MSnID Package for Handling MS/MS Identifications

Vladislav A. Petyuk

April 2, 2014

Contents

1 Introduction

The *MSnID* package is a convenience tool to manipulating MS/MS identifications in R/Bioconductor environment. The input file can be a flat text file with the results of MS/MS search in a table format or mzIdentML files. The main S4 class for storing data is *MSnID*. A class for handling MS/MS filtering criteria is *MSnIDFilter*.

2 Case study

```
> library("MSnID")
```

2.1 Downloading example mzIdentML files

As an example we will use datasets from C.elegans proteomics study describing the effect of daf-16 mutation on the protein abundances Depuydt et al. (2013, 2014). There will be 5 datasets corresponding to long-living glp-4; daf-2 strain and 5 datasets corresponding to the strain with compensatory mutation in daf-16 transcription factor that restores (shortens) life-span back to normal. Files available at PeptideAtlas repository PASS00308.

```
> try(setInternet2(FALSE),silent=TRUE)
> ftp <- "ftp://PASS00308:PJ5348t@ftp.peptideatlas.org/"
> meta <- read.delim(sprintf("%sSample_Metadata.txt", ftp), as.is=TRUE)
> meta <- subset(meta, Age == 'young' & Diet == 'ff') # explore the effect of daf-16.
> for( dataset in meta$PNNL.Dataset.Name){
+    cel.path <- sprintf("%s/MSGFPlus_Results/MZID_Files/%s_msgfplus.mzid.gz",
+                        ftp, dataset)
+    download.file(cel.path, sprintf("%s_msgfplus.mzid.gz", dataset))
+ }
```

2.2 Setting up working directory

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")
> # start the project by providing work directory
> msnid <- MSnID(".")
```

```
Note, the anticipated/suggested columns in the
peptide-to-spectrum matching results are:
Peptide, Accession, isDecoy, calculatedMassToCharge, experimentalMassToCharge, chargeState, specti
```

2.3 Reading MS/MS data

The main intended way to read MS/MS results is by parsing mzIdentML files (*.mzid or *.mzid.gz extensions) using *mzID* package facilities.

```
> mzids <- list.files(".", pattern=".mzid.gz")
> # in this case I read all *.mzid.gz files present in the current directory.
> msnid <- read_mzIDs(msnid, mzids)
```

```
Loaded cached data
```

Alternative way is to pass MS/MS search results as peptide-to-spectra matches (PSMs) *data.frame*. Internally PSMs stored as *data.table* object.

```
> psms(msnid) <- yourPSMresults
```

2.4 Updating columns

Note, to have a fully functioning `MSnID` object, the following columns has to be present.

```
> MSnID:::.mustBeColumns
```

```
[1] "Peptide"                 "Accession"               "isDecoy"
[4] "calculatedMassToCharge"  "experimentalMassToCharge" "chargeState"
[7] "spectrumFile"            "spectrumID"
```

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

```
> names(msnid)
```

```
 [1] "spectrumid"               "acquisitionnum"           "passthreshold"
 [4] "rank"                     "calculatedmasstocharge"   "experimentalmasstocharge"
 [7] "chargestate"              "ms-gf:denovoscore"        "ms-gf:evalue"
[10] "ms-gf:pepqvalue"          "ms-gf:qvalue"             "ms-gf:rawscore"
[13] "ms-gf:specevalue"         "assumeddissociationmethod" "isotopeerror"
[16] "pepseq"                   "modified"                 "modification"
[19] "isdecoy"                  "post"                     "pre"
[22] "end"                      "start"                    "accession"
[25] "length"                   "description"              "spectrumFile"
[28] "databaseFile"
```

Trivial updates to some of the columns to make sure upper/lower letter cases are correct.

```
> msnid$Accession <- msnid$accession
> msnid$isDecoy <- msnid$isdecoy
> msnid$calculatedMassToCharge <- msnid$calculatedmasstocharge
> msnid$experimentalMassToCharge <- msnid$experimentalmasstocharge
> msnid$chargeState <- msnid$chargestate
> msnid$spectrumID <- msnid$spectrumid
```

Creating "Peptide" column as peptide sequence with flanking amino acids (X.XXXX.X).

```
> msnid$Peptide <- paste(msnid$pre, msnid$pepseq, msnid$post, sep='.')
```

Removing the columns we don't need anymore.

