MSnID Package for Handling MS/MS Identifications

Vladislav A. Petyuk

May 3, 2016

Contents

1 Introduction

MS/MS identification is a process with some uncertainty. Some peptide or protein to spectrum matches are true and some are not. There are ways to score how well the peptide/protein fragmentation pattern observed in MS/MS spectrum matches the theoretical amino acid sequence. Other ways to assess confidence of identification are:

1. the difference in theoretical and experimental masses
2. frequency of observation (true identifications tend to be more consistent)
3. peptide sequence properties (only in the case of technique involving protein digestion) such as missed cleavages or presence of cleavages not typical for a given protease and chemical reagent.

A typical and currently most reliable way to quantify uncertainty in the list of identify spectra, peptides or proteins relies on so-called decoy database. For bottom-up (i.e. involving protein digestion) approaches a common way to construct a decoy database is simple inversion of protein amino-acid sequences. The other approach commonly used in top-down (that is intact protein) approaches is based on shuffling of amino acids within protein sequence. Typically normal and decoy sequences are concatenated into one FASTA file. Some software tools (e.g. MS-GF+) perform the task of constructing and appending the decoy sequence internally on the fly. If the spectrum matches to normal protein sequence it can be true or false match. Matches to decoy part of the database are false only (excluding the palindromes). Therefore the false discovery rate (FDR) of identifications can be estimated as ratio of hits to decoy over normal parts of the protein sequence database. There are multiple levels of identification that FDR can be estimated for. First, is at the level of peptide/protein-to-spectrum matches. Second is at the level of unique peptide sequences. Note, true peptides tend to be identified by more then one spectrum. False peptide tend to be sporadic. Therefore, after collapsing the redundant peptide identifications from multiple spectra to the level of unique peptide sequence, the FDR typically increases. The extend of FDR increase depends on the type and complexity of the sample. The same trend is true for estimating the identification FDR at the protein level. True proteins tend to be identified with multiple peptides, while false protein identifications are commonly covered only by one peptide. Therefore FDR estimate tend to be even higher for protein level compare to peptide level.
The estimation of the FDR is also affected by the number of LC-MS (runs) datasets in the experiment. Again, true identifications tend to be more consistent from run to run, while false are sporadic. After collapsing the redundancy across the runs, the number of true identification reduces much stronger compare to false identifications. Therefore, the peptide and protein FDR estimates need to be re-evaluated.

The main objective of the MSnID package is to provide convenience tools for handling tasks on estimation of FDR, defining and optimizing the filtering criteria and ensuring confidence in MS/MS identification data. The user can specify the criteria for filtering the data (e.g. goodness or p-value of matching of experimental and theoretical fragmentation mass spectrum, deviation of theoretical from experimentally measured mass, presence of missed cleavages in the peptide sequence, etc), evaluate the performance of the filter judging by FDRs at spectrum, peptide and protein levels, and finally optimize the filter to achieve the maximum number of identifications while not exceeding maximally allowed FDR upper threshold.

2 Starting the project

First, the `MSnID` object has to be initialized. The main argument is path to the working directory. This directory will be used for storing cached analysis results. Caching/memoisation mechanism is based on *R.cache*.

```
> library("MSnID")
> msnid <- MSnID(".")
```

3 Reading MS/MS data

One way to read in peptide/protein to MS/MS spectrum matching (PSM) results as a table from a text file and assing the *data.frame* object.

```
> PSMresults <- read.delim(system.file("extdata", "human_brain.txt",
+                                      package="MSnID"),
+                          stringsAsFactors=FALSE)
> psms(msnid) <- PSMresults
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 997 at 37 % FDR
#peptides: 687 at 57 % FDR
#accessions: 665 at 65 % FDR
```

Alternative and currently the preferred way to read MS/MS results is by parsing mzIdentML files (*.mzid or *.mzid.gz extensions). The `read_mzIDs` function leverages *mzID* package facilities.

```
> mzids <- system.file("extdata", "c_elegans.mzid.gz", package="MSnID")
> msnid <- read_mzIDs(msnid, mzids)
```

```
reading c_elegans.mzid.gz... DONE!
```

```
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 19055 at 29 % FDR
```

```
#peptides: 9489 at 44 % FDR
#accessions: 7414 at 76 % FDR
```

Internally PSMs stored as *data.table* object.
The example file "c_elegans.mzid.gz" is based on MS-GF+ search engine. The read_mzIDs function reads results of any MS/MS search engine as long as it compliant with mzIdentML standard. In general case, use aforementioned psms<- function.

4 Updating columns

Note, to take a full advantage of the *MSnID*, the the following columns have to be present. Checking of columns happens internally.

```
[1] "accession"                "calculatedMassToCharge"   "chargeState"
[4] "experimentalMassToCharge" "isDecoy"                  "peptide"
[7] "spectrumFile"             "spectrumID"
```

Check what are the current column names in the MS/MS search results table.

```
> names(msnid)
```

```
 [1] "spectrumID"               "scan number(s)"           "acquisitionNum"
 [4] "passThreshold"            "rank"                     "calculatedMassToCharge"
 [7] "experimentalMassToCharge" "chargeState"              "MS-GF:DeNovoScore"
[10] "MS-GF:EValue"             "MS-GF:PepQValue"          "MS-GF:QValue"
[13] "MS-GF:RawScore"           "MS-GF:SpecEValue"         "AssumedDissociationMethod"
[16] "IsotopeError"             "isDecoy"                  "post"
[19] "pre"                      "end"                      "start"
[22] "accession"                "length"                   "description"
[25] "pepSeq"                   "modified"                 "modification"
[28] "idFile"                   "spectrumFile"             "databaseFile"
[31] "peptide"
```

5 Basic info on the MSnID object instance

Printing the MSnID object returns some basic information such as

- Working directory.
- Number of spectrum files used to generate data.
- Number of peptide-to-spectrum matches and corresponding FDR.
- Number of unique peptide sequences and corresponding FDR.
- Number of unique proteins or amino acid sequence accessions and corresponding FDR.

