# PSMatch: an R/Bioconductor package to explore proteomics identification data.

## Running title: The PSMatch Bioconductor Package

Guillaume Deflandre[1], Sebastien Gibb[2], Laurent Gatto[1*]

[1]de Duve Institute, UCLouvain, Brussels, Belgium.
[2]University Medicine Greifswald, Greifswald, Germany.

*Corresponding author(s). E-mail(s): laurent.gatto@uclouvain.be;

**Abstract**

We present the R/Bioconductor *PSMatch* package, that helps proteomics practitioners to load, manage and handle peptide spectrum matches. It provides functions to model peptide-protein relations, visualise these as graphs and make informed decisions about shared peptide filtering. The package also provides functions to calculate fragments and, in conjunction with the *Spectra* package, annotate and visualise MS2 scans.

**Keywords:** Mass spectrometry, Proteomics, Identification, PTM, Visalisation, Bioconductor, R, Reproducible Research, Software

## 1 Introduction

Mass spectrometry (MS) is the current state-of-the-art technology to comprehensively study complex proteomes. It is compatible with diverse input sample types and provides a comprehensive proteome coverage, without the need for prior knowledge on the proteins present in a sample. The processing of MS-based proteomics data entails

two distinct, albeit related steps, namely the identification of peptides and their quantitation. Both of these are then combined to produce the quantitative data matrix that is further processed, analysed, and interpreted. Both are also critical to guarantee biologically accurate interpretation of the proteomics data. However, unlike for quantitative data that beneftis from a lot of attention and exploration and analysis tools [1–4], there are relatively few tools and methods to explore identification data [5–8] that tend to be accepted as is without undergoing further exploratory data analysis.

Here, we present *PSMatch* (Fig. 1), an R/Bioconductor [9, 10] package designed to handle identification data in the form of peptide-spectrum matches (PSMs), and offering functionalities to streamline exploration and visualisation of PSM data. It provides functions to load PSM data from *mzIdentML* (*mzID*) [11] or tabular files, generate theoretical fragment ions, model peptide-protein relations and facilitate various visualisations. Recent advancements provide functionalities to study post-translational modifications (PTMs). As the proportion of unidentified spectra in MS-based experiments typically range from 50% to 90% [12, 13] dedicated and flexible tools built for handling the actual identified spectra are essential.

## 2 Materials

To demonstrate the usefulness of the *PSMatch* package, a publicly available phosphoproteomic dataset [14] was downloaded and a database search was performed using the search engine Sage [15]. The *mzML* files were downloaded from the PRIDE [16] database (experiment PXD039419) and the `sage-results.tsv` file was generated with Sage *v.0.14.6*. The necessary files to reproduce the code in the paper are available on Zenodo [17]. The versions of the packages used to produce this document are listed in the session information below. Note that *PSMatch* is not dependent on Sage or any other search engine. It can handle any identification data stored in *mzIdentML* or tabular files from any search engine.

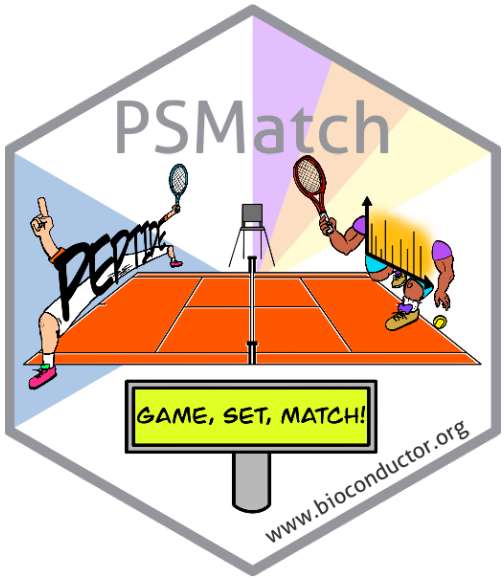


**Fig. 1** The *PSMatch* package sticker

## 2.1 Installation

*PSMatch* is an open source R/Bioconductor package, openly developed on GitHub. It can be installed from the Bioconductor project [10] using the `BiocManager` package.

```
if (!require("BiocManager"))
    install.packages("BiocManager")
BiocManager::install("PSMatch")
```

Once installed, the package can be loaded with

```
library(PSMatch)
```

## 2.2 Documentation

This tutorial showcases how *PSMatch* allows to handle identification data. For a more complete overview of *PSMatch*'s functionalities, please refer to the appropriate vignettes for the respective subjects. The vignettes are available online from the package's webpage or locally in R. To list all available *PSMatch* vignettes, we can use:

```
vignette(package = "PSMatch")
```

```
Vignettes in package 'PSMatch':

Fragments            MS2 fragment ions (source, html)
AdjacencyMatrix      Understanding protein groups with
                     adjacency matrices (source, html)
PSM                  Working with PSM data (source, html)
```

To open the vignette focusing on 'MS2 fragment ions', we add its name as shown in the next code chunk:

```
vignette("Fragments", package = "PSMatch")
```

## Session information

A complete list and version of R and packages used to run this complete analysis and produce the results is provided below. **Note 1** provides important details about the R and Bioconductor versions compatibility.