```
> msnid$accession <- NULL
> msnid$pepseq <- NULL
> msnid$pre <- NULL
> msnid$post <- NULL
> msnid$isdecoy <- NULL
> msnid$spectrumid <- NULL
> msnid$experimentalmasstocharge <- NULL
> msnid$calculatedmasstocharge <- NULL
> msnid$chargestate <- NULL
```

2.5 Exploring the MSnID object

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 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 unqiue accessions from decoy component to non-decoy component of sequence database.

```
> show(msnid)

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 190589 at 31 % FDR
#Peptides: 57662 at 75 % FDR
#Accessions: 28728 at 94 % FDR
```

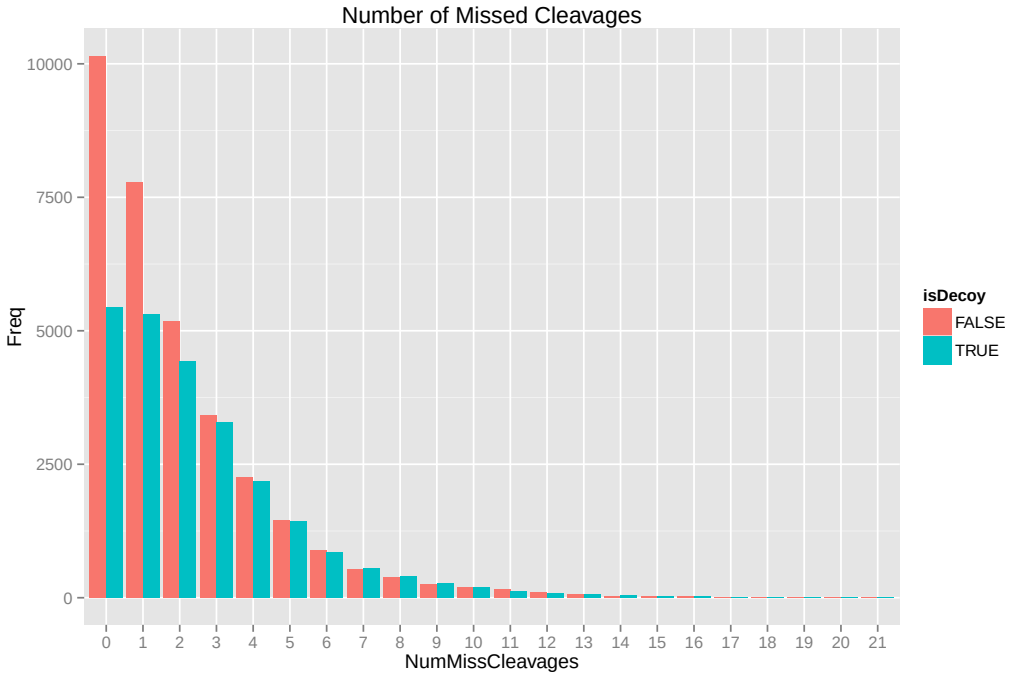
2.6 Analysis of peptide sequences

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.

```
> # creates new column NumIrregCleavages
> msnid <- assess_termini(msnid, validCleavagePattern="[RK]\\.[^P]")
> # creates new column NumMissCleavages
> msnid <- assess_missed_cleavages(msnid, missedCleavagePattern="[KR](?=[^P$])")
```

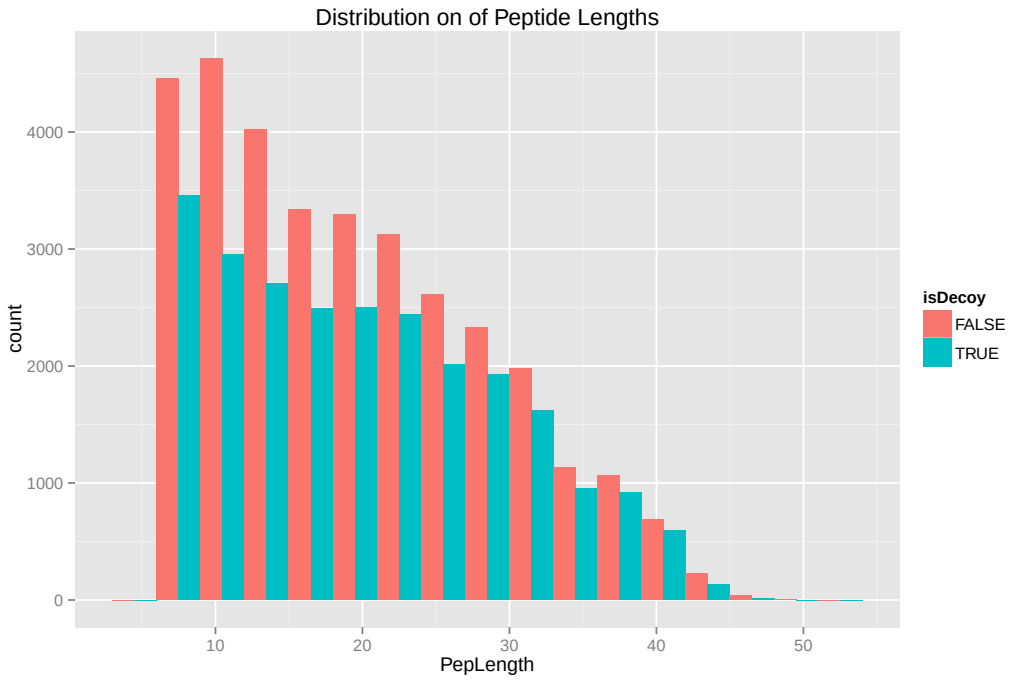
Now the object has two more columns, NumIrregCleavages and NumMissCleavages, evidently corresponding to the number of termini with irregular cleavages and number of missed cleavages within the peptide sequence.