False discovery rate or FDR is defined here as a ratio of hits to decoy accessions to the non-decoy (normal) accessions. In terms of forward and revese protein sequences that would mean ratio of #reverse/#forward. While computing FDRs of PSMs and unique peptide sequences is trivial, definition of protein (accession) FDR is a subject for discussion in the field of proteomics. Here, protein (accession) FDR is computed the same

way as in IDPicker software **?** and simply constitutes a ratio of unique accessions from decoy component to non-decoy component of the sequence database.

```
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 19055 at 29 % FDR
#peptides: 9489 at 44 % FDR
#accessions: 7414 at 76 % FDR
```

6 Analysis of peptide sequences

A particular properties of peptide sequences we are interested in are

1. irregular cleavages at the termini of the peptides and
2. missing cleavage site within the peptide sequences.

The default regular expressions of valid and missed cleavage patterns correspond to trypsin. Counting the number of irregular cleavage termimi (0,1 or 2) in peptides sequence creates a new column `numIrregCleavages`.

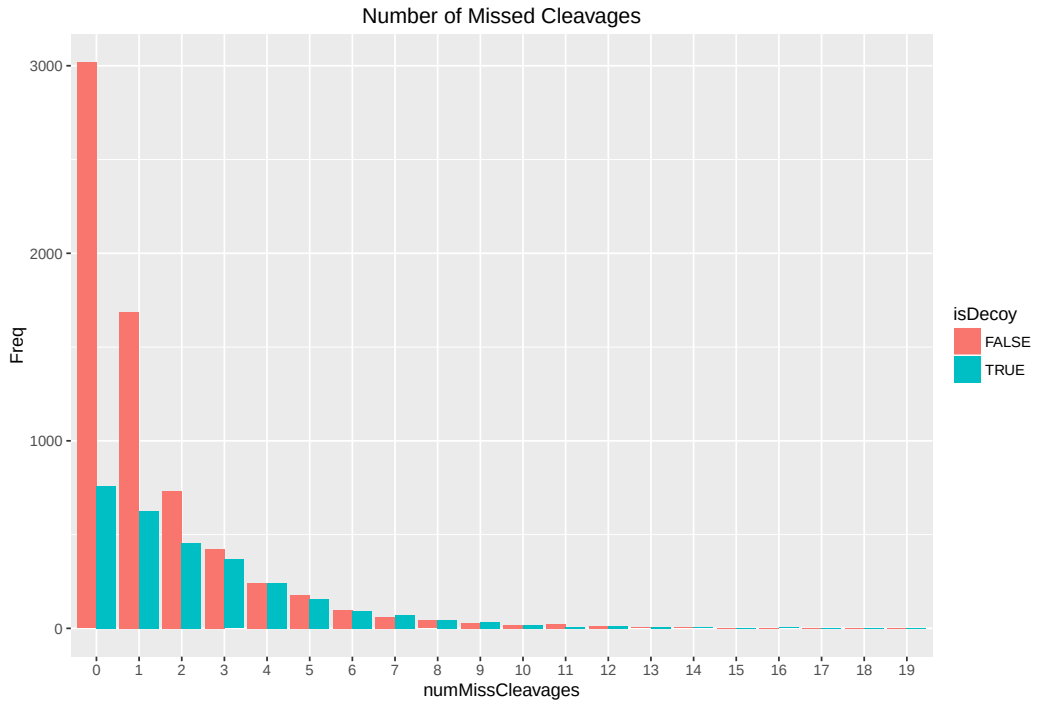
```
> msnid <- assess_termini(msnid, validCleavagePattern="[KR]\\.[^P]")
```

Counting the number of missed cleavages in peptides sequence creates a new column `numMissCleavages`.

```
> msnid <- assess_missed_cleavages(msnid, missedCleavagePattern="[KR](?=[^P$])")
```

Now the object has two more columns, `numIrregCleavages` and `numMissCleavages`, evidently corresponding to the number of termini with irregular cleavages and number of missed cleavages within the peptide sequence.

```
> pepCleav <- unique(psms(msnid)[,c("numMissCleavages", "isDecoy", "peptide")])
> pepCleav <- as.data.frame(table(pepCleav[,c("numMissCleavages", "isDecoy")]))
> library("ggplot2")
> ggplot(pepCleav, aes(x=numMissCleavages, y=Freq, fill=isDecoy)) +
+     geom_bar(stat='identity', position='dodge') +
+     ggtitle("Number of Missed Cleavages")
```

