Package 'protViz'

July 23, 2025

Type Package

Title Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics

Version 0.7.9

Depends R (>= 4.1), methods

Imports Rcpp (>= 1.0)

LinkingTo Rcpp (>= 1.0)

RcppModules FastaMod

Suggests lattice, testthat, xtable

Description Helps with quality checks, visualizations and analysis of mass spectrometry data, coming from proteomics experiments. The package is developed, tested and used at the Functional Genomics Center Zurich <https://fgcz.ch>. We use this package mainly for prototyping, teaching, and having fun with proteomics data. But it can also be used to do data analysis for small scale data sets.

License GPL-3

URL https://github.com/cpanse/protViz/

BugReports https://github.com/cpanse/protViz/issues

Collate aa2mass.R centroid.R comet.R deisotoper.R de_novo.R findNN.R fragmentIon.R genMod.R iTRAQ2GroupAnalysis.R lcmsmap.R mascot.R mdp.R parentIonMass.R peakplot.R pepxml.R pgImporter.R pgLFQ.R pp.R psm.R queue.R PTM_MarkerFinder.R ssrc.R zzz.R RcppExports.R protViz-package.R

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

Author Christian Panse [aut, cre] (ORCID: <https://orcid.org/0000-0003-1975-3064>), Jonas Grossmann [aut] (ORCID: <https://orcid.org/0000-0002-6899-9020>), Simon Barkow-Oesterreicher [ctb]

Maintainer Christian Panse <cp@fgcz.ethz.ch>

Date/Publication 2023-12-12 18:00:06 UTC

Contents

AA . . . 3
aa2mass . . . 4
ADPR . . . 5
as.psmSet . . . 5
assignPlatePosition . . . 6
averagine . . . 7
blockRandom . . . 8
centroid . . . 8
deisotoper . . . 10
de_novo . . . 11
Fasta . . . 12
fetuinLFQ . . . 13
findNN . . . 15
fragmentIon . . . 17
genMod . . . 20
HexNAc . . . 21
insertSamples . . . 22
iRTpeptides . . . 23
iTRAQ . . . 23
iTRAQ2GroupAnalysis . . . 24
lcmsmap . . . 25
mascot . . . 26
massDeviationPlot . . . 28
msms . . . 29
parentIonMass . . . 29
peakplot . . . 31
pgImporter . . . 33
pgLFQaov . . . 34
pgLFQfeature . . . 35
pgLFQprot . . . 36
pgLFQtNpq . . . 37
pressureProfile . . . 39
pressureProfilePlot . . . 40
pressureProfileSummary . . . 41
psm . . . 42
PTM_MarkerFinder . . . 43
PTM_MarkerFinder_util . . . 46
ssrc . . . 47

Index **49**

AA *AA - amino acid table*

Description

Among other attributes it contains '1-letter code', 'monoisotopic mass' and 'average mass' of each amino acids.

Format

contains a table

Value

returns a `data.frame`

Author(s)

Christian Panse 2013

See Also

- https://www.matrixscience.com/help/aa_help.html
- https://education.expasy.org/student_projects/isotopident/htdocs/aa-list.html
- https://www.unimod.org

Examples

```
data(AA)
AA
AA.lm<-lm(AA$Monoisotopic ~ AA$Average)

plot(AA$Monoisotopic, AA$Average);
abline(AA.lm, col='grey')
text(AA$Monoisotopic, AA$Average, AA$letter1, pos=3)

plot(AA$Average-AA$Monoisotopic)
axis(3,1:20,AA$letter1);
abline(v=1:20,col='grey')



# computes monoisotopic mass out of formula using the CDK package
## Not run:
 if (require(rcdk)){
plot(AA$Monoisotopic,
 sapply(AA$formula, function(x){
 get.formula(as.character(x), charge = 1)@mass
 }))
```

```
}

## End(Not run)
    ## Not run:
    if (require(XML)){
unimodurl <- url("http://www.unimod.org/xml/unimod_tables.xml")
unimod.list <- XML::xmlToList(
  XML::xmlParse(
    scan(unimodurl, what = character())))
unimod.AA <- data.frame(
  do.call('rbind', unimod.list$amino_acids))
rownames(unimod.AA) <- unimod.AA$one_letter
    }

## End(Not run)
```

`aa2mass` *determine the weight if a fiven amino acid sequence*

Description

The function determines the weight of each amino acid fiven sequence.

Usage

```
aa2mass(peptideSequence, mass=AA$Monoisotopic, letter1=AA$letter1)
```

Arguments

`peptideSequence`	a double vector which can be considered as query objects.
`mass`	A vector of size 20 containing the weight of the AA.
`letter1`	AA letter 1 code in the same size and order as the `mass` attribute.

Details

For the computation no C-Term and N-Term is considered. `aa2mass` is useful if you have AA modifications.

Author(s)

Christian Panse 2014

Examples

```
peptides<-c('HTLNQIDSVK', 'ALGGEDVR', 'TPIVGQPSIPGGPVR')

C_term <- 17.002740
N_term <- 1.007825
H_ <- 1.008

unlist(lapply(aa2mass(peptides), sum)) + C_term + N_term + H_ - parentIonMass(peptides)

# determine the fragment ions
lapply(aa2mass(peptides), function(x){fragmentIon(x)[[1]]})

# an example with [STY] AA modification

peptide<-'HTLNQIDSVK'
mod <- rep(0.0, nchar(peptide)); mod[8] <- 79.998;

aa2mass(peptide)[[1]] + mod
```

ADPR *ADP Ribosylated Peptide*

Description

Data for reproducing Figures 1-5 in doi:10.1021/jasms.0c00040.

Author(s)

Christian Panse 2022

---

as.psmSet *psmSet - a set of peptide spectrum matches*

Description

defines a class of peptide spectrum matches.

Usage

```
is.psmSet(object)


 ## S3 method for class 'mascot'
as.psmSet(object, ...)
```

Arguments

object	an psmSet S3 class object
...	whatoever

Details

the is.psmSet method checks if the as input given objects fullfills the proterties to be an psmSet object.

as.psmSet transformas an object into a psmSet object.

Value

while is.psmSet returns TRUE or FALSE, as.psmSet returns an instance of psmSet or NULL.

Author(s)

Christian Panse 2017

---

assignPlatePosition *Assign an instrument queue configuration to a plate*

Description

The function implements a space-filling curve mapping 1D to 2D. This function aims to assign a sequence of samples to an instrument plate, e.g., 48 well plate 85.4x127.5mm.

Usage

```
assignPlatePosition(
  S,
  x = as.character(1:8),
  y = c("A", "B", "C", "D", "E", "F"),
  plate = 1:4,
  volume = 1,
  reserve = 46:48
)
```

Arguments

S	input data frame
x	a vector of possible x-coordinates of the plate
y	a vector of possible y-coordinates of the plate
plate	a vector of plates
volume	injection volume
reserve	block plate positions

Value

a data.frame

Examples

```
iris[c(1:15, 51:65, 101:115), ] |>
  blockRandom(x = "Species", check=FALSE) |>
  assignPlatePosition()
```

---

averagine *averagine - a data set conatining isotope envelopes of averagine peptides*

Description

generated using proteowizards IsotopeEnvelopeEstimator class.

Format

first column mass, column 1:8 l2 normalized isotope intensities

Author(s)

Witold Wolski and Christian Panse 2013

References

A general approach to calculating isotopic distributions for mass spectrometry James A. Yergey Volume 52, Issues 2 1 September 1983, Pages 3379

doi:10.1016/00207381(83)850530 https://proteowizard.sourceforge.io/

Examples

```
data(averagine)
r<-seq(0,1,length=200); cm<-c(rgb(r,r,r), '#555599')
image(m<-as.matrix(averagine), col=rev(cm), axes=FALSE, main='protViz
averagine data - normalized isotope intensities ',
sub='the blue color indicates fields with value 0.',
xlab='isotops',
ylab='mass');
box()
axis(1, seq(0,1,length=nrow(m)), 1:nrow(m));
axis(2, seq(0,1,length=10), colnames(m)[seq(1,ncol(m), length=10)])
```

blockRandom *Derive a randomization of a table.*

Description

Derive a randomization of a table.

Usage

```
blockRandom(S, x = NA, check = TRUE)
```

Arguments

S	a data.frame
x	the column name of the group used for the block randomization.
check	check if the size of each group is equal.

Value

returns a randomization order.

Examples

```
set.seed(1)
iris[c(1:2, 51:52, 101:103), ] |>
  blockRandom(x = "Species", check=FALSE)
```

centroid *Centroid a spectrum acquired in profile mode*

Description

returns a centroid spectrum of a recorded profile mode spectrum.

Usage

```
centroid(mZ, intensity, tolppm=100, debug=FALSE)
```

Arguments

mZ	Numerical vector of profile recorded data and sorted mZ values.
intensity	corresponding intensity values.
tolppm	maximal distance in Da between to profile mZ values. default is set to 100ppm.
debug	if true all peak profiles are plotted. default is false.

Details

the method is tested on an Orbitrap Fusion Lumos FSN20242 data set.

Value

returns a `data.frame` with a mZ and a intensity column.

Note

thanks to Nienke Meekel and Andrea Brunner (kwrwater.nl) and Witold E. Wolski.