- R version 4.5.0 (2025-04-11), `x86_64-pc-linux-gnu`

- Running under: `Ubuntu 24.04.2 LTS`
- Matrix products: default
- BLAS: `/opt/R-4.5/lib/R/lib/libRblas.so`
- LAPACK: `/opt/R-4.5/lib/R/lib/libRlapack.so` ; LAPACK version3.12.1
- Base packages: base, datasets, graphics, grDevices, methods, stats, stats4, utils
- Other packages: AnnotationFilter 1.32.0, AnnotationHub 3.16.0, BiocFileCache 2.16.0, BiocGenerics 0.54.0, BiocParallel 1.42.0, BiocStyle 2.36.0, CompoundDb 1.12.1, dbplyr 2.5.0, generics 0.1.4, ggplot2 3.5.2, knitr 1.50, MsBackendSql 1.8.0, MsDataHub 1.8.0, pheatmap 1.0.12, RColorBrewer 1.1-3, reticulate 1.42.0, RSQLite 2.3.11, S4Vectors 0.46.0, Spectra 1.18.2, SpectriPy 0.99.5
- Loaded via a namespace (and not attached): AnnotationDbi 1.70.0, base64enc 0.1-3, Biobase 2.68.0, BiocManager 1.30.25, BiocVersion 3.21.1, Biostrings 2.76.0, bit 4.6.0, bit64 4.6.0-1, bitops 1.0-9, blob 1.2.4, cachem 1.1.0, ChemmineR 3.60.0, cli 3.6.5, clue 0.3-66, cluster 2.1.8.1, codetools 0.2-20, colorspace 2.1-1, compiler 4.5.0, crayon 1.5.3, curl 6.2.3, data.table 1.17.4, DBI 1.2.3, digest 0.6.37, dplyr 1.1.4, DT 0.33, evaluate 1.0.3, ExperimentHub 2.16.0, farver 2.1.2, fastmap 1.2.0, fastmatch 1.1-6, filelock 1.0.3, fs 1.6.6, GenomeInfoDb 1.44.0, GenomeInfoDbData 1.2.14, GenomicRanges 1.60.0, glue 1.8.0, grid 4.5.0, gridExtra 2.3, gtable 0.3.6, hms 1.1.3, htmltools 0.5.8.1, htmlwidgets 1.6.4, httr 1.4.7, IRanges 2.42.0, jsonlite 2.0.0, KEGGREST 1.48.0, labeling 0.4.3, lattice 0.22-6, lazyeval 0.2.2, lifecycle 1.0.4, magrittr 2.0.3, MASS 7.3-65, Matrix 1.7-3, memoise 2.0.1, MetaboCoreUtils 1.16.1, mime 0.13, MsCoreUtils 1.20.0, mzR 2.42.0, ncdf4 1.24, parallel 4.5.0, pillar 1.10.2, pkgconfig 2.0.3, png 0.1-8, prettyunits 1.2.0, progress 1.2.3, ProtGenerics 1.40.0, purrr 1.0.4, R6 2.6.1, rappdirs 0.3.3, Rcpp 1.1.0, RCurl 1.98-1.17, rjson 0.2.23, rlang 1.1.6, rmarkdown 2.29, rsvg 2.6.2, scales 1.4.0, stringi 1.8.7, tibble 3.2.1,

tidyselect 1.2.1, tinytex 0.57, tools 4.5.0, UCSC.utils 1.4.0, vctrs 0.6.5, withr 3.0.2, xfun 0.52, xml2 1.3.8, XVector 0.48.0, yaml 2.3.10

# 3 Methods

## 3.1 Creating `PSM` objects

There is a plethora of search engines available with each their own resulting identification file format with different variable names. Most of them provide variables such as a matched spectrum identifier, a PSM rank, a search engine score, a false discovery rate, a label indicating whether the PSM comes from the decoy or target peptide, a peptide sequence as well as the protein or proteins the matched peptide sequence stems from. These variables can be defined when creating the `PSM` object that holds the identification results. These variables are then used to filter and analyse the identification data. The other variables are still stored within the object and can be accessed in the same manner as one would be with a `data.frame`.