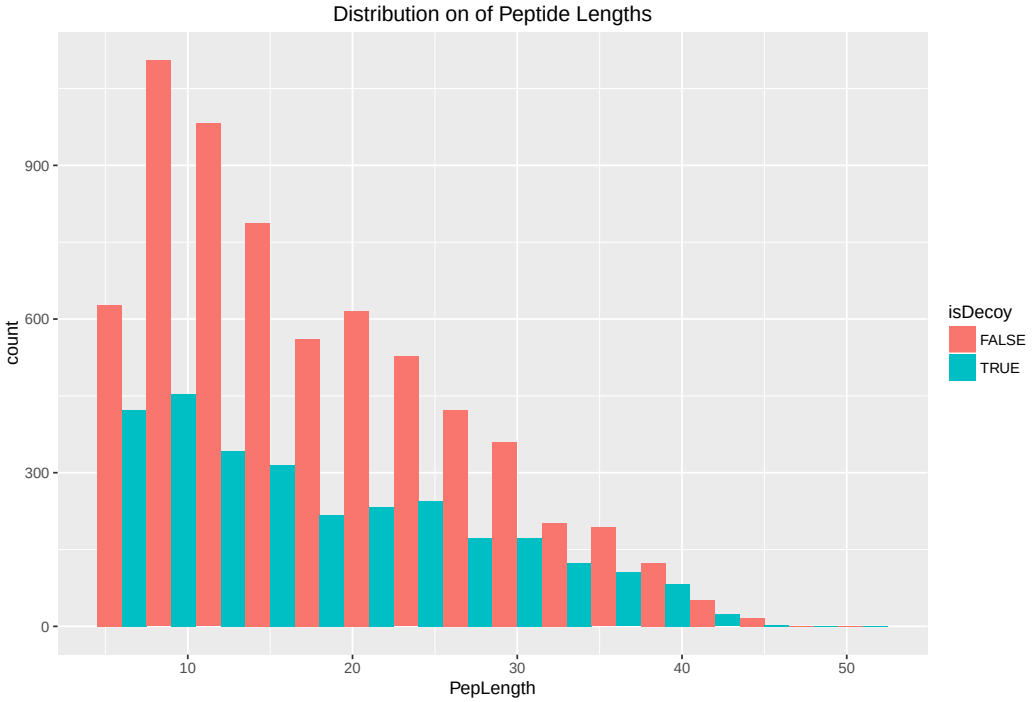


Peptide sequences, as any other column, can by accessed by directly using $ operator. For example: Counting number of cysteins per peptide sequence

```
> msnid$numCys <- sapply(lapply(strsplit(msnid$peptide,''),'==','C'),sum)
```

Calculating peptide lengths. Note, -4 decrements the AA count by two the flanking AAs and the two dots separating them from the actual peptide sequence.

```
> msnid$PepLength <- nchar(msnid$peptide) - 4
> pepLen <- unique(psms(msnid)[,c("PepLength", "isDecoy", "peptide")])
> ggplot(pepLen, aes(x=PepLength, fill=isDecoy)) +
+     geom_histogram(position='dodge', binwidth=3) +
+     ggtitle("Distribution on of Peptide Lengths")
```



7 Trimming the data

The main way for trimming or filtering the data is `apply_filter` function. The second argument can be either 1) a string representing expression that will be evaluated in the context of data.frame containing MS/MS results or 2) *MSnFilter* class object (explained below). Note, the reduction in FDR. Assuming that the sample has been digested with trypsin, the true identifications tend to be fully tryptic and contain fewer missed cleavages. Original FDRs.

```
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 19055 at 29 % FDR
#peptides: 9489 at 44 % FDR
#accessions: 7414 at 76 % FDR
```

Leaving only fully tryptic peptides.

```
> msnid <- apply_filter(msnid, "numIrregCleavages == 0")
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 15274 at 20 % FDR
#peptides: 7076 at 34 % FDR
#accessions: 5210 at 65 % FDR
```

Filtering out peptides with more then 2 missed cleavages.

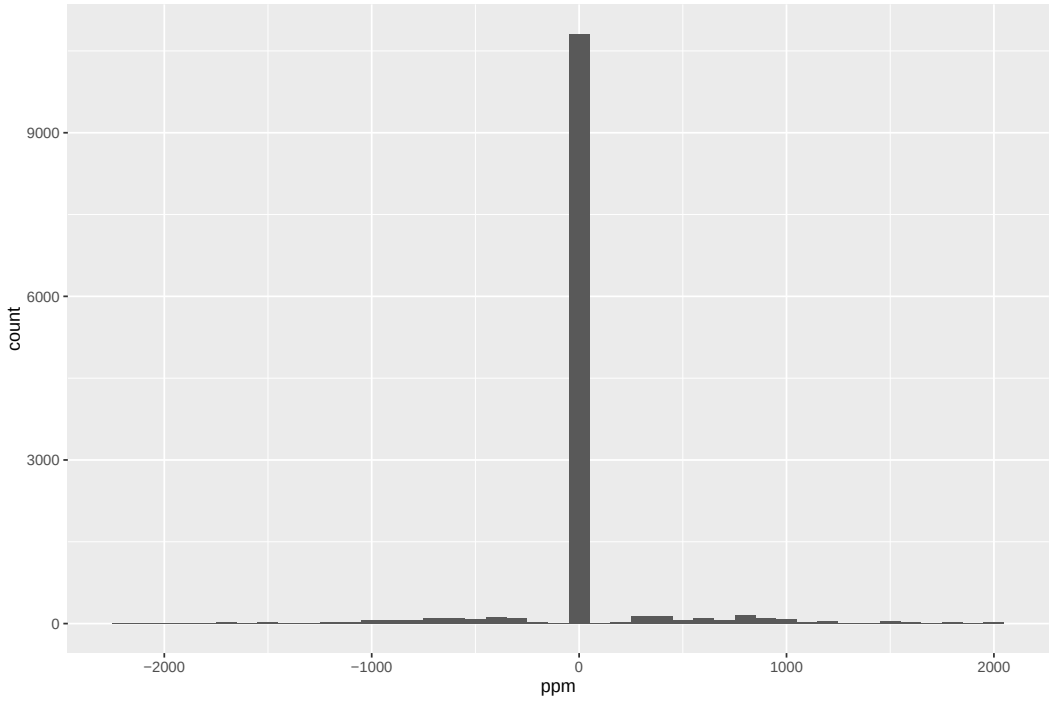
```
> msnid <- apply_filter(msnid, "numMissCleavages <= 2")
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 12848 at 14 % FDR
#peptides: 5598 at 23 % FDR
#accessions: 3759 at 53 % FDR
```