```
> 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 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
> 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")
```



2.7 Trimming the data

The main facility 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. True identifications tend to be fully tryptic and contain fewer missed cleavages. class object.

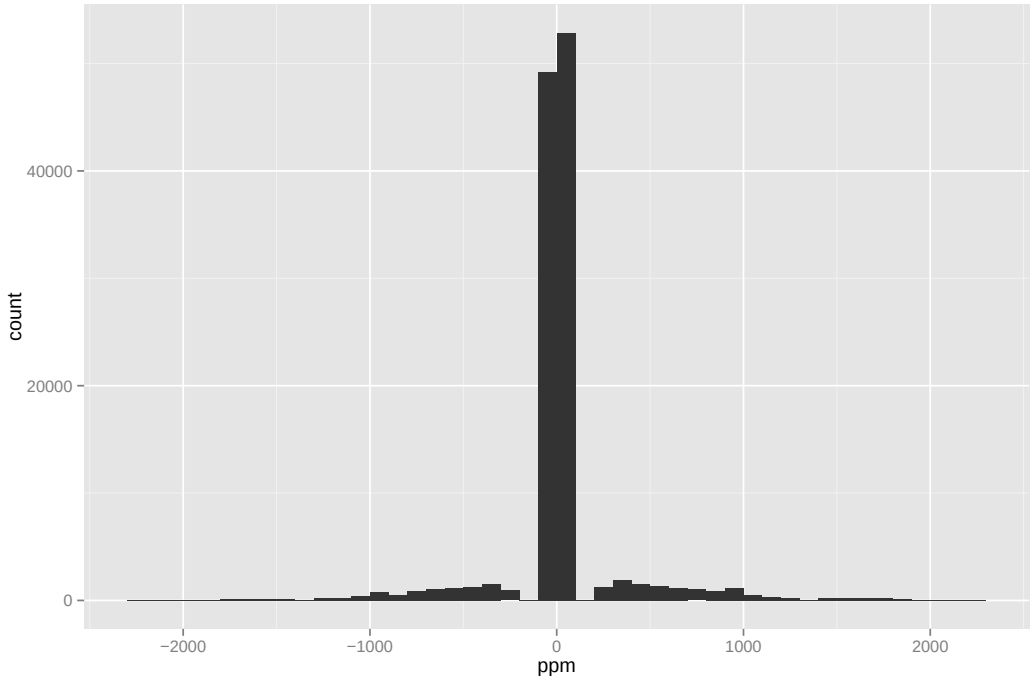
```
> # take a look at original FDRs
> show(msnid)

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 190589 at 31 % FDR
#Peptides: 57662 at 75 % FDR
#Accessions: 28728 at 94 % FDR
```

```
> # Leave only fully tryptic.
> msnid <- apply_filter(msnid, "NumIrregCleavages == 0")
> show(msnid)

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 150068 at 22 % FDR
#Peptides: 37081 at 66 % FDR
#Accessions: 22035 at 89 % FDR

> # Retain peptides with at most 2 missed cleavages.
> msnid <- apply_filter(msnid, "NumMissCleavages <= 2")
> show(msnid)

