Package 'prozor'

April 20, 2017

Type Package

Title Minimal Protein Set Explaining Peptide Spectrum Matches

Version 0.2.3

Author Witold Wolski

Maintainer Witold Wolski <wewolski@gmail.com>

Description Determine minimal protein set explaining
peptide spectrum matches. Utility functions for creating libraries with decoys.
Planned is peptide FDR estimation for search results.

License GPL-3

LazyData TRUE

Imports AhoCorasickTrie, Matrix, doParallel, foreach, plyr, seqinr,
stringr

URL https://github.com/protviz/prozor

BugReports https://github.com/protviz/prozor/issues

Repository CRAN

RoxygenNote 6.0.1

NeedsCompilation no

Date/Publication 2017-04-20 11:43:20 UTC

R topics documented:

`annotateAHO` *annotate peptides using AhoCorasickTrie*

Description

peptides which do not have protein assignment drop out

Usage

```
annotateAHO(pepseq, fasta)
```

Arguments

`pepseq`	- list of peptides - sequence, optional modified sequence, charge state.
`fasta`	- object as created by readPeptideFasta

Examples

```
library(prozor)
file = file.path(path.package("prozor"),"extdata/shortfasta.fasta" )
fasta = readPeptideFasta(file = file)
#res = annotateVec(pepprot[1:20,"peptideSeq"],fasta)
system.time(res2 <- annotateAHO(pepprot[1:20,"peptideSeq"],fasta))
colnames(res2)
```

annotatePeptides *Annotate peptides with protein ids*

Description

peptides which do not have protein assignment drop out

Usage

```
annotatePeptides(pepinfo, fasta, prefix = "(([RK])|(^)|(^M))", suffix = "")
```

Arguments

pepinfo	- list of peptides - sequence, optional modified sequence, charge state.
fasta	- object as created by readPeptideFasta
prefix	- default "(([RK])|(^)|(^M))"
suffix	- default ""

Examples

```
library(prozor)
data(pepprot)

file = file.path(path.package("prozor"),"extdata/shortfasta.fasta" )

fasta = readPeptideFasta(file = file)
res = annotatePeptides(pepprot[1:20,], fasta)
head(res)
res = annotatePeptides(pepprot[1:20,"peptideSeq"],fasta)
length(res)
```

annotateVec *annotate vector of petpide sequences against fasta file (Deprecated)*

Description

annotate vector of petpide sequences against fasta file (Deprecated)

Usage

```
annotateVec(pepseq, fasta, digestPattern = "(([RK])|(^)|(^M))",
  mcCores = NULL)
```

Arguments

pepseq	peptide sequences
fasta	fasta file
digestPattern	digest pattern as regex
mcCores	nr of cores to use

Examples

```
library(prozor)
file = file.path(path.package("prozor"),"extdata/shortfasta.fasta" )
fasta = readPeptideFasta(file = file)

res = annotateVec(pepprot[1:20,"peptideSeq"],fasta)
head(res)
```

---

createDecoyDB *create db with decoys and contaminants*

Description

create db with decoys and contaminants

Usage

```
createDecoyDB(dbs, useContaminants = TRUE, revLab = "REV_",
  annot = "zz|sourceOf|database")
```

Arguments

dbs	a path to a fasta file or an array of files
useContaminants	add fgcz contaminants
revLab	label for reversed peptides (if NULL do not generate decoys)
annot	source of database

Examples

```
file = file.path(path.package("prozor"),"extdata/uniprot_taxonomy_Oryctolagus_cuniculus.fasta.gz")
cont <- loadContaminantsFasta()
rabbit <-readPeptideFasta(file)
tmp <- 2*(2*length(rabbit)+length(cont)) + 1

res <- createDecoyDB(c(file,file))
length(res)
tmp
stopifnot(length(res) == tmp)
```

```
res <- createDecoyDB(c(file,file), revLab=NULL)
stopifnot(length(res) == (2*length(rabbit)+length(cont) + 1))
res <- createDecoyDB(c(file,file), revLab=NULL, useContaminants = FALSE)
stopifnot(length(res) == (2*length(rabbit) + 1) )
```

`filterSequences` *Filter for specific residues*

Description

Will check if AA at Offset is a valid cleavage site

Usage

```
filterSequences(matches, prefix = "(([RK])|(^)|(^M))", suffix = "")
```

Arguments

`matches`	must have 2 columns proteinSequnce and Offset
`prefix`	- regular expression describing the prefix of the peptide sequence e.g. (([RK])|(^)|(^M))
`suffix`	- regular expression describing the suffix of the peptide sequence