8 Parent ion mass measurement error

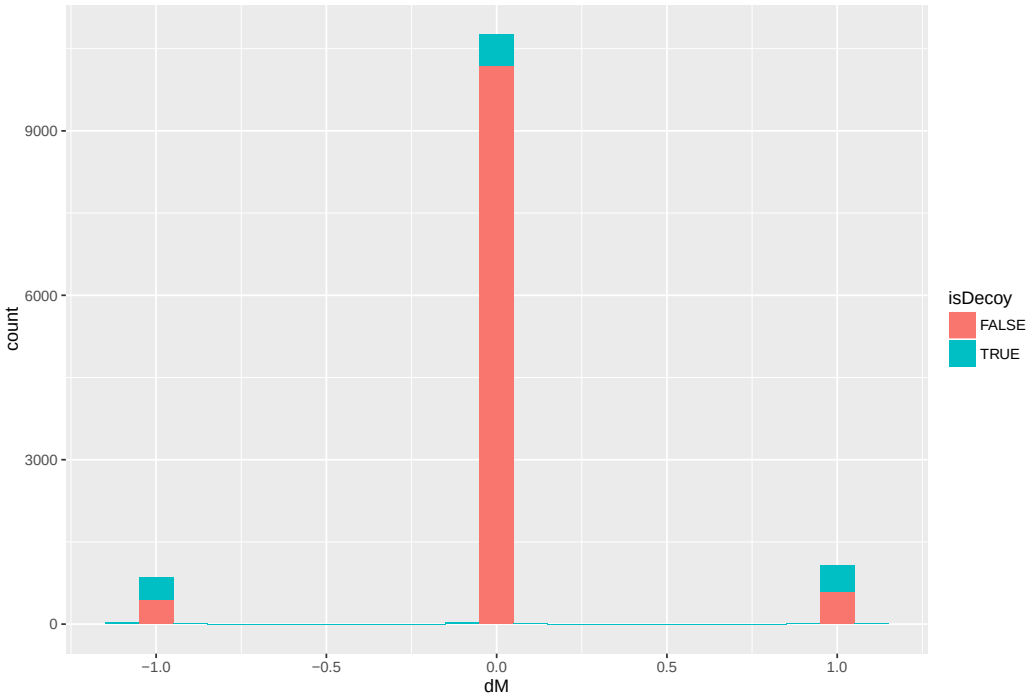
Assuming both `calculatedMassToCharge` and `experimentalMassToCharge` are present in `names(msnid)`, one can access parent ion mass measurement in points per million (ppm) units.

```
> ppm <- mass_measurement_error(msnid)
> ggplot(as.data.frame(ppm), aes(x=ppm)) +
+     geom_histogram(binwidth=100)
```



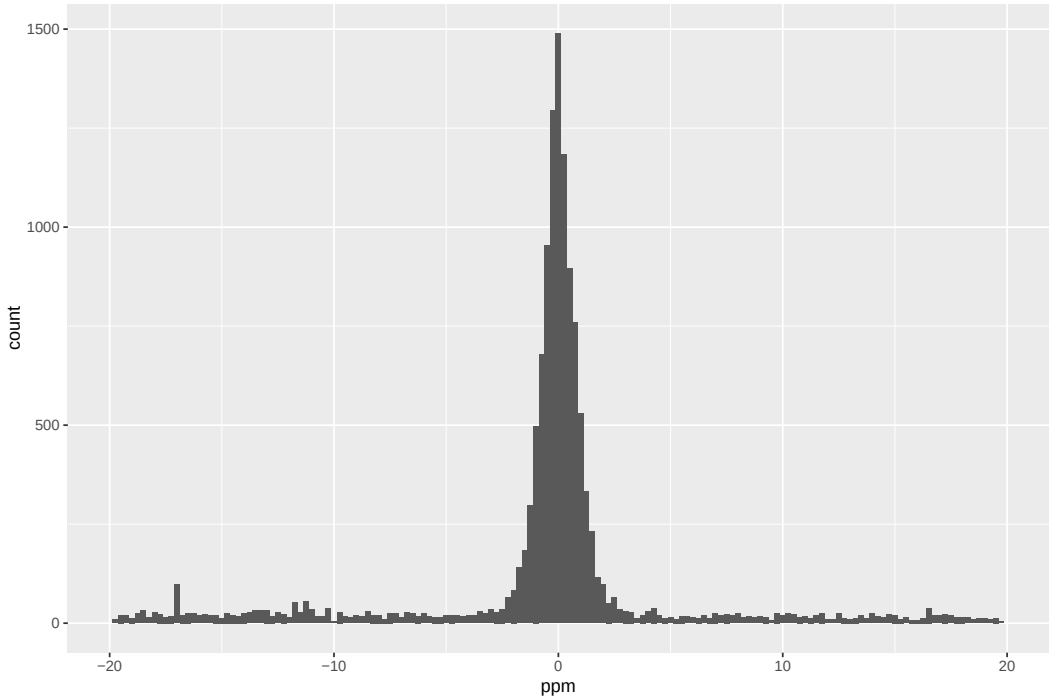
Note, although the MS/MS search was done with ± 20ppm parent ion mass tolerance, error stretch over 1000 in ppm units. The reason is that the settings of the MS/MS search engine MS-GF+ (used for the analysis of this LC-MS dataset) fairly assumed that the instrument could have picked non-monoisotopic peaks of parent ion for fragmentation and thus considered peptides that were off by ± 1 Da (^{13}C-^{12}C to be exact). Similar settings can be found in other search engines (e.g X!Tandem).

```
> dM <- with(psms(msnid),
+     (experimentalMassToCharge-calculatedMassToCharge)*chargeState)
> x <- data.frame(dM, isDecoy=msnid$isDecoy)
> ggplot(x, aes(x=dM, fill=isDecoy)) +
+     geom_histogram(position='stack', binwidth=0.1)
```