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 124163 at 14 % FDR
#Peptides: 24234 at 54 % FDR
#Accessions: 16641 at 84 % FDR
```

2.8 Parent ion mass measurement accuracy

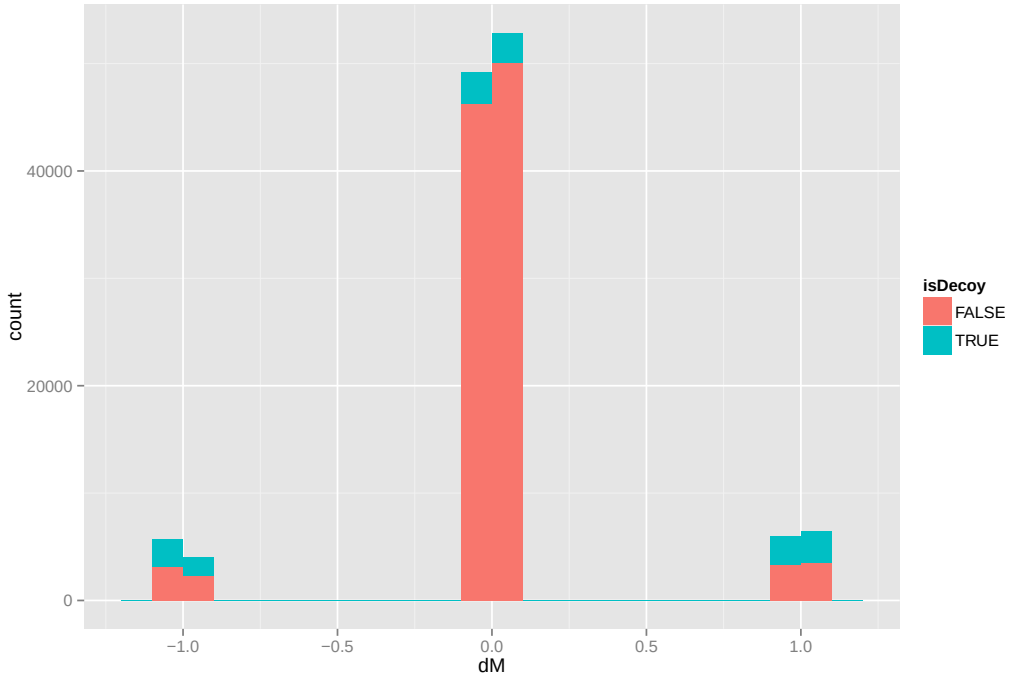
Assuming both `calculatedMassToCharge` and `experimentalMassToCharge` are present in `names(msnid)`, one can access parent ion mass measurement in points per million (ppm) units.

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



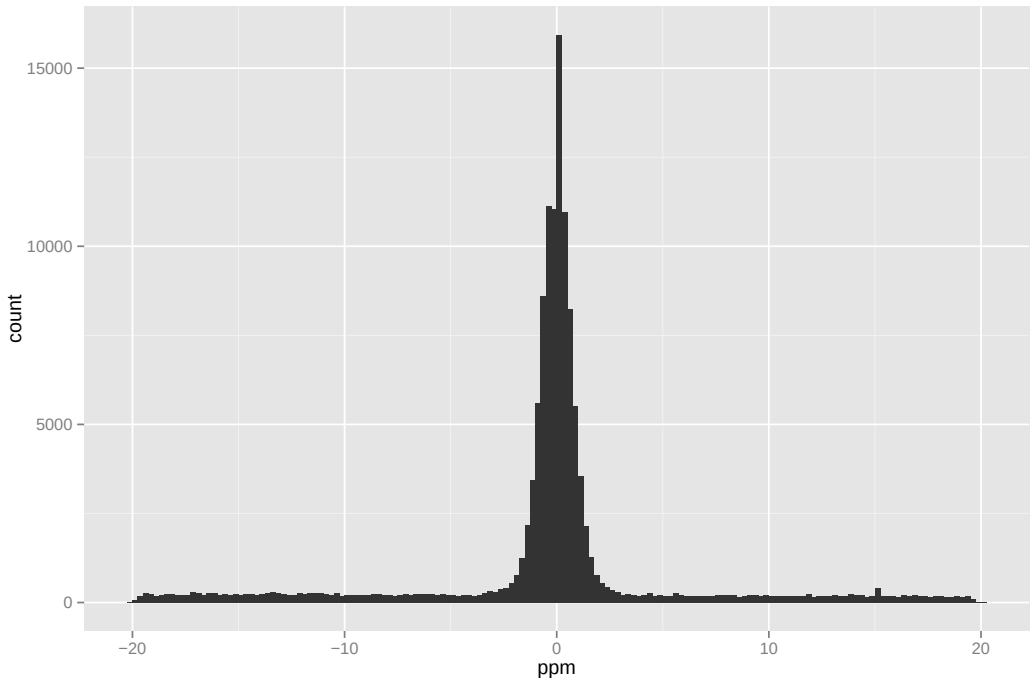
Note, although the MS/MS search was done with +/- 20ppm parent ion mass tolerance, error stretch over 1000 in ppm units. The reason is that setting of the MS/MS search engine MS-GF+ assumed imperfect peak picking for fragmentation and considered peptides that were +/- 1 Da (^{13}C-^{12}C to be exact) off.