Author(s)

Christian Panse and Jonas Grossmann, April 2020

See Also

- MSnbase functions-Spectrum.R
- https://proteowizard.sourceforge.io/

Examples

```
# Orbitrap Fusion Lumos FSN20242
# p2722/.../.../stds_pos_neg_MS_highconc_UVPD_50_300.raw
# scan 1959
# CC(C)(C)C(O)C(OC1=CC=C(Cl)C=C1)N1C=NC=N1 1
# exact.mass 295.1088
# FTMS + p ESI d Full ms2 296.1162@uvpd50.00

 p <- protViz:::.getProfileMS2()

 # determine eps
 plot((diff(p$mZ)) ~ p$mZ[2:length(p$mZ)], log='y');
 abline(h=0.1, col='red')
 points(p$mZ , 1E-4 * p$mZ, col='grey', type='l')
 abline(v=296.1162, col='cyan')

 op <- par(mfrow=c(2, 1))
 plot(p$mZ, p$intensity, type='h',
   main='profile', xlim=c(100,300))
 abline(v=296.1162, col='cyan')

 plot(centroid(p$mZ, p$intensity),type='h',
   main="centroid",xlim=c(100,300))
 par(op)


 op <- par(mfrow=c(2, 1), ask = TRUE)
     rv <- centroid(p$mZ, p$intensity, debug = TRUE)
```

`deisotoper` *find isotop pattern in a given mass spectrum*

Description

`deisotoper` returns the indices and scores of the isotop pattern. The score is computed by considering the isotop L2 norm of a given iostop model. `protViz` contains the `averagine` data.

Usage

```
deisotoper(data, Z=1:4, isotopPatternDF=averagine, massError=0.005, plot=FALSE)
```

Arguments

`data`	A mass spectrometric measurement containing a list.
`Z`	Charge states to be considered. The default is `Z=1:4`
`isotopPatternDF`	a data frame containing isotope envelopes of peptide averagine.
`massError`	mass error in Dalton. default is 0.005.
`plot`	boolean if the isotops should be plotted. The default is false.

Details

The output of the `deisotoper` function is a `list` data structure containing for each input MS data set a `result`, `score`, and `group` lists.

`result` is a list which contains for each charge state a list of isotop groups. Analoge to `result` a score is provide in the `score` list.

`group` provides information about the charge states of each isotop cluster.

The algorithm for finding the isotop chains as implemented in protViz is based on the idea of 'Features-Based Deisotoping Method for Tandem Mass Spectra'. Each peak represents one node of a graph `G=(V,E)` with `V = {1, ..., n}`. The edges are defined by `(v, node_array_G[v])` iff `node_array_G[v] > -1`. In other words an edge is defined between two peaks if the mZ difference is below the given massError. The isotop chains are determined by a DFS run. The time complexity is `O(n log (n))` where n is the number of peaks.

Author(s)

Witold Eryk Wolski, Christian Trachsel, and Christian Panse 2013

References

isotopic-cluster graphs:

Features-Based Deisotoping Method for Tandem Mass Spectra, Zheng Yuan Jinhong Shi Wenjun Lin Bolin Chen and Fang-Xiang Wu Advances in Bioinformatics Volume 2011 (2011), Article ID 210805, 12 pages doi:10.1155/2011/210805

Examples

```
# example1 - tandem ms
x <- list(mZ = c(726.068, 726.337, 726.589, 726.842, 727.343, 727.846, 728.346,
    728.846, 729.348, 730.248, 730.336, 730.581, 730.836),
    intensity = c(6.77850e+03, 2.81688e+04, 6.66884e+04, 1.22032e+07,
        9.90405e+06, 4.61409e+06, 1.50973e+06, 3.33996e+05, 5.09421e+04,
        1.15869e+03, 2.14788e+03, 5.37853e+03, 5.79094e+02))

# the plain C interface function
out1 <- .Call("deisotoper_main", x$mZ, x$intensity, Z=1:4, averagine,
    massError=0.01, PACKAGE="protViz")

out<-deisotoper(data=list(x), Z=2:4, isotopPatternDF=averagine)

# example2 - a ms from heavy labeld peptide in water background
x <- list(mZ=c(642.572, 643.054, 643.569, 644.062, 644.557),
    intensity=c(17000, 25000, 12000, 9000,4000))

diff(x$mZ)
diff(x$mZ,lag=2)
diff(x$mZ, difference=2)

out2<-deisotoper(data=list(x), Z=1:2, isotopPatternDF=averagine,
    massError=0.02, plot=TRUE)
```

---

de_novo *de-novo on tandem ms*

Description

This function computes the mass differences of MS2 peaks and matches the 'delta masses' to the amino acid weights.

Usage

```
de_novo(data)
```

Arguments

data	ms data set.

Details

This function should only be used for demonstration!

Author(s)

Christian Panse 2013

Examples

```
data(msms)
de_novo(msms[[1]])
```

Fasta *FASTA format reader*

Description

Implements as Rcpp module performing tryptic digests and summaries. The support is limited for protein sequences.

Usage

```
Fasta
```

Value

returns a `list` or `vector` objects.

Author(s)

Christian Panse <cp@fgcz.ethz.ch> 2006-2017

References

https://en.wikipedia.org/wiki/FASTA_format

Examples

```
# >sp|P12763|FETUA_BOVIN Alpha-2-HS-glycoprotein OS=Bos taurus
  fname <- system.file("extdata", name='P12763.fasta', package = "protViz")
  F <- Fasta$new(fname)
  F
  F$summary()
  F$getTrypticPeptides()

  plot(parentIonMass(F$getTrypticPeptides()), ssrc(F$getTrypticPeptides()),
    log='xy')
  text(parentIonMass(F$getTrypticPeptides()), ssrc(F$getTrypticPeptides()),
    F$getTrypticPeptides(), pos=3, cex=0.5)
```

`fetuinLFQ` *fetuinLFQ - A data set for evaluation of relative and absolute label-free quantification methods.*

Description

This data set contains the read-out abundance for three different methods (APEX, emPAI, T3PQ) for seven proteins from a complex yeast background (abundance levels unchanged) and the Fetuin that was spiked into the extracts in concentrations ranging from 20fmol to 300fmol on column (in triplicates).

Samples were analyzed on a LTQ-FTICR Ultra mass spectrometer (Thermo Fischer Scientific, Bremen, Germany) coupled to an Eksigent-Nano-HPLC system (Eksigent Technologies, Dublin (CA), USA).

The data can be derived out of the mzXML files:

1 186240760 2008-12-11 20080816_01_fetuin.mzXML

2 179013841 2008-12-11 20080816_02_fetuin_0.mzXML

3 178692924 2008-12-11 20080816_03_fetuin_0.mzXML

4 178608302 2008-12-11 20080816_04_fetuin_0.mzXML

5 180995463 2008-12-11 20080816_05_fetuin_20.mzXML

6 175934898 2008-12-11 20080816_06_fetuin_20.mzXML

7 178364454 2008-12-11 20080816_07_fetuin_20.mzXML

8 184376874 2008-12-11 20080816_08_fetuin_40.mzXML

9 187205031 2008-12-11 20080816_09_fetuin_40.mzXML

10 185646382 2008-12-11 20080816_10_fetuin_40.mzXML

11 183498944 2008-12-11 20080816_11_fetuin_60.mzXML

12 184752098 2008-12-11 20080816_12_fetuin_60.mzXML

13 186243524 2008-12-11 20080816_13_fetuin_60.mzXML

14 187794055 2008-12-11 20080816_14_fetuin_80.mzXML

15 183463368 2008-12-11 20080816_15_fetuin_80.mzXML

16 186331090 2008-12-11 20080816_16_fetuin_80.mzXML

17 188027950 2008-12-11 20080816_17_fetuin_100.mzXML

18 186881098 2008-12-11 20080816_18_fetuin_100.mzXML

19 187219923 2008-12-11 20080816_19_fetuin_100.mzXML

20 187157009 2008-12-11 20080816_20_fetuin_120.mzXML

21 190008885 2008-12-11 20080816_21_fetuin_120.mzXML

22 184226648 2008-12-11 20080816_22_fetuin_120.mzXML

23 190681343 2008-12-11 20080816_23_fetuin_160.mzXML

24 194653066 2008-12-11 20080816_24_fetuin_160.mzXML

25 191045349 2008-12-11 20080816_25_fetuin_160.mzXML

26 184869491 2008-12-11 20080816_26_fetuin_200.mzXML

27 185490782 2008-12-11 20080816_27_fetuin_200.mzXML

28 185635558 2008-12-11 20080816_28_fetuin_200.mzXML

29 187743192 2008-12-11 20080816_29_fetuin_300.mzXML

30 190613235 2008-12-11 20080816_30_fetuin_300.mzXML

31 189570723 2008-12-11 20080816_31_fetuin_300.mzXML

32 226404551 2008-12-11 20080819_32_fetuin.mzXML

contained in a 3GBytes compressed tar ball http://fgcz-data.uzh.ch/public/fqms.tgz md5=af804e209844d055c0edec

The t3pq data can be derived using the code from https://sourceforge.net/projects/fqms/

Format

A data set with approx. 600 rows and four variables, 4KBytes file size, derived out of 32 single LC-MS runs, 5GBytes size, of a protein mixture.