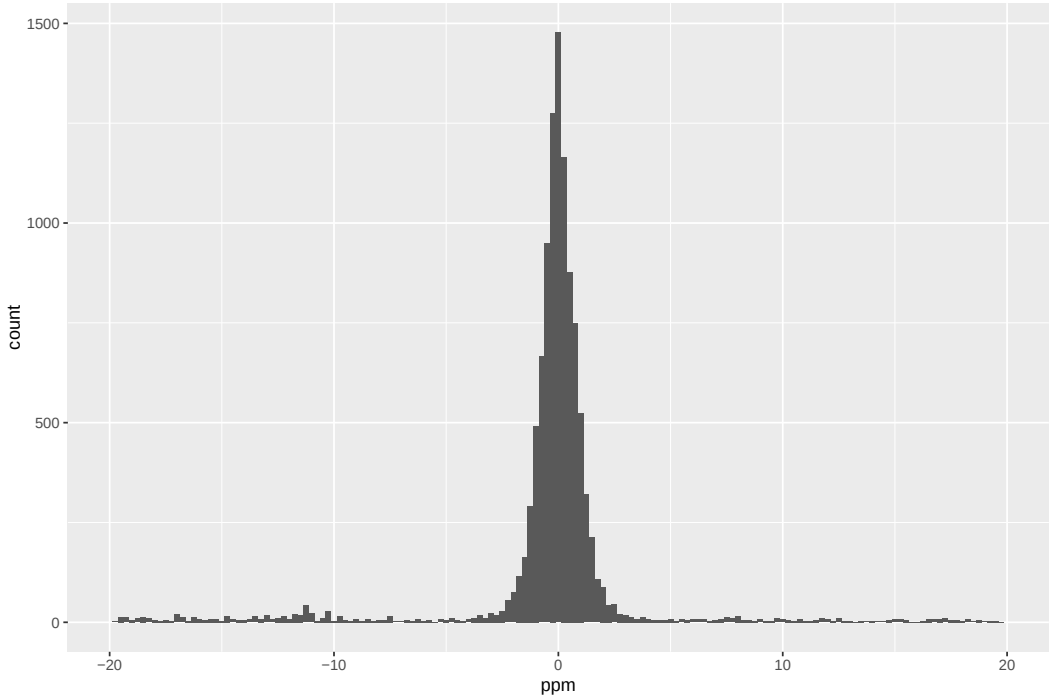
Ideally, to avoid this problem, the MS/MS datasets have to be either aquired in MIPS (monoisotopic ion precurson selection) mode or preprocessed with DeconMSn **?** tools that identifies the monoisotipic peaks post-experimentally. The *MSnID* package provide a simple `correct_peak_selection` function that simply adds or subtracts the difference between ^{13}C and ^{12}C to make the error less then 1 Dalton.

```
> msnid.fixed <- correct_peak_selection(msnid)
> ppm <- mass_measurement_error(msnid.fixed)
> ggplot(as.data.frame(ppm), aes(x=ppm)) +
+     geom_histogram(binwidth=0.25)
```



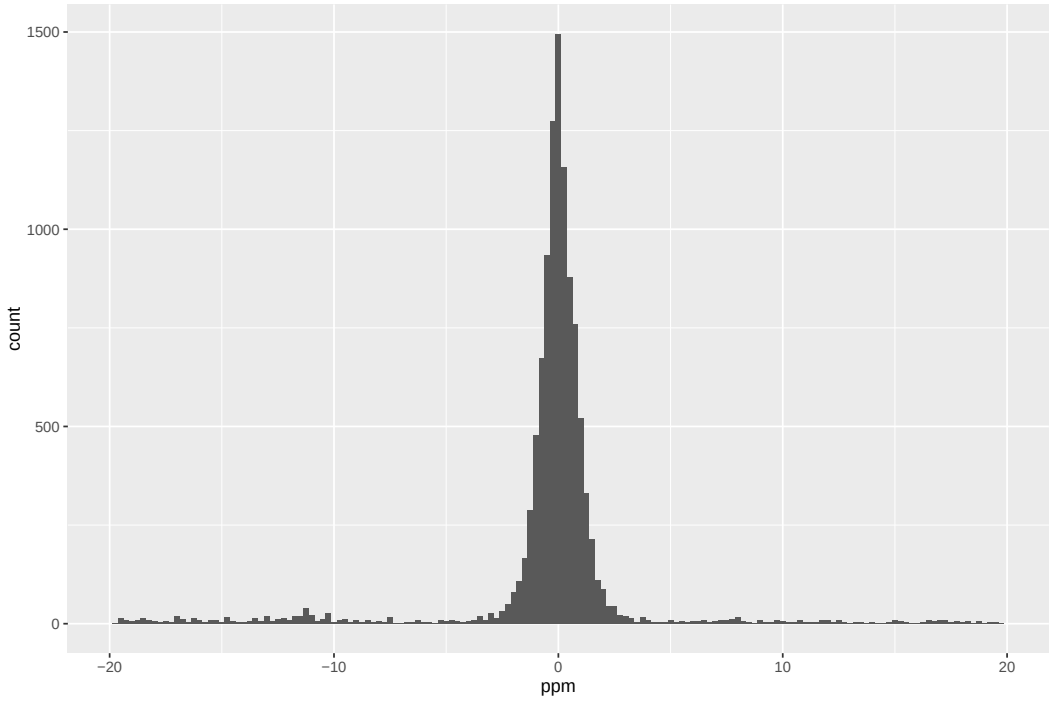
Alternatively, one can simply apply a filter to remove any matches that do not fit the ± 20 ppm tolerance.

```
> msnid.chopped <- apply_filter(msnid, "abs(mass_measurement_error(msnid)) < 20")
> ppm <- mass_measurement_error(msnid.chopped)
> ggplot(as.data.frame(ppm), aes(x=ppm)) +
+     geom_histogram(binwidth=0.25)
```