```
> 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)
```



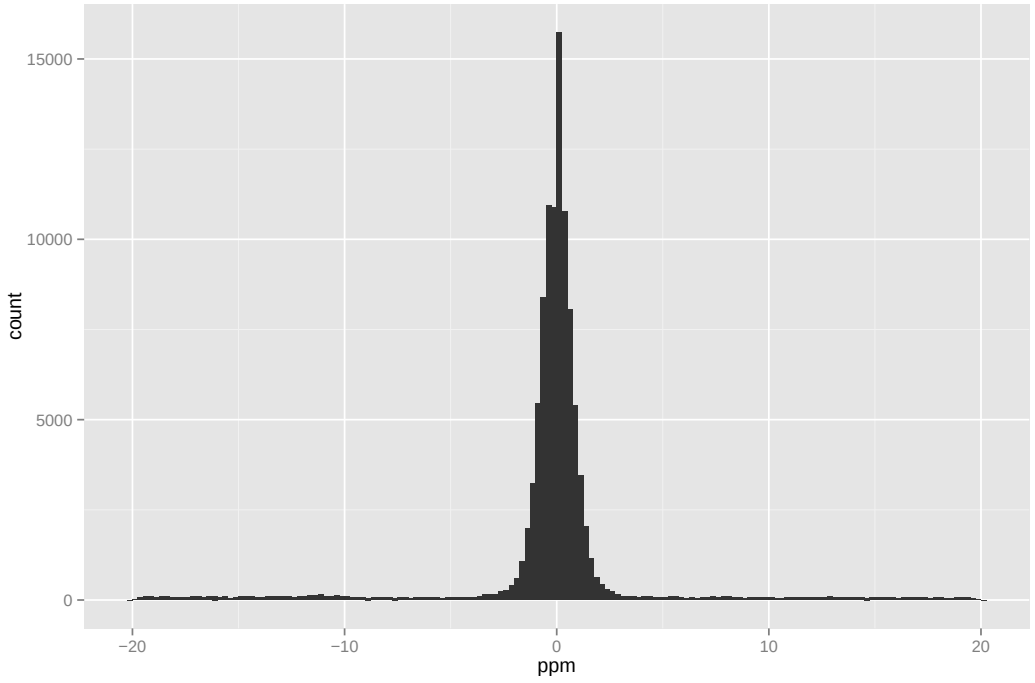
Selection and reporting of wrong monoisotopic masses can be corrected.

```
> 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)
```



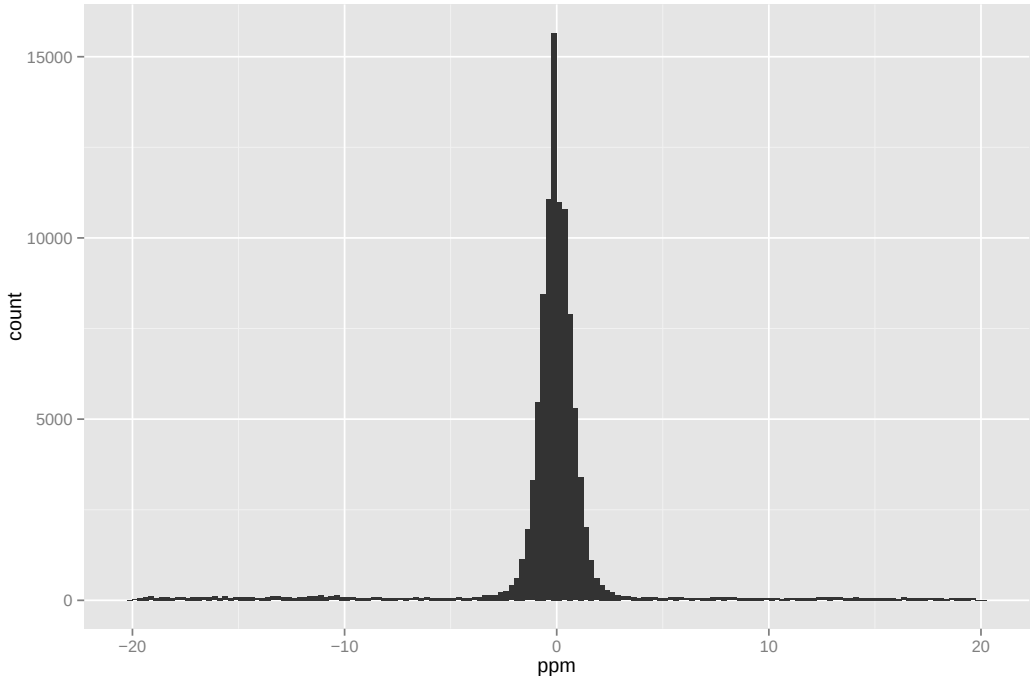
Alternatively, one can simply apply a filter to remove any matches that do not fit the +/- 20 ppm tolerance.