Let's first load the identification tabular data into a `data.frame` and look at the variable names to identify how these are encoded in the Sage output files. The code below is general and applicable to any search engine, but see **Note 2** on how to simplify importing data from Sage specifically.

```
sage_results <- read.delim("./data/PXD039419-sage-results.tsv")
colnames(sage_results)
```

```
 [1] "psm_id"                "peptide"
 [3] "proteins"              "num_proteins"
 [5] "filename"              "scannr"
 [7] "rank"                  "label"
```

```
 [9] "expmass"                 "calcmass"
[11] "charge"                  "peptide_len"
[13] "missed_cleavages"        "semi_enzymatic"
[15] "isotope_error"           "precursor_ppm"
[17] "fragment_ppm"            "hyperscore"
[19] "delta_next"              "delta_best"
[21] "rt"                      "aligned_rt"
[23] "predicted_rt"            "delta_rt_model"
[25] "ion_mobility"            "predicted_mobility"
[27] "delta_mobility"          "matched_peaks"
[29] "longest_b"               "longest_y"
[31] "longest_y_pct"           "matched_intensity_pct"
[33] "scored_candidates"       "poisson"
[35] "sage_discriminant_score" "posterior_error"
[37] "spectrum_q"              "peptide_q"
[39] "protein_q"               "ms2_intensity"
```

We first transform decoy labels into logical (boolean) values, and then construct the `PSM` object from the `data.frame`, setting the functions' arguments with the apporpriate column names.

```
sage_results[, "label"] <- sage_results[, "label"] < 0


psms <- PSM(
    sage_results,
    spectrum = "scannr", ## Scan index
    peptide = "peptide", ## Peptide sequence
```

```
    protein = "proteins", ## Protein(s) which the peptide stems from
    decoy = "label", ## Is the PSM a target or a decoy ?
    rank = "rank", ## Rank of the PSM
    score = "sage_discriminant_score", ## Score given by search engine
    fdr = "spectrum_q") ## Calculated q-value or FDR for that PSM

psms
```

```
PSM with 562183 rows and 40 columns.
Spectra: 58591 unique
  db: 444298 target, 117885 decoy
  ranks: 1:215239 2:185573 3:161371
names(40): psm_id peptide ... protein_q ms2_intensity
```

The `PSM()` constructor could also have taken an *mzIdentML* file, rather than a `data.frame` to create the `PSM` object.

The `psmVariables` of the object store the original variable names, that will be used later on during the filtering step.

```
psmVariables(psms)
```

```
                 spectrum                   peptide
                "scannr"                 "peptide"
                  protein                     decoy
                "proteins"                   "label"
                     rank                     score
                   "rank" "sage_discriminant_score"
```

```
                    fdr
"spectrum_q"
```

Before proceeding with the PSM data exploration, let's visualise and confirm the separation between target and decoy peptide scores for the 9 original raw data acquisitions (defined by their *mzML* file names) (Fig. 2).

```
library(ggplot2)
ggplot(psms, aes(x = sage_discriminant_score, colour = label)) +
    geom_density() +
    facet_wrap(~ filename) +
    theme(strip.text.x = element_text(size = 8))
```

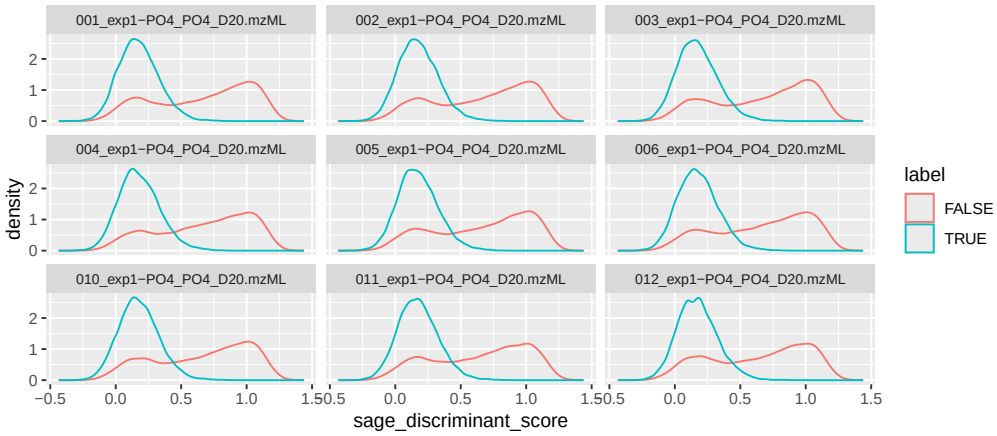


**Fig. 2** Target and decoy PSM score distributions.

## 3.2 Filtering data

We can now filter the PSMs to retain only the targets of rank 1 that have been confidently identified by using the dedicated functions.

```
psms <- psms |>
    filterPsmDecoy() |>
    filterPsmFdr(FDR = 0.01) |>
    filterPsmRank()
```

```
Removed 117885 decoy hits.
```

```
Removed 148826 PSMs with FDR > 0.01.
```

```
Removed 155088 PSMs with rank > 1.
```

The `filterPSMs()` function is a wrapper around the functions above, including also the `filterPsmShared()` function that keeps only unique peptides.

