

# Package ‘prozor’

May 9, 2026

**Type** Package

**Title** Minimal Protein Set Explaining Peptide Spectrum Matches

**Version** 0.3.1

**Description** Determine minimal protein set explaining peptide spectrum matches. Utility functions for creating fasta amino acid databases with decoys and contaminants. Peptide false discovery rate estimation for target decoy search results on psm, precursor, peptide and protein level. Computing dynamic swath window sizes based on MS1 or MS2 signal distributions.

**License** GPL-3

**Imports** AhoCorasickTrie, docopt, Matrix, methods, purrr, readr, rlang, seqinr, stringr, dplyr

**Suggests** knitr, rmarkdown,

**URL** <https://github.com/protviz/prozor>

**BugReports** <https://github.com/protviz/prozor/issues>

**Repository** CRAN

**RoxygenNote** 7.1.2

**Depends** R (>= 3.1.0)

**VignetteBuilder** knitr

**Collate** 'annotatePeptides.R' 'readjustWindow.R' 'cdsw.R'  
'computeFDR.R' 'createDecoyDB.R' 'create\_fgcz\_fasta\_db.R'  
'greedy.R' 'hello.R' 'loadContaminantsFasta.R'  
'prepareMatrix.R' 'readFasta.R' 'removeSignalPeptides.R'  
'reverseSeq.R' 'writeFasta.R' 'zzz.R'

**biocViews** Software, MassSpectrometry, Proteomics,  
ExperimentHubSoftware,

**NeedsCompilation** no

**Author** Witold Wolski [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-6468-120X>>)

**Maintainer** Witold Wolski <[wewolski@gmail.com](mailto:wewolski@gmail.com)>

**Date/Publication** 2021-12-07 16:20:02 UTC

## Contents

annotateAHO . . . . .	2
annotatePeptides . . . . .	3
Cdsw-class . . . . .	4
computeFDR . . . . .	6
computeFDRwithID . . . . .	6
createDecoyDB . . . . .	7
create_fgcz_fasta_db . . . . .	8
fdrSample . . . . .	9
greedy . . . . .	9
greedyRes2Matrix . . . . .	10
loadContaminantsFasta2019 . . . . .	11
loadContaminantsFasta2021 . . . . .	11
makeID . . . . .	12
makeIDUnip . . . . .	13
masses . . . . .	13
objectiveMS1Function . . . . .	14
pepprot . . . . .	14
plotFDR . . . . .	14
predictScoreFDR . . . . .	15
prepareMatrix . . . . .	16
protpepmetashort . . . . .	17
prozor . . . . .	17
readjustWindows . . . . .	17
readPeptideFasta . . . . .	18
removeSignalPeptide . . . . .	19
reverseSeq . . . . .	19
writeFasta . . . . .	20
<b>Index</b>	<b>22</b>

---

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

**Value**

A data.frame with proteinID, peptideSeq, Offset and proteinSequence

**Examples**

```
library(dplyr)

file = system.file("extdata/IDResults.txt.gz" , package = "prozor")
specMeta <- readr::read_tsv(file)
upeptide <- unique(specMeta$peptideSeq)
resCan <-
  prozor::readPeptideFasta(
    system.file("p1000_db1_example/Annotation_canSeq.fasta.gz" , package = "prozor"))
resCanU <- resCan[!duplicated(unlist(resCan))]
annotAll = annotateAHO(upeptide[seq_len(20)], resCanU)
dim(annotAll)
```

---

annotatePeptides      *Annotate peptides with protein ids*

---

**Description**

peptides which do not have protein assignment drop out

**Usage**

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

**Arguments**

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

**Value**

data.frame with columns "peptideSeq", "proteinID", "Offset", "proteinSequence", "matched", "length-Peptide", "proteinlength"

**Examples**

```

library(dplyr)

file = system.file("extdata/IDResults.txt.gz" , package = "prozor")
specMeta <- readr::read_tsv(file)
upeptide <- unique(specMeta$peptideSeq)
resCan <-
  prozor::readPeptideFasta(
    system.file("p1000_db1_example/Annotation_canSeq.fasta.gz" , package = "prozor"))

annotAll = prozor::annotatePeptides(upeptide[seq_len(20)], resCan)
dim(annotAll)

res <- mutate(annotAll, proteinlength = nchar(proteinSequence))
res <- select(res, proteinID, peptideSeq, proteinlength, Offset, lengthPeptide)
head(res)