For further processing we'll consider the `msnid.chopped` data that ignores matches with 1 Da errors. Note, if the center of the histogram is significantly shifted from zero, `experimentalMassToCharge` can be post-experimentally recalibrated. This MS/MS data was preprocessed with DtaRefinery tool **?** that post-experimentally eliminates any systematic mass measurement error. At this point, the `recalibrate` function implements the most simplistic algorithm avalable in the DtaRefinery tool.

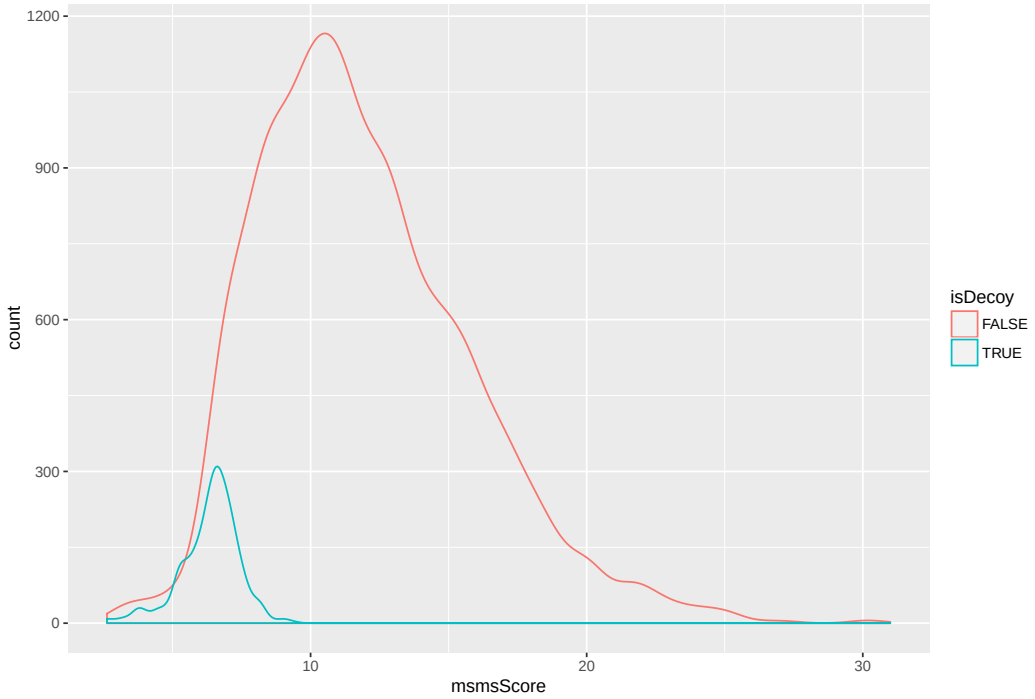
```
> msnid <- recalibrate(msnid.chopped)
> ppm <- mass_measurement_error(msnid)
> ggplot(as.data.frame(ppm), aes(x=ppm)) +
+     geom_histogram(binwidth=0.25)
```