## 3.3 Exploring peptide-protein relations

One of the most important aspects of identification data lies in how proteins and peptides are related. Some identified peptides stem from multiple proteins (called *shared peptides*) while others are unique to a single protein or isoform (called *unique* or *proteotypic* peptides). The `describePeptides()` and `describeProteins()` functions provide useful summaries of shared and unique peptides as well as protein and protein groups.

```
describePeptides(psms)
```

```
25344 peptides composed of

 unique peptides: 9177
```

```
 shared peptides (among protein):

  7649(2) 3983(3) 1912(4) 1099(5) 515(6) 362(7) 214(8)
  148(9) 57(10) 41(11) 39(12) 13(13) 6(14) 6(15) 35(16)
  15(17) 8(18) 5(19) 5(20) 14(21) 10(22) 9(23) 6(24)
  12(32) 1(47) 1(68) 1(113) 1(243)
```

```
describeProteins(psms)
```

```
10581 proteins composed of

 only unique peptides: 1596

 only shared peptides: 8437

 unique and shared peptides: 548
```

One way to store the peptide-protein relations is through connected components. They represent the relation between proteins and peptides and how shared and unique peptides define groups of proteins. Below, we see that the constituents of the protein groups, as defined in the the `proteins` variable, are separated by a semi-column, which later we set to split the groups and create the connected components.

```
head(psms$proteins)
```

```
[1] "sp|Q9H307|PININ_HUMAN"
[2] "sp|P02545-3|LMNA_HUMAN;sp|P02545-4|LMNA_HUMAN;sp|P02545-5|LMNA_HUMAN;sp|P02545|LMNA_HUMAN"
```

```
[3] "sp|Q13586-2|STIM1_HUMAN;sp|Q13586|STIM1_HUMAN"
[4] "sp|Q7Z417|NUFP2_HUMAN"
[5] "sp|Q7Z2T5|TRM1L_HUMAN"
[6] "sp|Q9NXV6|CARF_HUMAN"
```

```
cc <- ConnectedComponents(psms, split = ";")
```

The connected component summary shows the total number of individual proteins and components in the data, and the respective number of single proteins defined by a single unique peptide (`[1 x 1]`), protein groups defined by a single shared peptide (`[1 x n]`), single proteins defined by multiple unique peptides (`[n x 1]`) and protein groups defined by multiple peptides (`[n x n]`).

```
cc
```

```
An instance of class ConnectedComponents
 Number of proteins: 10581
 Number of components: 4884
 Number of components [peptide x proteins]:
  1476[1 x 1] 411[1 x n] 1127[n x 1] 1870[n x n]
```

Individual peptide-by-protein connected components can be extracted with `connectedComponents()` function, specifying the index of the component.

```
m <- connectedComponents(cc, 22)
m
```

```
3 x 3 sparse Matrix of class "dgCMatrix"
                          sp|A2AJT9-2|BCLA3_HUMAN
SQKLPDVKPS[+79.966]PINLR               4.6364287
RGS[+79.966]EDFETR                     0.7867669
GYS[+79.966]PGRGDSNRR                  1.0660958
                          sp|A2AJT9-3|BCLA3_HUMAN
SQKLPDVKPS[+79.966]PINLR               4.6364287
RGS[+79.966]EDFETR                     0.7867669
GYS[+79.966]PGRGDSNRR                  1.0660958
                          sp|A2AJT9|BCLA3_HUMAN
SQKLPDVKPS[+79.966]PINLR             4.6364287
RGS[+79.966]EDFETR                   0.7867669
GYS[+79.966]PGRGDSNRR                1.0660958
```

The values in the connected component matrix represent the sum of scores given by the search engine. Some PSMs may have identical peptide sequences but have different charges or modifications sites, when these aren't explicitly encoded in the sequence. Such instances increase the confidence of the protein that peptide stems from, which is why the scores are summed.