```

CdsW-class

*Compute dynamic swath windows***Description**

initialize  
 create equidistant breaks  
 quantile breaks  
 sampling breaks  
 barplot showing the number of precursors per window  
 Table with window boundaries and statistics  
 summary of the binning process (see objectiveMS1Function for more details)  
 moves window start and end to region with as few as possible precursor masses  
 shows the generated DIA cycle

**Arguments**

list	of masses
nbins	number of bins default 25
maxwindow	largest window size
minwindow	smallest window size
digits	mass precision default 2
digigits	mass precision
max	number of bins
plot	default TRUE
overlap	size of window overlap default 1 m/z

**Value**

array of masses

array with masses

array with masses

data.frame with columns: - from (window start) - to (window end) - mid (window centre), width (window width) - counts expected number of precursors

list with optimization scores

data.frame with optimized windows

**Fields**

masses MS1 masses

breaks the breaks

nbins number of bins

digits mass accuracy in result

**Methods**

asTable(overlap = 1) make windows

error() show error

optimizeWindows(digits = 1, maxbin = 15, plot = FALSE, overlap = 0) optimizes the windows

quantile\_breaks(digits = 2) same number of MS1 in each window but might violate hard constraints

sampling\_breaks(maxwindow = 150, minwindow = 5, digits = 2, plot = FALSE) starts with quantile breaks but mixes with uniform data to satisfy hard constraints

**Examples**

```
data(masses)
cdsw <- CdsW(masses)
tmp <- cdsw$sampling_breaks(maxwindow=100,plot=TRUE)
cdsw$plot()
cdsw$asTable()
cdsw$breaks
cdsw$optimizeWindows()
cdsw$showCycle()
```

---

computeFDR	<i>Compute FDR given a score</i>
------------	----------------------------------

---

**Description**

Same as computeFDRwithID but works with decoy\_hit boolean vector. For more details and references see package vignette vignette("TargetDecoyFDR\_Example", package = "prozor")

**Usage**

```
computeFDR(score, decoy_hit, larger_better = TRUE)
```

**Arguments**

score	score
decoy_hit	indicates if decoy hit
larger_better	is larger score the better one (default TRUE)

**Value**

list with decoy\_hit (indicates if decoy), score the search engine score, FDR1 false discovery rate estimated using the method of Gygi, SimpleFDR - estimated using the method of Kaell.

**Examples**

```
data(fdrSample)

fdr1 <- computeFDR(fdrSample$score, grepl("REV_", fdrSample$proteinID), larger_better = FALSE)
head(as.data.frame(fdr1))
```

---

computeFDRwithID	<i>Compute FDR given a score</i>
------------------	----------------------------------

---

**Description**

For more details and references see package vignette vignette("TargetDecoyFDR\_Example", package = "prozor")

**Usage**

```
computeFDRwithID(score, ID, decoy = "REV_", larger_better = TRUE)
```

**Arguments**

score	a vector with scores
ID	- list with protein id's
decoy	decoy pattern, default "REV_"
larger_better	if larger score better than small (default TRUE), If small score better set FALSE

**Value**

list with ID, decoy\_hit (indicates if decoy), score the search engine score, FDR1 false discovery rate estimated using the method of Elias and Gygi; FDR2 - estimated using the method of Kell.

**Examples**

```
data(fdrSample)
# call constructor
#nrow(fdrSample)
#fdrSample <- dplyr::slice_sample(fdrSample, n = 40000)

#usethis::use_data(fdrSample, overwrite = TRUE)
fdr1 <- computeFDRwithID(fdrSample$score, fdrSample$proteinID, larger_better = FALSE)
names(fdr1)
plot(fdr1$score, fdr1$FPR, type="l", xlim=c(0,0.001), ylim=c(0,0.0002))
lines(fdr1$score, fdr1$qValue_FPR, col=2)
lines(fdr1$score, fdr1$SimpleFDR, type="l", col=4)
lines(fdr1$score, fdr1$qValue_SimpleFDR, col=5)

fdr1 <- computeFDRwithID(fdrSample$score2, fdrSample$proteinID, larger_better = TRUE)
names(fdr1)
plot(fdr1$score, fdr1$FPR, type="l", xlim=c(2.5,5), ylim=c(0,0.001))
lines(fdr1$score, fdr1$qValue_FPR, col=2)
lines(fdr1$score, fdr1$SimpleFDR, type="l", col=4)
lines(fdr1$score, fdr1$qValue_SimpleFDR, col=5)
```