```
> 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)
```



!NOTE! Make sure the text is correct. For further processing we'll consider `msnid.chopped` data that is adjusted for 1 Da errors in picking of wrong monoisotopic peak. Note, if the center of the histogram is significantly shifted from zero, `experimentalMassToCharge` can be recalibrated.

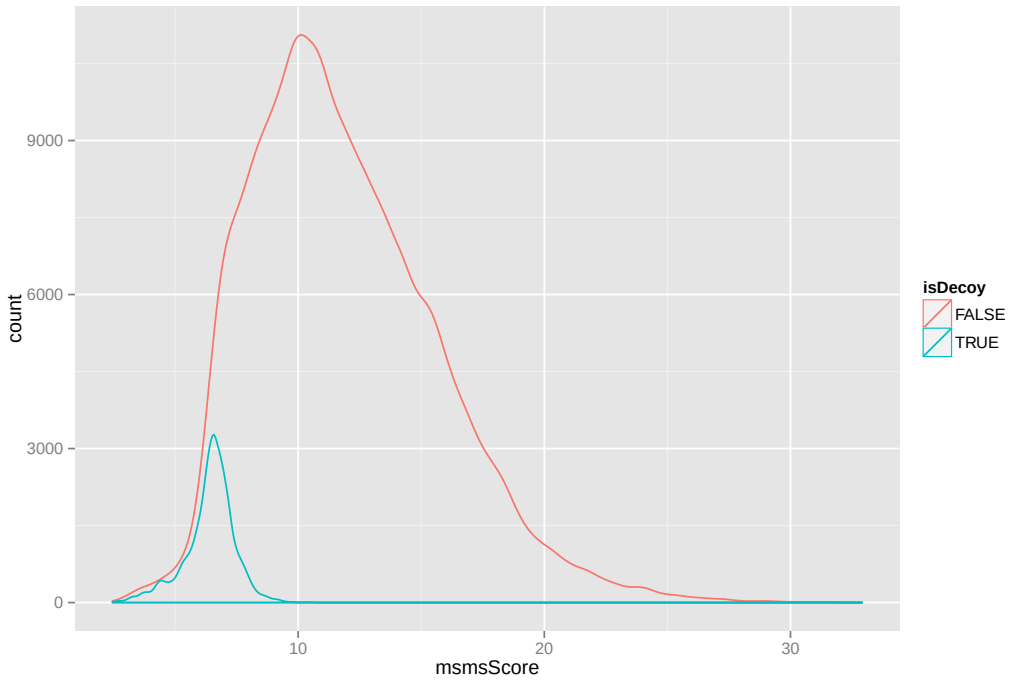
```
> msnid <- recalibrate(msnid.chopped)
> ppm <- mass_measurement_error(msnid)
> ggplot(as.data.frame(ppm), aes(x=ppm)) + geom_histogram(binwidth=0.25)
```



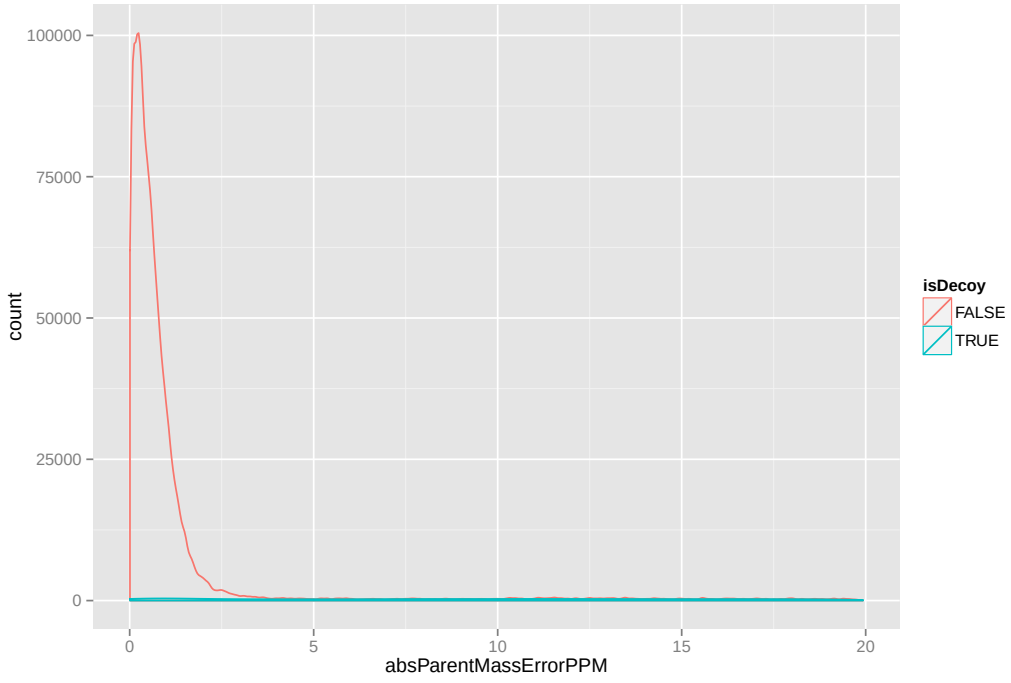
2.9 Defining and optimizing filters for MS/MS identifications

First, let's define criteria we are going to use for data filtration. First will be based on MS-GF+ Spectrum E-value, second will be absolute mass measurement error of the parent ion.

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