Individual components can be visualised using the `plotAdjacencyMatrix()` function. Fig. 3 shows the relation between 3 peptides and 3 proteins. Proteins are represented as coloured squares and peptides by circles.

```
plotAdjacencyMatrix(m)
```

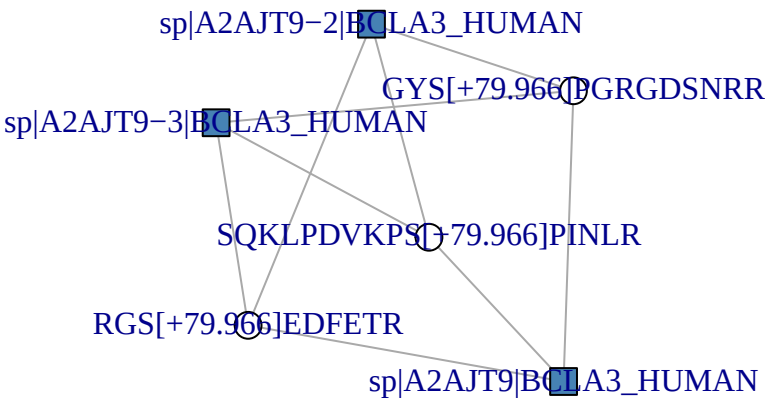


**Fig. 3** Network representing the relationships between peptides and proteins within a protein group.

In this example, all three peptides peptides are shared among the three isoforms of the BCLAF3 gene.

One way to select relevant connected components is to select those composed by more than one peptide (more than 1 row) and more than one protein (more that 1 column), i.e. those annotated as `[n x n]` in the textual summary of the object.

```
head(which(nrows(cc) > 1 & ncols(cc) > 1))
```

```
[1]  5  6  7 13 18 19
```

```
connectedComponents(cc, 7)
```

```
4 x 2 sparse Matrix of class "dgCMatrix"
                                          sp|A0A8I5KQE6|RPSA2_HUMAN
EHPWEVM[+15.995]PDLYFYRDPEEIEKEEQAAAEK                    2.9922049
FTPGTFTNQIQAAFREPR                                        .
EHPWEVMPDLYFYRDPEEIEKEEQAAAEK                             0.8481142
```

```
LLVVTDPR                                              2.7430056
                                          sp|P08865|RSSA_HUMAN
EHPWEVM[+15.995]PDLYFYRDPEEIEKEEQAAAEK             2.9922049
FTPGTFTNQIQAAFREPR                                    1.4500293
EHPWEVMPDLYFYRDPEEIEKEEQAAAEK                         0.8481142
LLVVTDPR                                              2.7430056
```

The vignette 'Understanding protein groups with adjacency matrices' from the *PSMatch* package provides further details on prioritizing interesting connected components for further exploration.

### 3.4 Calculating fragments masses

*PSMatch* also offers functions to compute theoretical fragments, including post-translational modifications. These can be set as fixed or variable modifications. It is also possible to set the maximum amount of modifications per peptide. Below, we define a single variable phosphorylation on threonines and compute the masses of $b$ and $y$ fragment ions for a short illustrative peptide.

```
calculateFragments("ATK",
                   fixed_modifications = NULL,
                   variable_modifications = c(T = 79.966),
                   type = c("b", "y"))
```

```
Fixed modifications used: None
Variable modifications used: T=79.966

          mz ion type pos z seq     peptide
```

```
1   72.04439  b1    b   1 1   A          ATK
2  173.09207  b2    b   2 1  AT          ATK
3  147.11280  y1    y   1 1   K          ATK
4  248.16048  y2    y   2 1  TK          ATK
5  129.10224 y1_   y_   1 1   K          ATK
6  230.14992 y2_   y_   2 1  TK          ATK
7   72.04439  b1    b   1 1   A AT[79.966]K
8  253.05807  b2    b   2 1  AT AT[79.966]K
9  147.11280  y1    y   1 1   K AT[79.966]K
10 328.12648  y2    y   2 1  TK AT[79.966]K
11 129.10224 y1_   y_   1 1   K AT[79.966]K
12 310.11592 y2_   y_   2 1  TK AT[79.966]K
```

### 3.5 Visualising annotated spectra

The theoretical fragments truly shine when combined with the observed peaks from the experimental spectra. The spectra can be loaded using the R/Bioconductor *Spectra* package [18] and the PSMs can be joined to the spectra using a common identifier. The *Spectra* package can also be installed using `BiocManager::install()`.

```
BiocManager::install("Spectra")
library(Spectra)
```

Let's first load the spectra from the *.mzML* file containing the raw MS data and create an object of class `Spectra`.

```
sp <- Spectra("./data/001_exp1-PO4_PO4_D20.mzML")
sp
```

```
MSn data (Spectra) with 62152 spectra in a MsBackendMzR backend:
        msLevel     rtime scanIndex
      <integer> <numeric> <integer>
1             1 0.0623463         1
2             1 0.4938655         2
3             2 0.5671415         3
4             1 0.9047552         4
5             1 1.3250339         5
...         ...       ...       ...
62148         2   6598.38     62148
62149         1   6598.72     62149
62150         1   6599.11     62150
62151         1   6599.50     62151
62152         1   6599.89     62152
 ... 34 more variables/columns.

file(s):
001_exp1-PO4_PO4_D20.mzML
```