Author(s)

Christian Panse 2009, 2010

References

doi:10.1016/j.jprot.2010.05.011

Examples

```
library(lattice)
data(fetuinLFQ)

cv<-1-1:7/10
t<-trellis.par.get("strip.background")
t$col<-(rgb(cv,cv,cv))
trellis.par.set("strip.background",t)

my.xlab="Fetuin concentration spiked into experiment [fmol]"
my.ylab<-"Abundance"

xyplot(abundance~conc|prot*method,
    data=fetuinLFQ$apex,
    groups=prot,
    aspect=1,
    panel = function(x, y, subscripts, groups) {
        if (groups[subscripts][1] == "Fetuin")  {
            panel.fill(col="#ffcccc")
        }
                panel.grid(h=-1,v=-1)
                panel.xyplot(x, y)
                panel.loess(x,y, span=1)
```

```
            },
        xlab=my.xlab,
        ylab=my.ylab
    )


    xyplot(abundance~conc|prot*method,
        data=fetuinLFQ$empai,
        groups=prot,
        aspect=1,
        panel = function(x, y, subscripts, groups) {
            if (groups[subscripts][1] == "Fetuin")  {
                panel.fill(col="#ffcccc")
            }
                    panel.grid(h=-1,v=-1)
                    panel.xyplot(x, y)
                    panel.loess(x,y, span=1)
                },
        xlab=my.xlab,
        ylab=my.ylab
    )


    xyplot(abundance~conc|prot*method,
        data=fetuinLFQ$t3pq,
        groups=prot,
        aspect=1,
        panel = function(x, y, subscripts, groups) {
            if (groups[subscripts][1] == "Fetuin")  {
                panel.fill(col="#ffcccc")
            }
                    panel.grid(h=-1,v=-1)
                    panel.xyplot(x, y)
                    panel.loess(x,y, span=1)
            if (groups[subscripts][1] == "Fetuin")  {
                panel.text(min(fetuinLFQ$t3pq$conc),
                    max(fetuinLFQ$t3pq$abundance),
                    paste("R-squared:",
                    round(summary(lm(x~y))$r.squared,2)),
                    cex=0.75,
                    pos=4)
            }
                },
        xlab=my.xlab,
        ylab=my.ylab
    )
```

---

findNN *find index of nearest neighbor*

Description

Given a vector of sorted double values vec of size n and a vector of m query objects q.

findNN determines for each element q[i] in q the nearest neighbor index o so that the following remains true:

there is no element k with $1 \leq k \leq n$ and k is not o so that

abs(vec[k] - q[i]) < abs(vec[o] - q[i]).

Usage

```
findNN(q, vec, check)
findNN_(q, vec, check)
```

Arguments

q	a double vector which can be considered as query objects.
vec	a sorted double vector which can be considered as a data base.
check	boolean enables test if vec is sorted. default is FALSE.

Details

The internal algorithm of findNN is implemented as binary search. findNN has $O(m * log_2(n))$ time complexity where n is defined as length(vec) and m is length(m).

findNN is implemented using C library function - bsearch(), while findNN_ uses C++11 STL function lower_bound().

Author(s)

Christian Panse 2007, 2008, 2009, 2010, 2012 , 2015 based on the C++ STL lower_bound method.

References

- https://cplusplus.com/reference/algorithm/lower_bound/

See Also

findInterval

Examples

```
(NNidx <- findNN(q <- c(1, 1.0001, 1.24, 1.26), DB <- seq(1, 5 , by = 0.25)))
(NNidx == c(1, 1, 2, 2))

DB <- sort(rnorm(100, mean=100, sd = 10))

# should be 0
unique(DB[findNN(DB,DB)] - DB)
```

```
q <- rnorm(100, mean=100)

idx.NN <- findNN(q,DB)
hist(DB[findNN(q,DB)] - q)

# definition of findNN holds
i <- 1:5
findNN(3.5, i)

i <- 1:6
findNN(3.5, i)

 # compare ANSI-C binary search with C++ std::lower_bound
DB <- c(rep(1.0, 3), rep(2.0, 3))
q <- c(-1, 1.0, 1.01, 1.5, 1.9)
abs(DB[findNN(q, DB)] - q)
abs(DB[findNN_(q, DB)] - q)


DB <- sort(rnorm(100, mean=100, sd=10))
# should be 0
unique(DB[findNN_(DB,DB)] - DB)

q <- rnorm(100, mean=100)

idx.NN <- findNN_(q, DB)
hist(DB[findNN_(q, DB)] - q)

# definition of findNN_ holds
i <- 1:5
findNN_(3.5, i)

i <- 1:6
findNN_(3.5, i)
```

`fragmentIon` *Compute the b and y Fragment Ions of a Peptide Sequence*

Description

The function computes the fragment ions of a peptide sequence, according to the rules of fragmentation in a collision cell. It can be either CID fragmentation (collision-induced dissociation) or ETD (electron transfer dissociation) which are common in proteomics experiments. All are in a positive mode.

If multiple peptide sequences are given it returns a list of fragment ion table.

Usage

```
fragmentIon(sequence, FUN, modified, modification, N_term, C_term)
```

Arguments

sequence	peptide sequence encoded as character sequence using the 20 amino acid letters or a double vector of amino acid weights.
FUN	the function to be applied to compute further ions. If no function is assigned fragmentIon will use defaultIon.
modified	a vector of interger containing the varmod id of the peptide position.
modification	a double vector of defining the varmod mass in Dalton.
N_term	N-Term mass in Dalton. Default is set to 1.007825.
C_term	C-Term mass in Dalton. Default is set to 17.002740.

Details

The fragment ions of a peptide can be computed following the rules proposed in PMID:6525415. Beside the b and y ions the FUN argument of fragmentIon defines which ions are computed. the default ions beeing computed are defined in the function defaultIon. The are no limits for defining other forms of fragment ions for ETD (c and z ions) CID (b and y ions).

NOTE: for simplicity, we have set a Carbamidomethyl (C) fixed modification with 160.030649 (Cysteine mono mass is 103.00919). The fixed modifications setting are not enabled in the package yet.

If only a vector of amino acids weights is given fragmentIon computes the ions series according to weights. If this case applies, the function returns only a list having one data frame.

Author(s)

Christian Panse, Bertran Gerrits 2006.

References

doi:10.1002/bms.1200111109

Protein Sequencing and Identification Using Tandem Mass Spectrometry, Michael Kinter and Nicholas E. Sherman, Wiley-Interscience; 1st edition (September 18, 2000)

See Also

first used in peakplot.

Examples

```
# Example 1
peptide.AA<-"KINHSFLR";

peptide.AA.weights <- c(128.09496,113.08406,114.04293,
    137.05891,87.03203,147.06841,113.08406,156.10111);

fragmentIon(peptide.AA);

fragmentIon(peptide.AA.weights);
```

```
HCD_Ion <- function(b, y){
    return(cbind(b = b, y = y))
}


ETD_Ion <- function(b, y){
    Hydrogen <- 1.007825
    Oxygen <- 15.994915
    Nitrogen <- 14.003074

    y_0 <- y - Oxygen - Hydrogen - Hydrogen
    c <- b + (Nitrogen + (3 * Hydrogen))
    z <- y - (Nitrogen + (3 * Hydrogen))

    return(cbind(y_0, c, z))
}

fragmentIon(peptide.AA, FUN = ETD_Ion)

peptides<-c('HTLNQIDSVK', 'ALGGEDVR', 'TPIVGQPSIPGGPVR')

pim <- parentIonMass(peptides)
fi <- fragmentIon(peptides)
(df <- as.data.frame(fi))

op <- par(mfrow=c(3,1));
for (i in 1:length(peptides)){
    plot(0, 0,
    xlab='m/Z',
    ylab='',
    xlim=range(c(fi[[i]]$b,fi[[i]]$y)),
    ylim=c(0,1),
    type='n',
    axes=FALSE,
    sub=paste(peptides[i], "/", pim[i], "Da"));
    box()
    axis(1, fi[[i]]$b, round(fi[[i]]$b,1), las=2)
    axis(1, fi[[i]]$y, round(fi[[i]]$y,1), las=2)

    pepSeq<-strsplit(peptides[i], "")
    axis(3,fi[[i]]$b, paste("b", row.names(fi[[i]]),sep=''),las=2)
    axis(3,fi[[i]]$y, paste("y", row.names(fi[[i]]),sep=''),las=2)

    text(fi[[i]]$b, rep(0.3, nchar(peptides[i])),
    pepSeq[[1]],pos=3,cex=4, lwd=4, col="#aaaaaaaa")

    abline(v=fi[[i]]$b, col='red')
    abline(v=fi[[i]]$y, col='blue',lwd=2)
}
par(op)
```

```
fi <- fragmentIon(c("ATSFYK","XGXFNAGVGK"))[[2]]
fi$b[1] + fi$y[9]
fi$b[2] + fi$y[8]


ION2C <- function(b, y){
    Hydrogen <- 1.007825
    Oxygen <- 15.994915
    Nitrogen <- 14.003074

    # yo <- fi$y - Oxygen - Hydrogen - Hydrogen
    c <- b + (Nitrogen + (3 * Hydrogen))
    z <- y - (Nitrogen + (3 * Hydrogen))

    # compute doubly charged fragment ions
    b2 <- (b + Hydrogen) / 2
    y2 <- (y + Hydrogen) / 2

    return(cbind(b, y, b2 ,y2))
}
```

---

genMod *Generates all possible modification of a given peptide seqence and modification pattern*

Description

This function can be used to screen precursor masses for meaningful mass shifts which could correspond to post translational modifications. We suggest to only use a maxiumum of a handful of mass shifts.