---

createDecoyDB

---

*Create db with decoys and contaminants*


---

**Description**

For more details and references see package vignette `vignette("CreateDecoyDB", package = "prozor")`

**Usage**

```
createDecoyDB(  
  dbs,  
  useContaminants = loadContaminantsFasta2021(),  
  revLab = "REV_",  
  annot = "zz|sourceOf|database"  
)
```

**Arguments**

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

**Value**

list of SeqFastaAA entries

**Examples**

```
file = system.file("extdata/fgcz_contaminants2021_20210929.fasta.gz", package="prozor")  
cont <- loadContaminantsFasta2021()  
rabbit <- readPeptideFasta(file)  
tmp <- 2*(2*length(rabbit)+length(cont)) + 1  
  
res <- createDecoyDB(c(file,file))  
length(res)  
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 = NULL)  
stopifnot(length(res) == (2*length(rabbit) + 1) )
```

---

create\_fgcz\_fasta\_db *create fasta db from one or more fasta files*

---

**Description**

create fasta db from one or more fasta files

**Usage**

```
create_fgcz_fasta_db(
  databasedirectory,
  useContaminants = loadContaminantsFasta2021(),
  revLab = "REV_",
  outputdir = NULL
)
```

**Arguments**

```
databasedirectory      directory with fasta files
useContaminants        contaminants to add
revLab                  reverse label
outputdir               output directory
```

**Value**

```
list list(resDB, filepath , summary, mcall, dbname)
```

**Examples**

```
print("NO exmple since function also writes the fasta files")
```

---

fdrSample	<i>Data frame score and proteinID</i>
-----------	---------------------------------------

---

**Description**

Data frame score and proteinID

---

greedy	<i>given matrix (columns protein rows peptides), compute minimal protein set using greedy algorithm</i>
--------	---

---

**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]))
xx = prepareMatrix(protpepmetashort, peptideID = "peptideModSeq")
dim(xx)
stopifnot(dim(xx)[1] == dim(unique(protpepmetashort[,4]))[1])
es = greedy(xx)
stopifnot(length(unique(names(es))) == dim(unique(protpepmetashort[,4]))[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

---

```
loadContaminantsFasta2019
  load list of contaminant sequences FGCZ 2019
```

---

**Description**

load list of contaminant sequences FGCZ 2019

**Usage**

```
loadContaminantsFasta2019(noHuman = FALSE)
```

**Arguments**

noHuman            should human contaminants be excluded? default FALSE

**Value**

list with contaminant sequences

**Examples**

```
#library(prozor)
cont <- loadContaminantsFasta2019()
length(cont)
contNH <- loadContaminantsFasta2019()
length(contNH)
#example how to create a protein db with decoy sequences
```

---

```
loadContaminantsFasta2021
  load list of contaminant sequences FGCZ 2021
```

---

**Description**

load list of contaminant sequences FGCZ 2021

**Usage**

```
loadContaminantsFasta2021(noHuman = FALSE)
```

**Arguments**

noHuman            should human contaminants be excluded? default FALSE

**Value**

list with contaminant sequences

**Examples**

```
#library(prozor)
cont <- loadContaminantsFasta2021()
length(cont)
contNH <- loadContaminantsFasta2021()
length(contNH)
#example how to create a protein db with decoy sequences
```

---

makeID	<i>make id for chain in format sp P30443 1A01_HUMANS25</i>
--------	--

---

**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

**Value**

string consisting of id,"s",sp

**Examples**

```
seq <- "MAVMAPRTL L L L L L S G A L A L T Q T W A G S H S M R Y F F T S V S R P G R \
G E P R F I A V G Y V D D T Q F V R F D S D A A S Q K M E P R A P W I E Q E G P E Y W D Q E T R N \
M K A H S Q T D R A N L G T L R G Y Y N Q S E D G S H T I Q I M Y G C D V G P D G R F L R G Y R Q \
D A Y D G K D Y I A L N E D L R S W T A A D M A A Q I T K R K W E A V H A A E Q R R V Y L E G R C \
V D G L R R Y L E N G K E T L Q R T D P P K T H M T H H P I S D H E A T L R C W A L G F Y P A E I \
T L T W Q R D G E D Q T Q D T E L V E T R P A G D G T F Q K W A A V V V P S G E E Q R Y T C H V Q \
H E G L P K P L T L R W E L S S Q P T I P I V G I I A G L V L L G A V I T G A V V A A V M W R R K \
S S D R K G G S Y T Q A A S S D S A Q G S D V S L T A C K V"
nam <- "sp|P30443|1A01_HUMAN"
sp <- 24
makeID(seq, nam, sp)
```