---

`greedy` *given matrix (columns protein rows peptides), compute minimal protein set using greedy algorithm*

Description

given matrix (columns protein rows peptides), compute minimal protein set using greedy algorithm

Usage

```
greedy(pepprot)
```

Arguments

`pepprot`	matrix as returned by prepareMatrix

Value

list of peptide protein assignment

Examples

```
library(prozor)

data(protpepmetashort)
colnames(protpepmetashort)
dim(unique(protpepmetashort[,4:5]))
xx = prepareMatrix(protpepmetashort, weight= "one")
dim(xx)
stopifnot(dim(xx)[1] == dim(unique(protpepmetashort[,4:5]))[1])
es = greedy(as.matrix(xx))
stopifnot(length(unique(names(es))) == dim(unique(protpepmetashort[,4:5]))[1])
```

---

greedyRes2Matrix *converts result of greedy function to a matrix with 3 columns - peptide - charge and protein*

Description

converts result of greedy function to a matrix with 3 columns - peptide - charge and protein

Usage

```
greedyRes2Matrix(res)
```

Arguments

res	result of function prozor::greedy

Value

matrix of peptide protein assignments

---

loadContaminantsFasta *load list of contaminant sequences*

Description

load list of contaminant sequences

Usage

```
loadContaminantsFasta()
```

Examples

```
library(prozor)
cont <- loadContaminantsFasta()
cont[[1]]
#example how to create a protein db with decoy sequences
```

`loadContaminantsNoHumanFasta`
load list of contaminant without human sequences

Description

load list of contaminant without human sequences

Usage

```
loadContaminantsNoHumanFasta()
```

Examples

```
library(prozor)
cont <- loadContaminantsNoHumanFasta()
cont[[1]]
#example how to create a protein db with decoy sequences
```

`loadHomoSapiensSignalPeptides`
load human signal peptides

Description

load human signal peptides

Usage

```
loadHomoSapiensSignalPeptides()
```

Examples

```
library(prozor)
signal <- loadHomoSapiensSignalPeptides()
```

loadMusMusculusSignalPeptides *load mus musculus signal peptides*

Description

load mus musculus signal peptides

Usage

```
loadMusMusculusSignalPeptides()
```

Examples

```
library(prozor)
signal <- loadMusMusculusSignalPeptides()
head(signal)
```

makeID *make id for chain in format sp|P30443|1A01_HUMANs25*

Description

make id for chain in format sp|P30443|1A01_HUMANs25

Usage

```
makeID(sequence, id, sp)
```

Arguments

sequence	- aa sequence as string
id	uniprot id id: sp|P30443|1A01_HUMAN
sp	start position of chain numeric

Examples

```
seq <- "MAVMAPRTLLLLLSGALALTQTWAGSHSMRYFFTSVSRPGR\
GEPRFIAVGYVDDTQFVRFDSDAASQKMEPRAPWIEQEGPEYWDQETRN\
MKAHSQTDRANLGTLRGYYNQSEDGSHTIQIMYGCDVGPDGRFLRGYRQ\
DAYDGKDYIALNEDLRSWTAADMAAQITKRKWEAVHAAEQRRVYLEGRC\
VDGLRRYLENGKETLQRTDPPKTHMTHHPISDHEATLRCWALGFYPAEI\
TLTWQRDGEDQTQDTELVETRPAGDGTFQKWAAVVVPSGEEQRYTCHVQ\
HEGLPKPLTLRWELSSQPTIPIVGIIAGLVLLGAVITGAVVAAVMWRRK\
SSDRKGGSYTQAASSDSAQGSDVSLTACKV"
nam <-"sp|P30443|1A01_HUMAN"
sp <- 24
makeID(seq, nam, sp)
```

makeIDUnip *make id for chain compatible with uniprot*

Description

make id for chain compatible with uniprot

Usage

```
makeIDUnip(sequence, id, sp)
```

Arguments

sequence	- aa sequence as string
id	uniprot id id: sp|P30443|1A01_HUMAN
sp	start position of chain numeric