Usage

```
genMod(sequences, modificationPattern, nModification=2)
```

Arguments

sequences	peptide sequences encoded as character sequence using the 20 amino acid letters.
modificationPattern	a rbind list structure containing the mono and avg mass as well as the description of a the modifications.
nModification	number of maximal modifications.

Details

t.b.d.

Author(s)

Hubert Rehrauer and Christian Panse 2012

Examples

```
ptm.0<-cbind(AA="-",
    mono=0.0, avg=0.0, desc="unmodified", unimodAccID=NA)

ptm.616<-cbind(AA='S',
    mono=-27.010899, avg=NA, desc="Substituition",
        unimodAccID=616)

ptm.651<-cbind(AA='N',
    mono=27.010899, avg=NA, desc="Substituition",
        unimodAccID=651)


m<-as.data.frame(rbind(ptm.0, ptm.616, ptm.651))

genMod(c('TAFDEAIAELDTLNEESYK','TAFDEAIAELDTLSEESYK'), m$AA)
```

HexNAc *HexNAc - Analysis of N-HexNAc glycopeptides by LC-MS/MS, using HCD and ETD fragmentation techniques*

Description

HexNAc is intented for demonstration of the PTM_MarkerFinder methode descriped in [PROTEOMICS pmic.201300036]. It contains the mass pattern (m/z) of c(126.05495, 138.05495, 144.06552, 168.06552, 186.07608, 204.08665).

A sample containing enriched glycopeptides from yeast was analysed by LC-MS/MS on a LTQ-Orbitrap Velos instrument using sequential HCD and ETD fragmentation techniques. The dataset contains only 11 tandem mass spectra extracted from this experiment. Three pairs of HCD/ETD spectra correspond to peptides carrying N-HexNAc modification identified with high confidence (6 spectra in total). One pair of HCD/ETD spectra corresponds to peptides carrying N-HexNAc modification, identified with very low confidence (2 spectra in total). The remaining 3 spectra are from unmodified peptides.

Format

A data set consists of eleven tandem mass spectra.

Author(s)

Paolo Nanni, Christian Panse, 2013

Examples

```
data(HexNAc)
HexNAc[[1]]

plot(HexNAc[[1]]$mZ, HexNAc[[1]]$intensity, type='h')
```

`insertSamples` *Insert sample on a given position*

Description

Insert sample on a given position

Usage

```
insertSamples(
  S,
  stdName = "autoQC01",
  stdPosX = "8",
  stdPosY = "F",
  plate = 1,
  volume = 2,
  method = "",
  ...
)
```

Arguments

`S`	input `data.frame`
`stdName`	name of the sample
`stdPosX`	x location on the plate
`stdPosY`	y location on the plate
`plate`	number of the plate
`volume`	injection volume
`method`	a path to the method file (optional)
`...`	addition parameter, e.g., `howoften = 1 howmany = 1`.

Value

`data.frame`

Examples

```
iris[c(1:15,51:65,101:115), ] |>
  assignPlatePosition() |>
  insertSamples(howoften=4, begin=FALSE, end=FALSE,
    stdPosX='6', stdPosY='F', plate=1, stdName = "clean")
```

---

iRTpeptides *iRT peptides - independent retention time peptides*

Description

iRTpeptides data are used for genSwathIonLib rt normalization assuming.

Format

contains a table

Author(s)

Jonas Grossmann and Christian Panse 2013

References

Using iRT, a normalized retention time for more targeted measurement of peptides. Escher C, Reiter L, MacLean B, Ossola R, Herzog F, Chilton J, MacCoss MJ, Rinner O. Source Proteomics. 2012 Apr;12(8):1111-21. doi: 10.1002/pmic.201100463.

Examples

```
plot(sort(iRTpeptides$rt))

plot(pim<-parentIonMass(as.character(iRTpeptides$peptide)) ~ iRTpeptides$rt)
```

---

iTRAQ *iTRAQ - A small 8-plex iTRAQ data set with confident identified peptides from 5 proteins.*

Description

This dataset contains quantification data from iTRAQ (isobaric tag for relative and absolute quantification) LC-MSMS identified peptides using the software ProteinPilot from ABSciex. The data is generated on a Thermo LTQ-Oribtrap. The mascot generic files are produced with MascotDistiller and searched with ProteinPilot V.4. The peptide summary export is used and filtered towards confident peptides (Confidence above 90) and only columns with Protein accession, peptide sequence, the 8 reporter ion channels (area) are kept as well as the protein description. The rest of the peptide summary is deleted. This data is only a small illustrative part out of the whole data produced. The package design is done in a way, that it is not restricted to ProteinPilot exports but can be used with all other kind of iTRAQ tools, as long as each reporter channel with areas or intensities can be exported. The main point of this function is, to get a hand on what peptides and proteins are used to quantify. Also to be able to not use ratio based quantification but to clearly specify the input and the experimental design.

Format

A data set with approx. 260 rows and 10 variables, 40KBytes file size, derived from biological data from an iTRAQ project..

Author(s)

Jonas Grossmann, Christian Panse, 2012

References

doi:10.1016/j.jprot.2012.09.011

Examples

```
data(iTRAQ)
iTRAQ[1:10,]
```

`iTRAQ2GroupAnalysis` *iTRAQ two group analysis*

Description

The function performes a two group analysis with a t-test for data sets like `iTRAQ`. Result files are generated where the 2 groups are compared for each protein, while a p-value is calculated. If plot equals TRUE, the boxplots are drawn.

Usage

```
iTRAQ2GroupAnalysis(data, group1, group2,
    INDEX, FUN=function(x){return(x)}, plot)
```

Arguments

data	a data set like iTRAQ.
group1	a vector of column ids.
group2	a vector of column ids.
INDEX	list of factors, each of same length as X. The elements are coerced to factors by as.factor.
FUN	function for doing the data transformation, e.g. log, asinh.
plot	boolean. if TRUE boxplots are drawn.

Details

t.b.d.

Author(s)

Christian Panse, Jonas Grossmann 2011

Examples

```
data(iTRAQ)
par(mfrow=c(2,3))
qProt<-iTRAQ2GroupAnalysis(data=iTRAQ,
 group1=c(3,4,5,6),
 group2=7:10, INDEX=iTRAQ$prot, plot=TRUE)
qProt
qPeptide<-iTRAQ2GroupAnalysis(data=iTRAQ,
 group1=c(3,4,5,6),
 group2=7:10,
 INDEX=paste(iTRAQ$prot,iTRAQ$peptide), plot=FALSE)
```

---

lcmsmap *LC-MS Map*

Description

The function graphs a LC-MS map having for all charge states and each single charge state. The rt is on the x axis and the y axis represents the peptide mass.

Usage

```
lcmsmap(data, charges=2:3, score.cutoff = 30, ...)
```

Arguments

`data`	an R data object as it can be obtained by `mascotDat2RData.pl` or by using mascots export function.
`charges`	a vector of charge states to be displayed.
`score.cutoff`	a numeric value taken as score cut-off.
`...`	pass arguments to `plot` function.

Author(s)

Christian Panse and Jonas Grossmann 2013;

References

idea taken from from the ISB TPP Pep3D view PMID: 15228367

`mascot` *Generic methods for mascot results*

Description

does something with a `mascot` object using the package's functions, e.g., `peakplot` or `lcmsmap`.

Usage

```
  ## S3 method for class 'mascot'
is(object, class2)

  ## S3 method for class 'mascot'
plot(x, ...)

  ## S3 method for class 'mascot'
as.data.frame(x, ...)

  ## S3 method for class 'mascot'
summary(object, ...)

  ## S3 method for class 'mascot_query'
is(object, class2)

  ## S3 method for class 'mascot_query'
plot(x, obj = NULL, FUN = defaultIon, ...)
```

Arguments

x	a mascot_query or mascot S3 class object.
object	a mascot S3 class object.
obj	a mascot S3 class object.
class2	dummy
...	arguments will be passwed through.
FUN	ion series.

Details

the object has been generated by using the mascot server command ./export_dat_2.pl $EXPORTOPTIONS file=$DAT.

$EXPORTOPTIONS is defined as "_minpeplen=5 _server_mudpit_switch=0.000000001 _showsubsets=1 _sigthreshold=0.05 do_export=1 export_format=XML group_family=1 pep_calc_mr=1 pep_delta=1 pep_end=1 pep_exp_mr=1 pep_exp_mz=1 pep_exp_z=1 pep_expect=1 pep_isbold=1 pep_isunique=1 pep_miss=1 pep_query=1 pep_rank=1 pep_scan_title=1 pep_score=1 pep_seq=1 pep_start=1 pep_var_mod=1 peptide_master=1 prot_acc=1 prot_cover=1 prot_desc=1 prot_empai=1 prot_hit_num=1 prot_len=1 prot_mass=1 prot_matches=1 prot_pi=1 prot_score=1 prot_seq=1 protein_master=1 query_master=1 query_params=1 query_peaks=1 query_qualifiers=1 query_raw=1 query_title=1 search_master=1 show_format=1 show_header=1 show_masses=1 show_mods=1 show_params=1 show_pep_dupes=1 use_homology=1 user=command line".

$DAT defines the mascot result file.

The output is written as XML file the following command is applied: XML::xmlToList(XML::xmlParse(xml_file_name)).

xmlParse and xmlToList are function of the XML package.