To match the PSMs from the 9 acquisitions to the scans from the first acquisition loaded in `sp`, we need to define a common identifier. Below, we generate such unique primary keys by combining the *mzML* filenames and the spectrum identifiers. If identification results from a single file were to be matched to a single *mzML* spectra, the spectrum identifiers would suffice, but we prefer to describe the most general case here.

```
## Create the common identifier for the spectra
head(sp$pkey <- paste0(basename(sp$dataOrigin),
                       sub("^.+scan=", "::", sp$spectrumId)))
```

```
[1] "001_exp1-PO4_PO4_D20.mzML::1"
[2] "001_exp1-PO4_PO4_D20.mzML::2"
[3] "001_exp1-PO4_PO4_D20.mzML::3"
[4] "001_exp1-PO4_PO4_D20.mzML::4"
[5] "001_exp1-PO4_PO4_D20.mzML::5"
[6] "001_exp1-PO4_PO4_D20.mzML::6"
```

```
## Create the common identifier for the PSMs
head(psms$pkey <- paste0(basename(psms$filename),
                         sub("^.+scan=", "::", psms$scannr)))
```

```
[1] "003_exp1-PO4_PO4_D20.mzML::32424"
[2] "012_exp1-PO4_PO4_D20.mzML::34473"
[3] "011_exp1-PO4_PO4_D20.mzML::26791"
[4] "010_exp1-PO4_PO4_D20.mzML::16044"
[5] "004_exp1-PO4_PO4_D20.mzML::28200"
[6] "003_exp1-PO4_PO4_D20.mzML::20011"
```

We can now join the MS2 scans (using the `filterMsLevel()` function) from the `Spectra` object and the identification results from the `PSM` object into a new annotated `Spectra` object.

```
sp <- joinSpectraData(
    filterMsLevel(sp, 2L),
    psms,
    by.x = "pkey")
```

As `calculateFragments()` needs a canonical sequence, we first add a non-modified `sequence` variable to the object.

```
sp$sequence <- gsub("[^A-Za-z]", "", sp$peptide)
```

We also remove the unmatched spectra, i.e. spectra that have a missing `sequence`. Notice that only 31.2% of the experimental MS2 scans have been identified. This low identificaiton rate highlights the relavance of tools that allow proper exploration of identication results.

```
table(is.na(sp$sequence))
```

```
FALSE  TRUE
17376 38292
```

```
sp <- sp[!is.na(sp$sequence), ]
```

Now that both PSMs and spectra have been matched, we can visualise the spectra with their matched peaks based on the identified sequences. As we're handling phosphoproteomic data, we're also adding variable modifications for phosphorylations.

The examples on Fig. 4 and 5 show how adding a modification has an impact on the identification and matched peaks.

```
plotSpectraPTM(
    sp[22],
    variable_modifications = c(T = 79.966, S = 79.966, Y = 79.966),
    USI = FALSE)
```

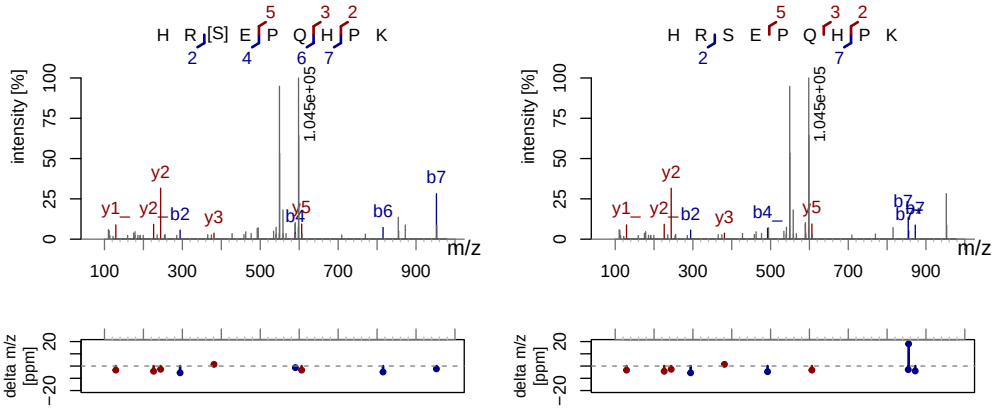


**Fig. 4** Annotated MS2 scan for HRSEPQHPK with (left) and without (right) phosphorylation. Note the mismatch of ion *b7* on the plot on the right, highlighted by a high delta mass over charge (m/z).

```
plotSpectraPTM(
    sp[1414],
    variable_modifications = c(T = 79.966, S = 79.966, Y = 79.966),
    USI = FALSE)
```