```
> 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 what to retain, and a threshold value.

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

An object of calss "MSnIDFilter"
Filter as string:
(absParentMassErrorPPM < 10) & (msmsScore > 10)

> # evaluate filter
> evaluate_filter(msnid, filtObj, level="PSM")

$fdr
[1] 1.604441e-05

$n
[1] 62328

> evaluate_filter(msnid, filtObj, level="Peptide")

$fdr
[1] 0.0002017349

$n
[1] 4958

> evaluate_filter(msnid, filtObj, level="Accession")

$fdr
[1] 0.000617284

$n
[1] 1621
```

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 given FDR upper limit.

```
> # brute-force optimization by enumeration all the parameter combinations
> # these should be good starting parameters for follow-up optimizations
> system.time({
+    filtObj.grid <- optimize_filter(filtObj, msnid, fdr.max=0.01,
+                                    method="Grid", level="Peptide", n.iter=500)})

   user  system elapsed
  0.300   0.074   0.373

> show(filtObj.grid)
```

```
An object of calss "MSnIDFilter"
Filter as string:
(absParentMassErrorPPM < 3.5) & (msmsScore > 8)
```

```
> # (absParentMassErrorPPM < 2) & (msmsScore > 7.8)
>
> # Fine tuning. Nelder-Mead optimization
> system.time({
+    filtObj.nm <- optimize_filter(filtObj.grid, msnid, fdr.max=0.01,
+                            method="Nelder-Mead", level="Peptide", n.iter=500)})
```

```
   user  system elapsed
  0.290   0.068   0.359
```

```
> show(filtObj.nm)
```

```
An object of calss "MSnIDFilter"
Filter as string:
(absParentMassErrorPPM < 3.6) & (msmsScore > 7.8)
```

```
> # (absParentMassErrorPPM < 3) & (msmsScore > 7.8)
```

Evaluating filter based on a good guess and optimization. Finally we'll apply optimized filter to proceed with further steps in the pipeline.

```
> # comparing orignal and optimized filters
> evaluate_filter(msnid, filtObj, level="Peptide")
```

```
$fdr
[1] 0.0002017349

$n
[1] 4958
```

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

```
$fdr
[1] 0.009967295

$n
[1] 6485
```

```
> # apply the optimized filter
> msnid <- apply_filter(msnid, filtObj.nm)
> show(msnid)
```

```
MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 82987 at 0.13 % FDR
#Peptides: 6485 at 1 % FDR
#Accessions: 1979 at 4.4 % FDR
```

Identifications that matched to 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.