Value

returns the peakplot return value.

Author(s)

Christian Panse, 2017

References

http://www.matrixscience.com/mascot_support_v2_6.html

See Also

- summary.mascot
- peakplot
- https://CRAN.R-project.org/package=XML

Examples

```
## Not run:
  # plot the top ten highes scored PSMs
  par(ask = TRUE)
  idx <- order(protViz:::.mascot.get.pep_score(S), decreasing = TRUE)[1:10]
  rv.peakplot <- lapply(S$queries[idx], plot)


  myAA <- do.call('rbind', lapply(F225712$masses,
    function(x){
      data.frame(letter1=as.character(x$.attrs), mass=as.numeric(x$text))
      }))

  aa2mass("ELVISR", mass=myAA$mass[1:25], letter1 = myAA$letter1[1:25])


## End(Not run)
```

`massDeviationPlot` *Mass Deviation Plot*

Description

`massDeviationPlot` computes the mass deviation of a data set having the list attributes `pepmass` and `peptideSequence`.

Usage

```
mdp(data, sub)
```

Arguments

`data`	an R data object as it can be obtained by `mascotDat2RData.pl`
`sub`	a sub title for the plot

Details

In this version it igrnoes modified peptides.

Author(s)

Christian Panse 2012-2013;

References

Zubarev R, Mann M. On the proper use of mass accuracy in proteomics. Mol Cell Proteomics. 2007 Mar;6(3):377-81. Epub 2006 Dec 12. PMID: 17164402

Examples

```
# load("F178767.Rdata")
# pdf("massError.pdf",12,9)
# massDeviationPlot (F178767)
# dev.off()
```

`msms` *A data set containing tandem mass spectra of an LCMS experiment.*

Description

The purpose of `msms` is for the demonstration of the peptide identification methods, e.g., `peakplot` or `psm`.

`msms` contains two tandem mass spectra meassured on a linear ion trap coupled to an Orbitrap mass analyzer (Thermo Scientific, Bremen, Germany).

The protein sample originates from 10 mouse brain homogenate, prepared in sterile PBS containing protease inhibitors (Complete; Riche, Switzerland) by repeated extrusion through syringe needles of successive smaller size. To remove gross cellular debris the homogenate was centrifuged for 10 min at 2400 rpm at 4 degrees centigrade.

The data were acquired withing the Sixth Framework Programm (FP6-37457). The project was called SysProt.

Author(s)

Dorothea Rutishauser, Jonas Grossmann, Christian Panse 2008

Examples

```
data(msms)
peakplot("TAFDEAIAELDTLNEESYK", msms[[1]])
```

`parentIonMass` *Compute Parent Ion Mass of a Peptide Sequence*

Description

It computes the parent ion mass of a vector of peptide sequences. The weights of the amino acids and the C-Term are hard coded in the C function. Important: The precursor mass is calculated with an additional proton, so to speak the singly charged precursor!

Usage

```
parentIonMass(sequence, fixmod, NTerm)
```

Arguments

sequence	peptide sequence encoded as character sequence using the 20 amino acid letters.
fixmod	a double vector of length 26 defining the amino acid mass.
NTerm	a double value defining the weight of the NTerm.

Details

The parentIonMass function reqires one argument secuence.

Author(s)

Christian Panse 2006

Examples

```
fetuin<-list( sequence=c('MK', 'SFVLLFCLAQLWGCHSIPLDPVAGYK',
'EPACDDPDTEQAALAAVDYINK',
'HLPR', 'GYK', 'HTLNQIDSVK', 'VWPR',
'RPTGEVYDIEIDTLETTCHVLDPTPLANCSVR',
'QQTQHAVEGDCDIHVLK', 'QDGQFSVLFTK',
'CDSSPDSAEDVR', 'K', 'LCPDCPLLAPLNDSR',
'VVHAVEVALATFNAESNGSYLQLVEISR',
'AQFVPLPVSVSVEFAVAATDCIAK',
'EVVDPTK', 'CNLLAEK', 'QYGFCK',
'GSVIQK', 'ALGGEDVR',
'VTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR',
'AHYDLR', 'HTFSGVASVESSSGEAFHVGK',
'TPIVGQPSIPGGPVR', 'LCPGR', 'IR', 'YFK', 'I'),
description=
    "FETUA_BOVIN Alpha-2-HS-glycoprotein ")

plot(peptideMass<-parentIonMass(sequence=fetuin$sequence),
    sub=list(fetuin$description, cex=0.75),
    main="tryptic digested peptides")


AA<-c(71.037114, 114.534940, 160.030649, 115.026943, 129.042593, 147.068414,
    57.021464, 137.058912, 113.084064, 0.000000, 128.094963, 113.084064,
    131.040485, 114.042927, 0.000000, 97.052764, 128.058578, 156.101111,
    87.032028, 101.047679, 150.953630, 99.068414, 186.079313, 111.000000,
    163.063329, 128.550590)

parentIonMass(c("ELVIS", "ELVISK"), fixmod=AA)
```

peakplot *Labelling of Peptide Fragment Mass Spectra*

Description

peakplot labels a mass spectrum from a peptide sequence assignment by using the psm function with the appropriate fragment ion labels. Using peakplot it is possible to compute the fragment ion coverage and customize the fragmentation ions used for labelling.

Usage

```
peakplot(peptideSequence, spec,
    FUN=defaultIon,
    fi=fragmentIon(peptideSequence, FUN=FUN)[[1]],
    sub=paste(peptideSequence, spec$title, sep=" / "),
    itol=0.6,
    pattern.abc="[abc].*",
    pattern.xyz="[xyz].*",
    ion.axes=TRUE, ...)
```

Arguments

peptideSequence	peptide sequence encoded as a character sequence using the 20 amino acid letter code.
spec	a peaklist or a tandem mass spec data structure which is a list having two eq sized vectors calles mZ and intensity. mZ values are sorted.
FUN	the function to be applied to compute further ions. If no function is assigned fragmentIon will use defaultIon.
fi	fragment ion table, if not specified fragmentIon(sequence, FUN=FUN)[[1]] is called. This argument is useful if you have PTM or specific ion series.
sub	a sub title for the plot
itol	fragment ion tolerance default is 0.6 Dalton. values below are considered as hit.
pattern.abc	regexpr pattern for a, b, c like ions. if the pattern does not match ion are not plotted.
pattern.xyz	regexpr pattern for x, y, z like ions.
ion.axes	boolean default is TRUE. determines whether the fragment ion labels should be plotted instead of m/z values.
...	passed to the default plot function.

Details

peakplot computes the in-silico fragment ion, using the fragmentIon function, and matches them again with the MS2. If the mass error between the in-silico peak and the measured one is below 0.6 Da (default setting), the peptide spectrum match (psm) is considered as a hit.

The major objective of the labelling is the avoidance of overlapping labels, for which purpose peakplot applies filtering. A label is only drawn if the corresponding ion count of the m/z peak is higher than a given threshold. Experience from several hundred annotations shows that the 0.9 percentile is a good cut-off value. To most efficiently use the limited screen and printout space and to ensure that labels representing important local peaks are also considered for the drawing, peakplot divides the display space into a number of bins depending on the peptide sequence length along the m/z axis. From these bins, the top n labels are ordered according to abundance. For the visual clustering effect, the abc and xyz ions are drawn on different y-axis levels using different colors. Ion types considered for labelling is dependent on the instrument setting applied during the initial search.

Value

returns a list object containing all the peptide spectrum match relevant information.

Author(s)

Bertran Gerrits, Christian Panse

2006-2017;

References

- PEAKPLOT: Visualizing Fragmented Peptide Mass Spectra in Proteomics (2009) Panse Christian, Gerrits Bertran, Schlapbach Ralph, useR!2009,

  abstract: https://www.r-project.org/conferences/useR-2009/abstracts/pdf/Panse+Gerrits+Schlapbach.pdf,

  slides: https://www.r-project.org/conferences/useR-2009/slides/Panse+Gerrits+Schlapbach.pdf.

See Also

- mascot
- fragmentIon
- psm
- PTM_MarkerFinder
- peakplot has been used for generating figures and supplemental material in:
  doi:10.1074/mcp.M600046MCP200, doi:10.1038/msb4100182, doi:10.1371/journal.pone.0036980, doi:10.1002/pmic.201300036, doi:10.1074/mcp.M115.052548

Examples

```
data(msms)
op <- par(mfrow=c(2,1))
peakplot("TAFDEAIAELDTLNEESYK", msms[[1]])
peakplot("TAFDEAIAELDTLSEESYK", msms[[2]])
par(op)


# filter cand. fragment ions
fi <- fragmentIon("TAFDEAIAELDTLNEESYK")
fi.cyz <- as.data.frame(cbind(c=fi[[1]]$c, y=fi[[1]]$y, z=fi[[1]]$z))

p <- peakplot("TAFDEAIAELDTLNEESYK", spec=msms[[1]],
    fi=fi.cyz,
    itol=0.6,
    ion.axes=FALSE)


# customizing the ion series
ions <- function(b, y){
  Hydrogen <- 1.007825
  Oxygen <- 15.994915
  Nitrogen <- 14.003074

  y0 <- fi$y - Oxygen - Hydrogen - Hydrogen
  c <- b + (Nitrogen + (3 * Hydrogen))
  z <- y - (Nitrogen + (3 * Hydrogen))

  return(cbind(b, y, c ,z, y0))
}
```

`pgImporter` *Progenesis Importer Function*

Description

The function reads a CSV file which is exported using ProgenesisLCMS. The object is flexible when it comes to exported fields, ideally, everything is in the CSV. The delimiter is a semicolon (this setting is ProgenesisLCMS version AND system language setting sensitive). All the information is kept and accessible.

