomics 0.1 could be used.

Value

An object of class "loess".

getPepPeakCorp *Establishes pairwise correspondence between extracted-ion chromatograms.*

Description

This function calculates three matrices for affine gap alignment using input similarity matrix and affine gap opening and gap closing penalties. An implementation of Needleman-Wunsch alignment and overlap alignment is also provided. All three matrices are clubbed together in an S4 object.

Usage

```
getPepPeakCorp(featureTable, pairName, XICs.A = NULL, XICs.B = NULL,
  alignType, oswFeatureList = NULL, spanvalue = 0.1,
  normalization = "mean", simMeasure = "dotProductMasked",
  goFactor = 0.125, geFactor = 40, cosAngleThresh = 0.3,
  OverlapAlignment = TRUE, dotProdThresh = 0.96, gapQuantile = 0.5,
  hardConstrain = FALSE, samples4gradient = 100, expRSE = 8,
  samplingTime = 3.4, RSEdistFactor = 3.5)
```

Arguments

featureTable	A table of retention time identifications for each peptides. The retention times could be annotated manually or through softwares like OpenSWATH.
pairName	Names of runs joined with underscore. e.g. runA_runB, This will allow alignment of runB to runA.
XICs.A	List of extracted ion chromatograms of runA.
XICs.B	List of extracted ion chromatograms of runB.
alignType	Available alignment methods are "global", "local" and "hybrid".
oswFeatureList	List of features detected in each run. For feature detection OpenSWATH, Spectronaut, Skyline etc can be used.
spanvalue	A numeric Spanvalue for LOESS fit.
normalization	A character string. Must be selected from (mean, l2)
simMeasure	A character string. Must be selected from (dotProduct, cosineAngle, cosine2Angle, dotProductMasked, euclideanDist, covariance and correlation)
goFactor	A numeric scalar. Penalty for introducing first gap in alignment. This value is multiplied by base gap-penalty.
geFactor	A numeric scalar. 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	A numeric scalar. Must be between 0 and 1. This is used to calculate base gap-penalty from similarity distribution.
hardConstrain	A logical scalar. if false; indices farther from noBeef distance are filled with distance from linear fit line.
samples4gradient	A numeric scalar. This parameter modulates penalization of masked indices.
expRSE	Expected residual standard error(rse) for constraining hybrid alignment, if rse from LOESS fit comes out to be large.
samplingTime	Time difference between two data-points in each chromatogram. For hybrid and local alignment, samples are assumed to be equally time-spaced.
RSEdistFactor	A numeric scalar. This defines how much distance in the unit of rse remains a noBeef zone.

Value

A numeric matrix

Examples

```
pair <- "run1_run2"
MappedTime2 <- getPepPeakCorp(StrepAnnot, pair, StrepChroms[["run1"]], StrepChroms[["run2"]],
"hybrid", oswOutStrep, expRSE = 7.4)
```

---

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 = F)
```

---

mapIdxToTime *Establishes mapping back 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`. For `NA` indices it uses last index to fill time value. For `NA` appearing at the start of `idx`, it uses next index to get time value.

Usage

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

Arguments

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

Value

A mutated time vector

Index

AffineAlignObj *(AffineAlignObj-class)*, 2
AffineAlignObj-class, 2
alignChromatogramsCpp, 2
AlignObj *(AlignObj-class)*, 4
AlignObj-class, 4

constrainSimCpp, 4

DIAlignR, 5
DIAlignR-package *(DIAlignR)*, 5
doAffineAlignmentCpp, 5
doAlignmentCpp, 6

getAlignedObj, 7
getBaseGapPenaltyCpp, 8
getChromSimMatCpp, 9
getGlobalAlignMaskCpp, 11
getLinearfit, 12
getLOESSfit, 12
getPepPeakCorp, 13
getSeqSimMatCpp, 14

mapIdxToTime, 15