9 `MSnIDFilter` object for filtering MS/MS identifications

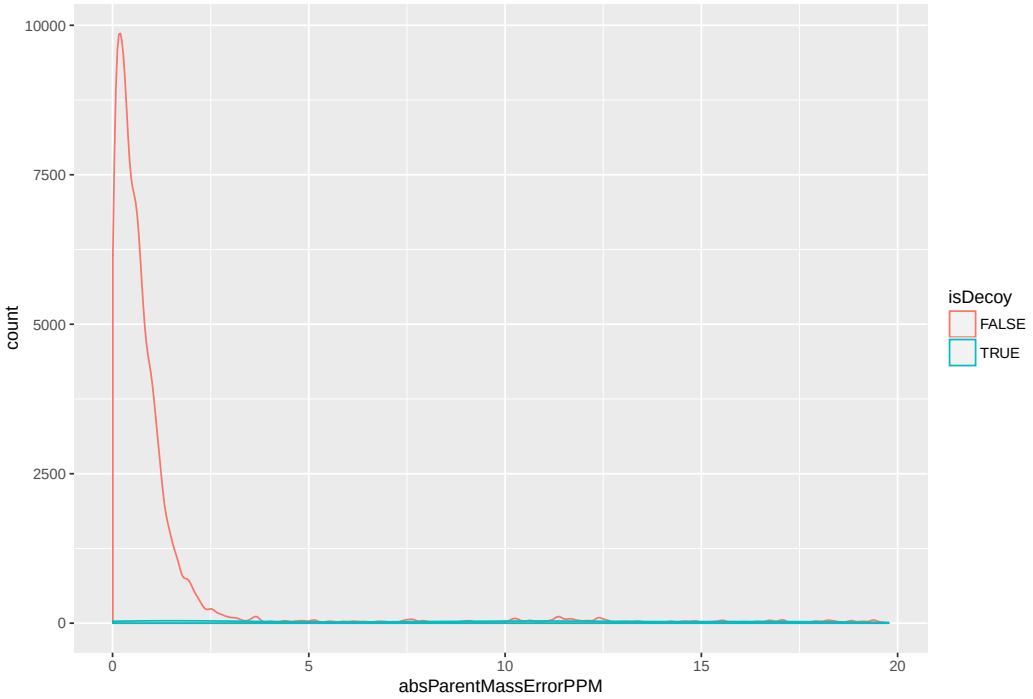
The criteria that will be used for filtering the MS/MS data has to be present in the `MSnID` object. We will use -log10 transformed MS-GF+ Spectrum E-value, reflecting the goodness of match experimental and theoretical fragmentation patterns as one the filtering criteria. Let's store it under the "msmsScore" name. The score density distribution shows that it is a good discriminant between non-decoy (red) and decoy hits (green).
For alternative MS/MS search engines refer to the engine-specific manual for the names of parameters reflecting the quality of MS/MS spectra matching. Examples of such parameters are `E-Value` for X!Tandem and `XCorr` and Δ`Cn2` for SEQUEST.

```
> msnid$msmsScore <- -log10(msnid$`MS-GF:SpecEValue`)
> params <- psms(msnid)[,c("msmsScore","isDecoy")]
> ggplot(params) +
+     geom_density(aes(x = msmsScore, color = isDecoy, ..count..))
```



As a second criterion we will be using the absolute mass measurement error (in ppm units) of the parent ion. The mass measurement errors tend to be small for non-decoy (enriched with real identificaiton) hits (red line) and is effectively uniformly distributed for decoy hits.

```
> msnid$absParentMassErrorPPM <- abs(mass_measurement_error(msnid))
> params <- psms(msnid)[,c("absParentMassErrorPPM","isDecoy")]
> ggplot(params) +
+     geom_density(aes(x = absParentMassErrorPPM, color = isDecoy, ..count..))
```



MS/MS fiters are handled by a special *MSnIDFilter* class objects. Individual filtering criteria can be set by name (that is present in `names(msnid)`), comparison operator ($>$, $<$, $=$, ...) defining if we should retain hits with higher or lower given the threshold and finally the threshold value itself.

```
> filtObj <- MSnIDFilter(msnid)
> filtObj$absParentMassErrorPPM <- list(comparison="<", threshold=10.0)
> filtObj$msmsScore <- list(comparison=">", threshold=10.0)
> show(filtObj)
```

```
MSnIDFilter object
(absParentMassErrorPPM < 10) & (msmsScore > 10)
```

Let's evaluate the performace of the filter at three different levels of confidence assessment.

```
> evaluate_filter(msnid, filtObj, level="PSM")
```

```
    fdr    n
PSM   0 6646
```

```
> evaluate_filter(msnid, filtObj, level="peptide")
```

```
        fdr    n
peptide   0 2455
```

```
> evaluate_filter(msnid, filtObj, level="accession")
```

```
          fdr    n
accession   0 1009
```

10 Optimizing the MS/MS filter to achieve the maximum number of identifications within a given FDR upper limit threshold

The threshold values in the example above are not necessarily optimal and set just be in the range of probable values. Filters can be optimized to ensure maximum number of identifications (peptide-to-spectrum matches, unique peptide sequences or proteins) within a given FDR upper limit.
First, the filter can be optimized simply by stepping through individual parameters and their combinations. The idea has been described in **?**. The resulting `MSnIDFilter` object can be used for final data filtering or can be used as a good starting parameters for follow-up refining optimizations with more advanced algorithms.

```
> filtObj.grid <- optimize_filter(filtObj, msnid, fdr.max=0.01,
+                                 method="Grid", level="peptide",
+                                 n.iter=500)
> show(filtObj.grid)
```

```
MSnIDFilter object
(absParentMassErrorPPM < 3) & (msmsScore > 7.4)
```

The resulting `filtObj.grid` can be further fine tuned with such optimization routines as simulated annealing or Nelder-Mead optimization.

```
> filtObj.nm <- optimize_filter(filtObj.grid, msnid, fdr.max=0.01,
+                                 method="Nelder-Mead", level="peptide",
+                                 n.iter=500)
> show(filtObj.nm)
```

```
MSnIDFilter object
(absParentMassErrorPPM < 4.1) & (msmsScore > 7.2)
```

Let's compare the original (good guess) and optimized fileters. Obviously the latter yields much more peptide identifications, while not exceeding the maximally allowed FDR threshold of 1

```
> evaluate_filter(msnid, filtObj, level="peptide")
```

```
        fdr    n
peptide   0 2455
```

```
> evaluate_filter(msnid, filtObj.nm, level="peptide")
```

```
                fdr    n
peptide 0.009918846 3360
```