---

makeIDUnip	<i>make id for chain compatible with uniprot</i>
------------	--

---

**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

**Value**

string consisting of sp,"-", length of sequece

**Examples**

```
seq <- "MAVMAPRTLLLLLSGALALTQTWAGSHSMRYFFTSVSRPGR\
GEPRFIAVGYVDDTQFVRFSDAASQKMEPRAPWIEQEGPEYWDQETRN\
MKAHSQTDLANLGLRGYYNQSEDSHTIQIMYGCDVGPDRFLRGYRQ\
DAYDGKDYIALNEDLRSWTAADMAAQITKRKWEAVHAAEQRRVYLEGRC\
VDGLRRYLENGKETLQRTDPPKTHMTHHPISDHEATLRCWALGFYPAEI\
TLTWQRDGEDQTQDELVETRPAGDGTQKWAAVVVPSGEEQRYTCHVQ\
HEGLPKPLTLRWELSSQPTIPVIGIIAGLVLLGAVITGAVVAVMWRRK\
SSDRKGGSYTQAASSDSAQGSVDVSLTACKV"
nam <- "sp|P30443|1A01_HUMAN"
sp <- 24
makeIDUnip(seq, nam, sp)
```

---

masses	<i>MS masses A dataset containing approx 150000 MS1 precursor masses</i>
--------	--

---

**Description**

MS masses A dataset containing approx 150000 MS1 precursor masses

---

objectiveMS1Function    *compute the deviation from optimum: equal number of MS1 per bin*

---

**Description**

compute the deviation from optimum: equal number of MS1 per bin

**Usage**

objectiveMS1Function(splits, data)

**Arguments**

splits            the new window boundaries  
 data             the data

**Value**

list with score1 - manhattan distance, score2 - euclidean distance, counts - observed counts, optimumN - optimum counts

---

pepprot            *Table containing peptide information*

---

**Description**

Table containing peptide information

---

plotFDR            *plot FDR*

---

**Description**

For more details and references see package vignette vignette("TargetDecoyFDR\_Example", package = "prozor")

**Usage**

plotFDR(data)

**Arguments**

data             data returned by computeFDR function

**Value**

creates a plot

**Examples**

```
#library(prozor)
data(fdrSample)
fdr1 <- computeFDRwithID(fdrSample$score, fdrSample$proteinID, larger_better = FALSE)
fdr2 <- computeFDRwithID(fdrSample$score2, fdrSample$proteinID, larger_better = TRUE)
plotFDR(fdr1)
plotFDR(fdr2)
data<-fdr1
```

---

predictScoreFDR	<i>Predict score given FDR</i>
-----------------	--------------------------------

---

**Description**

For more details and references see package vignette vignette("TargetDecoyFDR\_Example", package = "prozor")

**Usage**

```
predictScoreFDR(fdrObj, qValue = 1, method = "SimpleFDR")
```

**Arguments**

fdrObj	object generated by computeFDR
qValue	false discovery rate in percent, default 1 percent
method	either FPR or SimpleFDR, default is SimpleFDR

**Value**

score for a given FDR

**Examples**

```
data(fdrSample)
fdr1 <- computeFDRwithID(fdrSample$score, fdrSample$proteinID, larger_better = FALSE)

predictScoreFDR(fdr1,qValue=5)
fdr2<-computeFDRwithID(fdrSample$score2, fdrSample$proteinID, larger_better = TRUE)
predictScoreFDR(fdr2,qValue=5)
```

---

```
prepareMatrix      given table of peptide protein assignments generate matrix
```

---

**Description**

given table of peptide protein assignments generate matrix

**Usage**

```
prepareMatrix(  
  data,  
  proteinID = "proteinID",  
  peptideID = "strippedSequence",  
  weighting = NULL,  
  sep = "|" )
```