There is an option which allows you to switch the separator.

Usage

```
pgImporter(file, sep=';')
```

Arguments

file	A csv file.
sep	The field separator character. default vaule is set to a semicolon.

Details

This importer is dedicated to the commercial software ProgenesisLCMS. Nevertheless it is has a simple structure and can be adapted and used also for other maps.

todo: Pass an additonal argument to the pgImporter function which allows to define the separator.

Author(s)

Christian Panse, Hubert Rehauer, Jonas Grossmann 2012

---

pgLFQaov *iTRAQ two group analysis*

Description

The function performes an ANOVA for data sets like pgLFQ.

Usage

```
pgLFQaov(data, groups, names, idx, plot, FUN)
```

Arguments

data	a data set like pgLFq.
groups	a factor.
names	a vector of strings for the main attribute of each plot.
idx	a integer vector of indice to be processed.
plot	logical. If 'TRUE' (non default), a boxplot is drawn.
FUN	function for doing the data transformation, e.g. log, asinh.

Details

The methode performs an oav analysis using the R oav function. It returns a vector of the ANOVA "Pr(>F)" values in the same order.

Author(s)

Christian Panse, Jonas Grossmann 2012

Examples

```
data(pgLFQprot)

par(mfrow=c(4,3))
ANOVA<-pgLFQaov(pgLFQprot$"Normalized abundance",
    groups=as.factor(pgLFQprot$grouping),
    names=pgLFQprot$output$Accession,
    plot=TRUE)
ANOVA
```

---

pgLFQfeature *pgLFQfeature - A data set with a featuremap export ProgenesisLCMS*

Description

This data set contains a fraction of the most abundant features from 24 LC-MSMS runs measured on an Orbitrap-Velos. The data contains digested proteins from human HeLa cells infected with Shigella bacteria grown on a time course. It is structured in a way, that 6 biological replicates from 4 conditions are measured (Not_infected, Infected_1hr, Infected_2hr, Infected_3hr). The 24 LC-MSMS runs are aligned with each other and a so called mastermap (feature map) is generated where "normalized volumes" on MS1 are extracted for all features in the respective LC-MSMS runs. Some of the features are further annotated with peptide sequences and protein accessions (using the Mascot search algorithm and an ion score cutoff of 25).

We realized, depending on your language and keyboard setting, the separator for the FeatureData as well as for ProteinMeasurements are different (semicolon and commas are used depending on your setting). We assume, that semicolons are and the individual cells are escaped by ". If this differes, we have an option that can be switched in the pgImporter function.

More information on the commercial software can be found here: http://www.nonlinear.com/products/progenesis/lc-ms/overview/.

Format

A data object from ProgenesisImporter.R

Author(s)

Christian Panse, Jonas Grossmann 2012

References

doi:10.1016/j.jprot.2010.05.011

Examples

```
data(pgLFQfeature)
op<-par(mfrow=c(1,1),mar=c(18,18,4,1),cex=0.5)
samples<-names(pgLFQfeature$"Normalized abundance")
image(cor(asinh(pgLFQfeature$"Normalized abundance")),
    col=gray(seq(0,1,length=20)),
    main='pgLFQfeature correlation',
    axes=FALSE)

axis(1,at=seq(from=0, to=1,
    length.out=length(samples)),
    labels=samples, las=2)

axis(2,at=seq(from=0, to=1,
    length.out=length(samples)), labels=samples, las=2)
par(op)

# example assemble (and quantify) all proteins from peptides
# using intensities from the master(feature) map
# 1239 features
pgLFQfeature$peptideInfo$Sequence[1239]
pgLFQfeature$peptideInfo$Protein[1239]
# conflicts can be produced through:
# 1. shared peptides,
# 2. mapping to more than one peptide,
# 3. rank two above threshold
pgLFQfeature$output$Included[1239]
pgLFQfeature$"Normalized abundance"[1239,]

# tNpq
    par(mfrow=c(4,3), mar=c(1,1,4,1))
    for (i in 1:12)
        pgLFQtNpq(QuantitativeValue=pgLFQfeature$"Normalized abundance",
        peptide=pgLFQfeature$peptideInfo$Sequence,
        protein=pgLFQfeature$peptideInfo$Protein, N=i)
```

---

pgLFQprot *pgLFQprot - A data set with a featuremap export ProgenesisLCMS*

Description

This data set contain the top 250 most abundant Proteins identified from 24 LC-MSMS runs measured on an Orbitrap-Velos. The original data contains digested proteins from HeLa cells infected with Shigella bacteria grown in a time course. It is structured in a way, that 6 biological replicates from 4 conditions are measured (Not_infected, Infected_1hr, Infected_2hr, Infected_3hr). The 24 LC-MSMS runs are aligned with each other using ProgenesisLCMS. "Normalized volumes" on MS1 are extracted for all features in the respective LC-MSMS runs. The features with an annotation such as peptide sequences and protein accessions (using Mascot search algorithm) are assembled

to proteins. Features volumes are stacked (for non conflicting features) to generate the quantitative protein value for each LC-MSMS run. The csv file is an exported Protein data map.

We realized, depending on your language and keyboard setting, the separator for the FeatureData as well as for ProteinMeasurements are different (semicolon and commas are used depending on your setting). We assume, that semicolons are and the individual cells are escaped by ". If this differes, we have an option that can be switched in the pgImporter function.

More information on the commercial software can be found here: http://www.nonlinear.com/products/progenesis/lc-ms/overview/.

Format

A data object from ProgenesisImporter.R

Author(s)

Christian Panse, Jonas Grossmann 2012

References

doi:10.1016/j.jprot.2010.05.011

Examples

```
data(pgLFQprot)
op<-par(mfrow=c(1,1),mar=c(18,18,4,1),cex=0.5)
samples<-names(pgLFQprot$"Normalized abundance")
image(cor(asinh(pgLFQprot$"Normalized abundance")),
    main='pgLFQprot correlation',
    axes=FALSE,
    col=gray(seq(0,1,length=20)))
axis(1,at=seq(from=0, to=1,
    length.out=length(samples)), labels=samples, las=2)
axis(2,at=seq(from=0, to=1,
    length.out=length(samples)), labels=samples, las=2)
par(op)
```

---

pgLFQtNpq *Label Free Quantification using the top N peptide approach*

Description

This Function implements the recently emerged TopN strategy which uses only the top N intense features for calculating the proteinVolume. This approach should reveal a quantitative protein value, which should make the protein itself comparable within one condition. This allows to estimate protein stochiometries and simplifies modelling and calculations with copy numbers per cell.

Usage

```
pgLFQtNpq(QuantitativeValue,
peptide, protein, N=3, plot=TRUE, FUN=asinh)
```

Arguments

QuantitativeValue	a data set like pgLFQfeature$"Normalized abundance".
peptide	a vector of peptide sequences.
protein	a vector of protein information.
N	top N peptide flyers.
plot	logical. If 'TRUE' (non default), a boxplot is drawn.
FUN	function for doing the data transformation for the correlation matrix for the image plot, default transformation is asinh.

Details

The approach has first been described by Silva et al. in 2005 for Waters Q-tof instruments running in the MSe mode. Grossmann et al, showed in 2010 that this approach also works for more widely spread instruments such as Orbitrap-Velos or FTICR instruments from Thermo.

todo: additional columns (or additonal object) for 'protein names' and the total number of features assigned to protein in the master map. The length should be the same as for how many Ns chosen in the assembly method. Double check, if 'empty' protein names.. - basically - not assigned features.. are also in the list! - get rid of it.

Author(s)

Christian Panse, Jonas Grossmann 2012

References

doi:10.1074/mcp.M500230MCP200 doi:10.1016/j.jprot.2010.05.011

Examples

```
data(pgLFQfeature)
par(mfrow=c(2,4), mar=c(4,4,4,1))
for (i in c(1, 2, 3, 4)){
    tNpq<-pgLFQtNpq(QuantitativeValue=pgLFQfeature$"Normalized abundance",
        peptide=pgLFQfeature$peptideInfo$Sequence,
        protein=pgLFQfeature$peptideInfo$Protein,
        N=i)
}

for (i in c(1, 2, 3, 4)){
    tNpq<-pgLFQtNpq(QuantitativeValue=pgLFQfeature$"Normalized abundance",
        peptide=pgLFQfeature$peptideInfo$Sequence,
```

```
            protein=pgLFQfeature$peptideInfo$Protein,
            plot=FALSE,
            N=i)

        boxplot(t(tNpq), xlab='proteins', ylab='protein value')
    }
```

---

pressureProfile *NanoLC pressure profile*

Description

pressureProfile is a data frame with a variable number of cases (rows) and two variables (columns) named time(min), Pc(psi)

Details

The pressureProfile data set gives the flow profile in nL/min and the pressure profile in psi as a function of time, respectively, for 24 consecutive injections on an Eksigent NanoLC-Ultra 1D plus system.