Finally we'll apply the optimized filter to proceed with further steps in the analysis pipeline.

```
> msnid <- apply_filter(msnid, filtObj.nm)
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 9278 at 0.43 % FDR
#peptides: 3360 at 0.99 % FDR
#accessions: 1247 at 3.2 % FDR
```

Identifications that matched decoy and contaminant protein sequences can be removed by providing filters in the forms of text strings that will be evaluated in the context of PSM table.

```
> msnid <- apply_filter(msnid, "isDecoy == FALSE")
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 9238 at 0 % FDR
#peptides: 3327 at 0 % FDR
#accessions: 1208 at 0 % FDR
```

```
> msnid <- apply_filter(msnid, "!grepl('Contaminant',accession)")
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  1
#PSMs: 9230 at 0 % FDR
#peptides: 3321 at 0 % FDR
#accessions: 1205 at 0 % FDR
```

11 Data output and interface with other Bioconductor packages

One can extract the entire PSMs tables as `data.frame` or `data.table`

```
> psm.df <- psms(msnid)
> psm.dt <- as(msnid, "data.table")
```

If only interested in the non-redundant list of confidently identified peptides or proteins

```
> peps <- peptides(msnid)
> head(peps)
```

```
[1] "K.AISQIQEYVDYYGGSGVQHIALNTSDIITAIEALR.A"
[2] "K.SAGSGYLVGDSLTFVDLLVAQHTADLLAANAALLDEFPQFK.A"
[3] "K.NSIFTNVAETANGEYFWEGLEDEIADKNVDITTWLGEK.W"
[4] "R.VFCLLGDGESAEGSVWEAAAFASIYKLDNLVAIVDVNR.L"
[5] "R.TTDSDGNNTGLDLYTVDQVEHSNYVEQNFLDFIFVFR.K"
[6] "R.VIVTEIDPINALQAAMEGYEVTTLEEAAPK.A"
```

```
> prots <- accessions(msnid)
> head(prots)
```

```
[1] "CE02347" "CE07055" "CE12728" "CE36358" "CE36359" "CE36360"
```

```
> prots <- proteins(msnid) # may be more intuitive then accessions
> head(prots)
```

```
[1] "CE02347" "CE07055" "CE12728" "CE36358" "CE36359" "CE36360"
```

The *MSnID* package is aimed at providing convenience functionality to handle MS/MS identifications. Quantification *per se* is outside of the scope of the package. The only type of quantitation that can be seamlessly tied with MS/MS identification analysis is so-called *spectral counting* approach. In such an approach a peptide abundance is considered to be directly proportional to the number of matched MS/MS spectra. In its turn protein abunance is proportional to the sum of the number of spectra of the matching peptides. The *MSnID* object can be converted to an *MSnSet* object defined in *MSnbase* that extends generic Bioconductor *eSet* class to quantitative proteomics data. The spectral count data can be analyzed with *msmsEDA*, *msmsTests* or *DESeq* packages.

```
> msnset <- as(msnid, "MSnSet")
> library("MSnbase")
> head(fData(msnset))
```

```
                                                                    peptide
-.APPSQDVLKEIFNLYDEELDGK.I                          -.APPSQDVLKEIFNLYDEELDGK.I
-.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A -.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A
-.APPTFADLGK.S                                                  -.APPTFADLGK.S
-.GFQNLWFSHPR.K                                                -.GFQNLWFSHPR.K
-.GIDINHKHDR.V                                                  -.GIDINHKHDR.V
-.MKVNPFVSSDSGK.S                                              -.MKVNPFVSSDSGK.S
                                                        accession
-.APPSQDVLKEIFNLYDEELDGK.I                        CE01236, CE30652
-.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A            CE01236, CE30652
-.APPTFADLGK.S                                           CE29443
-.GFQNLWFSHPR.K                                          CE26849
-.GIDINHKHDR.V                                           CE16650
-.MKVNPFVSSDSGK.S                       CE03278, CE20731, CE32883
```

```
> head(exprs(msnset))
```

```
                                    c_elegans_A_3_1_21Apr10_Draco_10-03-04_dta.txt
-.APPSQDVLKEIFNLYDEELDGK.I                                                       1
-.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A                                            4
-.APPTFADLGK.S                                                                   2
-.GFQNLWFSHPR.K                                                                  1
-.GIDINHKHDR.V                                                                   2
-.MKVNPFVSSDSGK.S                                                                1
```

Note, the convertion from `MSnID` to `MSnSet` uses peptides as features. The number of redundant peptide observations represent so-called spectral count that can be used for rough quantitative analysis. Summing of all of the peptide counts to a proteins level can be done with `combineFeatures` function from *MSnbase* package.

```
> msnset <- combineFeatures(msnset,
+                           fData(msnset)$accession,
+                           redundancy.handler="unique",
+                           fun="sum",
+                           cv=FALSE)
> head(fData(msnset))
```

```
                                    peptide accession
CE00078                          K.RLPVAPR.G   CE00078
CE00103 K.LPNDDIGVQVSYLGEPHTFTPEQVLAALLTK.L   CE00103
CE00133                        K.AILKEVAK.G   CE00133
CE00134               K.ATITAEFPILTGFFQNEK.I   CE00134
CE00209            K.ALEGPGPGEDAAHSENNPPR.N   CE00209
CE00302                 K.LTYFDIHGLAEPIR.L   CE00302
```

```
> head(exprs(msnset))
```

```
        c_elegans_A_3_1_21Apr10_Draco_10-03-04_dta.txt
CE00078                                              5
CE00103                                              3
CE00133                                              1
CE00134                                              2
CE00209                                              8
CE00302                                              2
```