**Arguments**

data	generated by annotatePeptides
proteinID	protein ID column
peptideID	peptide / precursor ID column
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, peptideID = "peptideSeq" )  
dim(count)  
inverse = prepareMatrix( protpepmetashort, peptideID = "peptideSeq" , weight = "inverse")  
#aa = prepareMatrix(protpepmetashort, peptideID = "peptideSeq" , weight = "AA")  
#xx = prepareMatrix(protpepmetashort, peptideID = "peptideSeq" , 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(inverse) )
image(corProtn)

```

---

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

Determine minimal protein set explaining peptide spectrum matches. Utility functions for creating fasta amino acid databases with decoys and contaminants. Peptide false discovery rate estimation for target decoy search results on psm, precursor, peptide and protein level. Computing dynamic swath window sizes based on MS1 and MS2 signal distributions.

---

readjustWindows      *Readjust windows so that boundaries in regions of few peaks.*

---

### Description

Readjust windows so that boundaries in regions of few peaks.

### Usage

```
readjustWindows(wind, ms1data, digits = 1, maxbin = 15, plot = FALSE)
```

### Arguments

wind	a data frame with columns from and to
ms1data	masses
digits	mass accuracy
maxbin	maximum number of bins
plot	diagnostic plots (default FALSE)

**Value**

data.frame of same format as wind but with improved start and end masses.

**Examples**

```
data(masses)
cdsw <- CdsW(masses)
breaks <- cdsw$sampling_breaks(maxwindow=100,plot=TRUE)
table <- cdsw$asTable()
dim(table)
head(table)

tmp <- readjustWindows(table, masses,maxbin=10)
data.frame(tmp)
```

---

readPeptideFasta	<i>wrapper setting the correct parameters seqinr::read.fasta for reading peptide sequences</i>
------------------	--

---

**Description**

peptides which do not have protein assignment drop out

**Usage**

```
readPeptideFasta(file)
```

**Arguments**

file           - fasta file

**Value**

list with sequences

**Examples**

```
library(seqinr)

file = system.file("extdata/fgcz_contaminants2021_20210929.fasta.gz",package = "prozor")
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

**Value**

list with sequences with signal peptide removed

---

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_

**Value**

string with reversed sequence

**Examples**

```

library(seqinr)
#library(prozor)

file = system.file("extdata/fgcz_contaminants2021_20210929.fasta.gz", package="prozor")
fasta = readPeptideFasta(file = file)
getAnnot(fasta[[1]])
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

**Value**

writes a file.

**Examples**

```

#example how to create a protein db with decoy sequences
library(seqinr)
#library(prozor)
file = system.file("extdata/fgcz_contaminants2021_20210929.fasta.gz", package = "prozor")
fasta = readPeptideFasta(file = file)
revfasta <- reverseSeq(fasta)
decoyDB <- c(fasta, revfasta)
stopifnot(length(decoyDB) == 2 * length(fasta))
## Not run:

```

```
writeFasta(decoyDB, file="test.fasta")
```

```
## End(Not run)
```

# Index

## \* data

- fdrSample, [9](#)
- masses, [13](#)
- pepprot, [14](#)
- protpepmetashort, [17](#)

annotateAHO, [2](#)  
annotatePeptides, [3](#)

Cdsw (Cdsw-class), [4](#)  
Cdsw-class, [4](#)  
computeFDR, [6](#)  
computeFDRwithID, [6](#)  
create\_fgcz\_fasta\_db, [8](#)  
createDecoyDB, [7](#)

fdrSample, [9](#)

greedy, [9](#)  
greedyRes2Matrix, [10](#)

loadContaminantsFasta2019, [11](#)  
loadContaminantsFasta2021, [11](#)

makeID, [12](#)  
makeIDUnip, [13](#)  
masses, [13](#)

objectiveMS1Function, [14](#)

pepprot, [14](#)  
plotFDR, [14](#)  
predictScoreFDR, [15](#)  
prepareMatrix, [16](#)  
protpepmetashort, [17](#)  
prozor, [17](#)

readjustWindows, [17](#)  
readPeptideFasta, [18](#)  
removeSignalPeptide, [19](#)  
reverseSeq, [19](#)

writeFasta, [20](#)