The dataset consists of 24 HPLC pressure profiles of 140 min. Data was acquired with a Eksigent NanoLC-Ultra 1D plus system. Samples were separated with a 100 1E-6m ID. column of 150 mm lenght packed with C-18 AQ 200A beads (Dr. Maisch GmbH). The column was heated to a temperature of 50 grad C during the entire run. Acquisition queue was as follows:

- A block of three (3) samples with a gradient length of 140 min was allways followed a autocalibration run with a gradient length of 25 min.

- A total of seven (7) such blocks was acquired to cover the entire dataset.

Author(s)

Christian Trachsel 2012

Examples

```
data(pressureProfile)
ppp(pressureProfile[pressureProfile$filename=="F01",])

par(mfrow=c(1,1))
pps.data<-pps(pressureProfile, time=seq(1,140,by=5))
boxplot(Pc~time, data=pps.data,
    xlab='time [in minutes]', ylab='Pc(psi)')

library(lattice)
pps.data<-pps(pressureProfile, time=seq(25,40,by=5))
xyplot(Pc ~ as.factor(file) | paste("time =", as.character(time), "minutes"),
    panel = function(x, y){
        m<-sum(y)/length(y)
        m5<-(max(y)-min(y))*0.05
```

```
    panel.abline(h=c(m-m5,m,m+m5), col=rep("#ffcccc",3),lwd=c(2,4,2))
    panel.grid(h=-1, v=0)
    panel.xyplot(x, y)
},
ylab='Pc [psi]',
layout=c(1,4),
sub='The three read lines indicate avg plus min 5%.',
scales = list(x = list(rot = 45)),
data=pps.data)
```

`pressureProfilePlot` *Plotting pressure profile data from Eksigent LC pumps*

Description

plots the pressure profiles data on a scatter plot `Pc` versus `time` grouped by time range.

Usage

```
ppp(data)
```

Arguments

`data`	A data set like `pressureProfile`. The data must have an attribute `file`, `time`, and `Pc`.

Details

This is a useful function to generate an overview over a sequence of measurments run with the same LC gradient. It shows the behaviour of the LC pumps basically the pressures over the whole gradient are plotted to spot sources of problems.

Author(s)

Bernd Roschitzki, Christian Trachsel, Christian Panse 2012

Examples

```
data(pressureProfile)
ppp(pressureProfile[pressureProfile$filename=="F12",])
```

`pressureProfileSummary` *A misc function for finding NN time slots in pressure profile data.*

Description

The function computes a list of Pc values for the in time provided NN using findNN. It returns file, time, timediff and the Pc valyes as list.

Usage

```
pps(data, time)
```

Arguments

`data`	the data set to be plotted. It reqires the following attributes: `Pc`, `file`, `time`.
`time`	specifies the timeslots to be used.

Details

The function is useful for levelplots. It can be used to make a quality check over a sequence of experiments. At our site, it is used in conjugtion with Eksigent pumps. These devices store a text file at the instrument pc. These files are preprocessed as specified in the file structure.

Author(s)

Christian Panse 2012

Examples

```
library(lattice)
data(pressureProfile)

# do the pre processing
pps.data<-pps(pressureProfile, time=seq(25,40,by=5))

print(xyplot(Pc ~ as.factor(file) | paste("time =",
    as.character(time), "minutes"),
    panel = function(x, y){
        m<-sum(y)/length(y)
        m5<-(max(y)-min(y))*0.05
        panel.abline(h=c(m-m5,m,m+m5),
            col=rep("#ffcccc",3),lwd=c(1,2,1))
        panel.grid(h=-1, v=0)
        panel.xyplot(x, y)
    },
    ylab='Pc [psi]',
    layout=c(1,4),
```

```
sub='The three read lines indicate avg plus min 5%.',
scales = list(x = list(rot = 45)),
data=pps.data))
```

---

psm *Compute a matching between a peptide sequence and a MS2 spectrum*

Description

The function computes a matching between a given peptide sequence and a given tandem mass spectrum (MS2). psm determines for each fragment ion mass peak the smallest mass error to a peak in the theretical spectrum from the peptide sequence. If the mass error is below the given fragmentIonError the match is considered a hit. psm returns a list of computed fragment ions and a vector of the mass error (Da and ppm). psm uses a generic ANSI-C function to determines the nearest mass peak of array of double values. If the plot is set to TRUE an error plot is drawn.

The function psm reqires the arguments sequence and spec. All other arguments are optional.

Usage

```
psm(sequence,
    spec,
    FUN=defaultIon,
    plot=TRUE,
    fi=fragmentIon(sequence, FUN=FUN)[[1]],
    fragmentIonError)
```

Arguments

sequence	peptide sequence encoded as character sequence using the 20 amino acid letters.
spec	MS2 which is a R list having a sorted mZ vector and an intensity vector of the same size.
FUN	this function is passed to the fragmentIon function. the function to be applied to compute further ions. If no function is assigned fragmentIon will use defaultIon.
plot	boolean if the error plot function is to be called.
fi	fragment ion table, if not specified fragmentIon(sequence, FUN=FUN)[[1]] is called.
fragmentIonError	fragment ion error cut-off. default is 0.6 Da.

Details

This function can be very useful to make assignements to spectra. Moreover it is used for validation or clairifying ambiguities between different sequences assigned to the same spectrum. Additinally it can be used to generate spectral libraries.

Value

returns a psm match.

Author(s)

Christian Panse 2007, 2008, 2009, 2010, 2012, 2017

See Also

peakplot and fragmentIon

Examples

```
spec <- list(scans=1138,
    title="178: (rt=22.3807) [20080816_23_fetuin_160.RAW]",
    rtinseconds=1342.8402,
    charge=2,
    mZ=c(195.139940, 221.211970, 239.251780, 290.221750,
    316.300770, 333.300050, 352.258420, 448.384360, 466.348830,
    496.207570, 509.565910, 538.458310, 547.253380, 556.173940,
    560.358050, 569.122080, 594.435500, 689.536940, 707.624790,
    803.509240, 804.528220, 822.528020, 891.631250, 909.544400,
    916.631600, 973.702160, 990.594520, 999.430580, 1008.583600,
    1017.692500, 1027.605900),
    intensity=c(931.8, 322.5, 5045, 733.9, 588.8, 9186, 604.6,
    1593, 531.8, 520.4, 976.4, 410.5, 2756, 2279, 5819, 2.679e+05,
    1267, 1542, 979.2, 9577, 3283, 9441, 1520, 1310, 1.8e+04,
    587.5, 2685, 671.7, 3734, 8266, 3309)
    )

m <- psm('HTLNQIDSVK', spec,plot=TRUE)
hist(m$mZ.Da.error)
hist(m$mZ.ppm.error)
```

PTM_MarkerFinder *PTM MarkerFinder*

Description

PTM_MarkerFinder is a fucntion to identify and validate spectra from peptides carrying post-translational modifications.

Usage

```
PTM_MarkerFinder(data,
    modification,
    modificationName,
    mZmarkerIons,
```

```
    minNumberIons=2,
    itol_ppm=10,
    minMarkerIntensityRatio=5,
    mgfFilename=-1,
    PEAKPLOT=TRUE
    )

findMz(data, mZmarkerIons, itol_ppm = 10, minNumberIons = 2, minMarkerIntensityRatio = 10)

## S3 method for class 'psmSet'
findMz(data, mZmarkerIons, itol_ppm, minNumberIons, minMarkerIntensityRatio)

## S3 method for class 'mascot'
findMz(data, mZmarkerIons, itol_ppm = 10, minNumberIons = 2, minMarkerIntensityRatio = 10)
```

Arguments

`data`	A list of spectra where each list element contains a list of `mZ`, `intensity` vectors. This object can be received by using the `mascotDat2RData.pl` perl script.
`modification`	A double vector containing the mono mass PTMs.
`modificationName`	A character vector containing the Name of the PTMs. This is the string which will show up in the peptide sequence in square brackets.
`mZmarkerIons`	The m/z patterns which should be searched.
`minNumberIons`	Minimal number of marker ions to be found for further analysis.
`itol_ppm`	The ion tolerance of the marker ions in ppm. default is set to 10ppm.
`minMarkerIntensityRatio`	The marker ions intencity percentage compared to the sum of all peak intensities.
`mgfFilename`	if a mgf(mascot generic file) filename is given, the function writes an mgf file containing only the ms2 having the `mZmarkerIons`.
`PEAKPLOT`	If this boolean is `FALSE` the `peakplot` function is not used.

Details

The function screens MS2 spectra for the presence of fragment ions specific Post Translational Modifications (PTMs). The function requires an R-object containing the mass spectrometric measurement, the peptide assignments, potential modification information, and a list of marker ions. The R-object can be retrieved right out of the Mascot Server search result dat files using a perl script `mascotDat2RData.pl` which is included in the package's `exec/` directory.

The functions iterates over each spectrum of the mass spectrometric measurement and searches for the as input provided marker ions. If a certain number of marker ions (default is, that two marker ions are required) are found and the maker ion intensity ratio is higher than a given threshold, the tandem mass spectrum is considered as HCD scan type and the corresponding ion series are drawn using the protViz:peakplot methode. Furthermore the function is searching for the corresponding ETD scan having the same peptide mass by screening the succeeding scans for ETD spectra. If such

a spectrum is found the peptide spectrum assignment containing the c, z, and y ions is drawn. Note that the PTM MarkerFinder expects the ETD scan right after the HCD scan. If the MS protocol changes the PTM MarkerFinder methode has to be adapted.