Examples

```
seq <- "MAVMAPRTLLLLLSGALALTQTWAGSHSMRYFFTSVSRPGR\
GEPRFIAVGYVDDTQFVRFDSDAASQKMEPRAPWIEQEGPEYWDQETRN\
MKAHSQTDRANLGTLRGYYNQSEDGSHTIQIMYGCDVGPDGRFLRGYRQ\
DAYDGKDYIALNEDLRSWTAADMAAQITKRKWEAVHAAEQRRVYLEGRC\
VDGLRRYLENGKETLQRTDPPKTHMTHHPISDHEATLRCWALGFYPAEI\
TLTWQRDGEDQTQDTELVETRPAGDGTFQKWAAVVVPSGEEQRYTCHVQ\
HEGLPKPLTLRWELSSQPTIPIVGIIAGLVLLGAVITGAVVAAVMWRRK\
SSDRKGGSYTQAASSDSAQGSDVSLTACKV"
nam <-"sp|P30443|1A01_HUMAN"
sp <- 24
makeIDUnip(seq, nam, sp)
```

pepprot *Table containing peptide information*

Description

Table containing peptide information

prepareMatrix *given table of peptide protein assigments generate matrix*

Description

given table of peptide protein assigments generate matrix

Usage

```
prepareMatrix(data, weighting = "one", sep = "|")
```

Arguments

data	generated by annotatePeptides
weighting	weight type to use. Options are "one" , "AA" - amino acids, "coverage" - coverage , "inverse" - inverse peptide frequencies
sep	separator for precursor (rownames)

Value

sparse matrix

Examples

```
library(prozor)
data(protpepmetashort)
library(Matrix)
colnames(protpepmetashort)
head(protpepmetashort)
dim(protpepmetashort)
count = prepareMatrix( protpepmetashort)
inverse = prepareMatrix( protpepmetashort, weight = "inverse")
aa = prepareMatrix(protpepmetashort, weight = "AA")
#xx = prepareMatrix(protpepmetashort, weight = "coverage")
image( as.matrix(count) )

corProt = cor( as.matrix(count) )
par(mfrow =c(1,2))
image(corProt)

#penalise peptides matching many proteins
corProtn = cor( as.matrix(aa) )
image(corProtn)
```

protpepmeta *Generated from pepprot using method annotatePeptides*

Description

Generated from pepprot using method annotatePeptides

protpepmetashort *Small version of pepprot dataset to speed up computation*

Description

Small version of pepprot dataset to speed up computation

prozor *Minimal Protein set Explaining Peptides*

Description

Minimal Protein set Explaining Peptides

readPeptideFasta *wrapper setting the correct parameters*

Description

peptides which do not have protein assignment drop out

Usage

```
readPeptideFasta(file)
```

Arguments

file	- fasta file

Examples

```
library(seqinr)
library(prozor)
file = file.path(path.package("prozor"),"extdata/fgcz_contaminants_20150123.fasta")
fasta = readPeptideFasta(file)
```

removeSignalPeptide *remove signal peptides from main chain*

Description

remove signal peptides from main chain

Usage

```
removeSignalPeptide(db, signal, idfun = makeID)
```

Arguments

db	uniprot fasta database as list
signal	tab delimited file with signals
idfun	function to generate id's

Examples

```
## Not run:
library(prozor)

hsfasta <- loadHomoSapiensFasta()
hssignal <- loadHomoSapiensSignalPeptides()
xx <- removeSignalPeptide(hsfasta, hssignal)
db <- hsfasta
signal <- hssignal

## End(Not run)
```

reverseSeq *create rev sequences to fasta list*

Description

peptides which do not have protein assignment drop out

Usage

```
reverseSeq(fasta, revLab = "REV_")
```

Arguments

fasta	- an r list with SeqFastaAA
revLab	- how to label reverse sequences, default = REV_

Examples

```
library(seqinr)
library(prozor)

file = file.path(path.package("prozor"),"extdata/fgcz_contaminants_20150123.fasta")
file
fasta = readPeptideFasta(file = file)
x <- reverseSeq(fasta)


revseq <- reverseSeq(fasta ,revLab = "REV_")
stopifnot(length(revseq) == length(fasta))
stopifnot(grep("^REV_","REV_zz|ZZ_FGCZCont0000|")==1)

tmp <- list(as.SeqFastaAA(("DYKDDDDK"),name="Flag|FLAG|p2079",Annot=""))

reverseSeq(tmp)
```

`writeFasta` *write fasta lists into file*

Description

peptides which do not have protein assignment drop out

Usage

```
writeFasta(file, ...)
```

Arguments

`file`	where to write
`...`	fasta list or single file

Examples

```
#example how to create a protein db with decoy sequences
library(seqinr)
library(prozor)
file = file.path(path.package("prozor"),"extdata/fgcz_contaminants_20150123.fasta")
fasta = readPeptideFasta(file = file)
revfasta <- reverseSeq(fasta)
decoyDB <- c(fasta,revfasta)
stopifnot(length(decoyDB) == 2 * length(fasta))
writeFasta(decoyDB, file="test.fasta")
```

Index