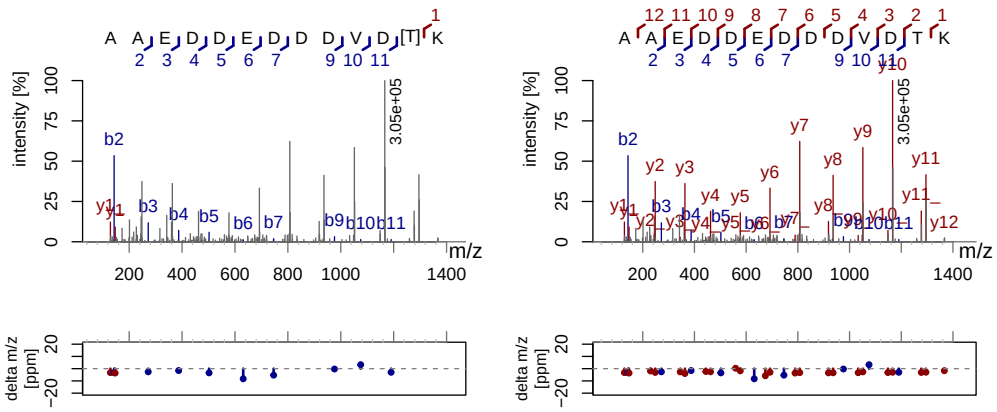


**Fig. 5** Annotated MS2 scan for AAEDDEDDDVDTK with (left) and without (right) phosphorylation.

In this chapter, we introduced *PSMatch*, an open-source R/Bioconductor package that provides a flexible and comprehensive infrastructure for handling mass spectrometry identification data. *PSMatch* facilitates the import, processing, and visualisation of PSMs, supporting diverse workflows regardless of the search engine used. Through intuitive functions, users can create structured PSM objects, filter identifications based on confidence measures, explore peptide–protein relationships, generate theoretical fragments including post-translational modifications, and visualise experimental spectra alongside theoretical predictions in conjunction with the *Spectra* package. Given the significant proportion of spectra that remain unidentified in MS-based experiments, *PSMatch* offers the necessary tools to handle, explore and understand identification data, from the actual identified spectra to the final inferred protein groups.

# 4 Notes

1. The versions of R and the specific Bioconductor release are tight. It is thus necessary to use a recent version of R itself for `BiocManager::install()` to install the latest versions of Bioconductor packages.

2. The code that generates the Sage-derived `PSM` object can be simplified using the *sager* package [19], that provides functions customized for Sage identification and quantitation data.


### Funding

GD is a FRIA grantee of the Fonds de la Recherche Scientifique - FNRS and acknowledges the support of a Special Research Funds (FSR) from the UCLouvain.


### Data and software availability

All data needed to reproduce the code is available on Zenodo (10.5281/zenodo.15854241). All software packages are open-source and freely available as part of the Bioconductor project as well as in their respective GitHub repositories (https://github.com/RforMassSpectrometry/PSMatch).

### Competing Interests

The authors report no competing interests.

## References


[1] Sticker, A., Goeminne, L., Martens, L., Clement, L.: Robust summarization and inference in proteome-wide label-free quantification. Mol. Cell. Proteomics **19**(7), 1209–1219 (2020) https://doi.org/10.1074/mcp.RA119.001624

[2] Zhu, Y., Orre, L.M., Zhou Tran, Y., Mermelekas, G., Johansson, H.J., Malyutina, A., Anders, S., Lehtiö, J.: DEqMS: A method for accurate variance estimation in differential protein expression analysis. Mol. Cell. Proteomics **19**(6), 1047–1057 (2020) https://doi.org/10.1074/mcp.TIR119.001646

[3] Hediyeh-Zadeh, S., Webb, A.I., Davis, M.J.: MsImpute: Estimation of missing peptide intensity data in label-free quantitative mass spectrometry. Mol. Cell. Proteomics **22**(8), 100558 (2023) https://doi.org/10.1016/j.mcpro.2023.100558

[4] Balliau, T., Frambourg, A., Langella, O., Martin, M.-L., Zivy, M., Blein-Nicolas, M.: MCQR: Enhancing the processing and analysis of quantitative proteomics data by incorporating chromatography and mass spectrometry information. J. Proteome Res. (2025) https://doi.org/10.1021/acs.jproteome.4c01119

[5] Panse, C., Grossmann, J.: protViz: Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics. (2023). R package version 0.7.7. https://CRAN.R-project.org/package=protViz