For each HCD scan PTM MarkerFinder plots both HCD and ETD scans, a ppm error versus maker ion m/z scatter plot, a intensity versus marker ion m/z plot, and two pie charts where the relative and absolute maker ion intensity are shown.

As a summery report the function returns a table containing the following column attributes: "scans", "mZ", "markerIonMZ", "markerIonIntensity", "markerIonMzError", "markerIonPpmError", and "query" which can be used for statistics.

Furthermore, if `mgfFilename` is defined, a Mascot Generic File (MGF) is created containig the HCD scans (having the marker ions) and the corresponding ETD scans.

Author(s)

Paolo Nanni, Peter Gehrig, Christian Panse 2011-2013;

References

- Nanni P, Panse C, Gehrig P, Mueller S, Grossmann J, Schlapbach R.(2013), PTM MarkerFinder, a software tool to detect and validate spectra from peptides carrying post-translational modifications. Proteomics. 2013 Aug;13(15):2251-5. doi:10.1002/pmic.201300036.
- `ADP_Ribose <- c(136.0618, 250.0935, 348.0704, 428.0367)` marker ions have been used in: Bilan V, Leutert M, Nanni P, Panse C, Hottiger MO. Combining Higher-Energy Collision Dissociation and Electron-Transfer/Higher-Energy Collision Dissociation Fragmentation in a Product-Dependent Manner Confidently Assigns Proteomewide ADP-Ribose Acceptor Sites, Anal. Chem., 2017, 89 (3), pp 1523-1530 doi:10.1021/acs.analchem.6b03365.

See Also

`peakplot`

Examples

```
 # some marker ions

 Glykan_MarkerIons <- c(109.02841, 127.03897, 145.04954, 163.06010, 325.11292)

HexNAc_MarkerIons <- c(126.05495, 138.05495, 144.06552, 168.06552, 186.07608, 204.08665)

 # DOI: 10.1021/acs.analchem.6b03365
 # Anal Chem 2017 Feb 13;89(3):1523-1530. Epub 2017 Jan 13.
 ADP_Ribose <- c(136.0618, 250.0935, 348.0704, 428.0367)

 data(HexNAc)

 # prepare modification
 ptm.0 <- cbind(AA="-",
     mono=0.0, avg=0.0, desc="unmodified", unimodAccID=NA)
```

```
    ptm.1 <- cbind(AA='N',
          mono=317.122300, avg=NA, desc="HexNAc",
                  unimodAccID=2)

    ptm.2 <- cbind(AA='M',
            mono=147.035400, avg=NA, desc="Oxidation",
                    unimodAccID=1)

    m <- as.data.frame(rbind(ptm.0, ptm.1, ptm.2), stringsAsFactors = TRUE)

    S <- PTM_MarkerFinder(data=HexNAc, modification=m$mono,
        modificationName=m$desc,
        minMarkerIntensityRatio=3,
        itol_ppm=20,
        mZmarkerIons=HexNAc_MarkerIons)


    boxplot(markerIonIntensity ~ markerIonMZ,
        data=S,
        log='y',
        main='Summary plot: boxplot of marker ion intensities from all pPTM spectra',
        xlab='markerIon m/z',
        ylab='log10 based marker ion intensity')

    # export
    w <- reshape(S[,c(1,7,3,4)],
        direction='wide',
        timevar="markerIonMZ",
idvar=c('scans','query'))

    write.table(w,
        file=file.path(tempdir(), "HexNAc_PTM_markerFinder.csv"),
        sep=',',
row.names=FALSE,
col.names=TRUE,
quote=FALSE)
```

`PTM_MarkerFinder_util` *PTM MarkerFinder util plot*

Description

`PTM_MarkerFinder_util` is a utility function for `PTM_MarkerFinder`.

Usage

```
PTM_MarkerFinder_util(dataFileName,
    mZmarkerIons,
    minMarkerIntensityRatio,
    minNumberIons,
```

```
itol_ppm,
write_csv)
```

Arguments

dataFileName	RData file name without 'RData' ending.
mZmarkerIons	A double vector of the m/z patterns which should be searched.
minMarkerIntensityRatio	The marker ions intencity percentage compared to the sum of all peak intensities.
minNumberIons	Minimal number of marker ions to be found for further analysis.
itol_ppm	The ion tolerance of the marker ions in ppm. default is set to 10ppm.
write_csv	boolean.

Details

The function plots summeries in form of boxplots pie charts and scatter plots of the found marker ions.

Author(s)

Paolo Nanni, Christian Panse 2012-2013;

---

ssrc *Sequence Specific Retention Calculator*

Description

This function returns as output one hydrophobicity value for a given sequence of amino acids (tryptic peptide) which can be used to predict the retention times. The calculation is based on the method described in PMID:15238601.

Usage

```
ssrc(x, H = list())
```

Arguments

x	sequence of amino acids, e.g., x="ELIVSK"
H	A list of retention coefficients. The default is set to the values of PMID:15238601 table II column 2(Rc values).

Author(s)

Christian Panse, Christian Trachsel 2015

References

Krokhin, O. V. et al. An improved model for prediction of retention times of tryptic peptides in ion pair reversed-phase HPLC: its application to protein peptide mapping by off-line HPLC-MALDI MS. Mol. Cell Proteomics 3, 908-919 (2004). doi:10.1074/mcp.M400031MCP200

See Also

- iRTpeptides

Examples

```
# example of table iv [PMID:15238601]
lapply(c("SCHTAVGR", "SCHTGLGR", "EDLIAYLK"), ssrc)

plot(sapply(as.character(iRTpeptides$peptide), ssrc) ~  iRTpeptides$rt)
```

Index

∗ **AA**
    Fasta, 12
∗ **FASTA**
    Fasta, 12
∗ **centroid**
    centroid, 8
∗ **datasets**
    ADPR, 5
    fetuinLFQ, 13
    HexNAc, 21
    iTRAQ, 23
    msms, 29
    pgLFQfeature, 35
    pgLFQprot, 36
    pressureProfile, 39
∗ **profile**
    centroid, 8

AA, 3
aa2mass, 4
ADPR, 5
apex *(*fetuinLFQ*)*, 13
as.data.frame.fragmentIonSet *(*fragmentIon*)*, 17
as.data.frame.mascot *(*mascot*)*, 26
as.data.frame.psmSet *(*psm*)*, 42
as.psm.mascot_query *(*mascot*)*, 26
as.psmSet, 5
assignPlatePosition, 6
averagine, 7

blockRandom, 8
bymatrix *(*fragmentIon*)*, 17

centroid, 8

de_novo, 11
defaultIon *(*fragmentIon*)*, 17
deisotoper, 10

empai *(*fetuinLFQ*)*, 13

FASTA *(*Fasta*)*, 12
Fasta, 12
fasta *(*Fasta*)*, 12
fetuinLFQ, 13
findInterval, *16*
findMz *(*PTM_MarkerFinder*)*, 43
findNN, 15
findNN_ *(*findNN*)*, 15
fragmentIon, 17, *32*, *42*, *43*

genMod, 20

HexNAc, 21
hydrophobicity *(*ssrc*)*, 47

insertSamples, 22
ionseries *(*fragmentIon*)*, 17
iRT *(*iRTpeptides*)*, 23
irt *(*iRTpeptides*)*, 23
iRTpeptides, 23, *48*
is.mascot *(*mascot*)*, 26
is.mascot_query *(*mascot*)*, 26
is.psm *(*psm*)*, 42
is.psmSet *(*as.psmSet*)*, 5
iTRAQ, 23
iTRAQ2GroupAnalysis, 24

lcmsmap, 25, *26*
lcmsoverview *(*lcmsmap*)*, 25
lower_bound_ *(*findNN*)*, 15

mascot, 26, *32*
massDeviationPlot, 28
mdp *(*massDeviationPlot*)*, 28
msms, 29

NN *(*findNN*)*, 15

parentIonMass, 29
peaklist *(*peakplot*)*, 31
peaklistSet *(*peakplot*)*, 31

peakplot, *18*, *26, 27*, 31, *32*, *43*, *45*
pep3d *(*lcmsmap*)*, 25
pepmass *(*parentIonMass*)*, 29
pgImporter, 33
pgLFQaov, 34
pgLFQfeature, 35
pgLFQprot, 36
pgLFQtNpq, 37
pim *(*parentIonMass*)*, 29
plot.mascot *(*mascot*)*, 26
plot.mascot_query *(*mascot*)*, 26
plot.psm *(*peakplot*)*, 31
plot.psmSet *(*lcmsmap*)*, 25
ppp *(*pressureProfilePlot*)*, 40
pps *(*pressureProfileSummary*)*, 41
pPTM *(*PTM_MarkerFinder*)*, 43
pressureProfile, 39
pressureProfilePlot, 40
pressureProfileSummary, 41
profile *(*centroid*)*, 8
protViz *(*peakplot*)*, 31
psm, *32*, *42*, 42
PTM_MarkerFinder, *32*, 43
PTM_MarkerFinder_util, 46

ssrc, 47
summary.cometdecoy *(*mascot*)*, 26
summary.mascot, *27*
summary.mascot *(*mascot*)*, 26
summary.psmSet *(*psm*)*, 42

t3pq *(*fetuinLFQ*)*, 13
tNpq *(*pgLFQtNpq*)*, 37

xmlParse, *27*
xmlToList, *27*