[6] Gabriels, R., Declercq, A., Bouwmeester, R., Degroeve, S., Martens, L.: psm_utils: A high-level python API for parsing and handling peptide-spectrum matches and proteomics search results. J. Proteome Res. (2022)

[7] Deutsch, E.W., Mendoza, L., Moritz, R.L.: Quetzal: Comprehensive peptide fragmentation annotation and visualization. J. Proteome Res. (2025)

[8] Ming, L., Zou, Y., Zhao, Y., Zhang, L., He, N., Chen, Z., Li, S.S.-C., Li, L.: MMS2plot: An R package for visualizing multiple MS/MS spectra for groups of modified and non-modified peptides. Proteomics **20**(15-16), 2000061 (2020)

[9] Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y.H., Zhang, J.: Bioconductor: open software development for computational biology and bioinformatics. Genome Biology **5**(10), 80 (2004) https://doi.org/10.1186/gb-2004-5-10-r80

[10] Huber, W., Carey, V.J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B.S., Bravo, H.C., Davis, S., Gatto, L., Girke, T., Gottardo, R., Hahne, F., Hansen, K.D., Irizarry, R.A., Lawrence, M., Love, M.I., MacDonald, J., Obenchain, V., Oleś, A.K., Pagès, H., Reyes, A., Shannon, P., Smyth, G.K., Tenenbaum, D., Waldron, L., Morgan, M.: Orchestrating high-throughput genomic analysis with Bioconductor. Nat. Methods **12**(2), 115–121 (2015) https://doi.org/10.1038/nmeth.3252

[11] Jones, A.R., Eisenacher, M., Mayer, G., Kohlbacher, O., Siepen, J., Hubbard, S.J., Selley, J.N., Searle, B.C., Shofstahl, J., Seymour, S.L., Julian, R., Binz, P.-A., Deutsch, E.W., Hermjakob, H., Reisinger, F., Griss, J., Vizcaíno, J.A., Chambers, M., Pizarro, A., Creasy, D.: The mzIdentML data standard for mass spectrometry-based proteomics results. Molecular & cellular proteomics : MCP **11**(7), 111–014381 (2012) https://doi.org/S1535-9476(20)33012-7[pii]

[12] Chick, J.M., Kolippakkam, D., Nusinow, D.P., Zhai, B., Rad, R., Huttlin, E.L., Gygi, S.P.: A mass-tolerant database search identifies a large proportion of unassigned spectra in shotgun proteomics as modified peptides. Nat. Biotechnol. **33**(7), 743–749 (2015) https://doi.org/10.1038/nbt.3267

[13] Gholamizoj, S., Ma, B.: SPEQ: quality assessment of peptide tandem mass spectra with deep learning. Bioinformatics **38**(6), 1568–1574 (2022) https://doi.org/10.1093/bioinformatics/btab874

[14] Codilupi, T., Szybinski, J., Arunasalam, S., Jungius, S., Dunbar, A.C., Stivala, S., Brkic, S., Albrecht, C., Vokalova, L., Yang, J.L., Buczak, K., Ghosh, N., Passweg, J.R., Rovo, A., Angelillo-Scherrer, A., Pankov, D., Dirnhofer, S., Levine, R.L., Koche, R., Meyer, S.C.: Development of resistance to type II JAK2 inhibitors in MPN depends on AXL kinase and is targetable. Clin. Cancer Res. **30**(3), 586–599

(2024) https://doi.org/10.1158/1078-0432.CCR-23-0163

[15] Lazear, M.R.: Sage: An Open-Source Tool for Fast Proteomics Searching and Quantification at Scale. J. Proteome Res. **22**(11), 3652–3659 (2023) https://doi.org/10.1021/acs.jproteome.3c00486

[16] Perez-Riverol, Y., Bandla, C., Kundu, D.J., Kamatchinathan, S., Bai, J., Hewapathirana, S., John, N.S., Prakash, A., Walzer, M., Wang, S., Vizcaíno, J.A.: The PRIDE database at 20 years: 2025 update. Nucleic acids research **53**(D1), 543–553 (2025) https://doi.org/7874848[pii]

[17] Deflandre, G., Gatto, L.: Data for 'PSMatch: an R/Bioconductor package to explore proteomics identification data. Zenodo (2025). https://doi.org/10.5281/zenodo.15854241 . https://doi.org/10.5281/zenodo.15854241

[18] Rainer, J., De Graeve, M., Louail, P., Gibb, S., Gatto, L.: An Open Infrastructure for Mass Spectrometry Data in R. OSF Preprints (2025). https://doi.org/10.31219/osf.io/cwt2v_v1 . osf.io/cwt2v_v1

[19] Gatto, L.: sager: Analysing sage results with R. R package version 0.3.3