```
> # let's remove reverse/decoy and Contaminants
> msnid <- apply_filter(msnid, "!isDecoy")
> show(msnid)

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 82878 at 0 % FDR
#Peptides: 6421 at 0 % FDR
#Accessions: 1895 at 0 % FDR

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

MSnID object
Working directory: "."
#Spectrum Files:  10
#PSMs: 82781 at 0 % FDR
#Peptides: 6409 at 0 % FDR
#Accessions: 1889 at 0 % FDR
```

2.10 Downstream quantitative analysis

The *MSnID* package is aimed at providing convenience functionality to handle MS/MS identifications. Quantitation *per se* is outside of the scope of the package. The only type of quantitation that can be seamlessly blended with MS/MS identification analysis is so-called *spectral counting* approah. 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 *MSnSet* object form *MSnbase* that extends generic Bioconductor *eSet* to quantitative proteomics data.

```
> msnset <- as(msnid, "MSnSet")
> # Note, feature data is peptide-centric. Peptide to protein assigments
> # stored in feature data.
> head(fData(msnset))

                                                                    Peptide
-.APGASAPAASR.S                                                -.APGASAPAASR.S
-.APPSQDVLKEIFNLYDEELDGK.I                                -.APPSQDVLKEIFNLYDEELDGK.I
-.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A -.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A
-.APPTFADLGK.S                                              -.APPTFADLGK.S
-.GFQNLWFSHPR.K                                             -.GFQNLWFSHPR.K
-.GIDINHKHDR.V                                              -.GIDINHKHDR.V
                                          Accession
-.APGASAPAASR.S                              CE25901
-.APPSQDVLKEIFNLYDEELDGK.I            CE30652, CE01236
-.APPSQDVLKEIFNLYDEELDGKIDGTQVGDVAR.A CE30652, CE01236
```

```
-.APPTFADLGK.S                                  CE29443
-.GFQNLWFSHPR.K                                 CE26849
-.GIDINHKHDR.V                                  CE16650

> # Note, sample names in msnset are based on file names that were used as input
> # MS/MS search engine. Let's trim the file names to make them compatible with
> # dataset names.
> head(sampleNames(msnset))

[1] "c_elegans_A_3_1_21Apr10_Draco_10-03-04_dta.txt"
[2] "c_elegans_A_3_3_21Apr10_Draco_10-03-04_dta.txt"
[3] "c_elegans_B_2_1_21Apr10_Draco_10-03-05_dta.txt"
[4] "c_elegans_B_2_3_21Apr10_Draco_10-03-05_dta.txt"
[5] "c_elegans_C_1_1_21Apr10_Draco_10-03-06_dta.txt"
[6] "c_elegans_C_1_3_21Apr10_Draco_10-03-06_dta.txt"

> head(meta)

                       PNNL.Dataset.Name Letter.Replicate Number.Replicate Diet
1  c_elegans_A_3_1_21Apr10_Draco_10-03-04                A                1   ff
3  c_elegans_A_3_3_21Apr10_Draco_10-03-04                A                1   ff
9  c_elegans_B_2_1_21Apr10_Draco_10-03-05                B                2   ff
11 c_elegans_B_2_3_21Apr10_Draco_10-03-05                B                2   ff
17 c_elegans_C_1_1_21Apr10_Draco_10-03-06                C                3   ff
19 c_elegans_C_1_3_21Apr10_Draco_10-03-06                C                3   ff
   Daf.16.type   Age Submitted.Dataset.Name
1          mut young            Y-ctrl-FF.1
3           wt young            Y-daf2-FF.1
9          mut young            Y-ctrl-FF.2
11          wt young            Y-daf2-FF.2
17         mut young            Y-ctrl-FF.3
19          wt young            Y-daf2-FF.3

> # Update sample names.
> # Retain only dataset name portion from spectrum file names,
> # that were used as sample names
> sampleNames(msnset) <- sub("(.*)_dta\\.txt","\\1",sampleNames(msnset))
> show(msnset)

MSnSet (storageMode: lockedEnvironment)
assayData: 6409 features, 10 samples
  element names: exprs
protocolData: none
phenoData: none
featureData
  featureNames: -.APGASAPAASR.S -.APPSQDVLKEIFNLYDEELDGK.I ... R.YYYDHSKK.H
    (6409 total)
  fvarLabels: Peptide Accession
  fvarMetadata: labelDescription
```

```
experimentData: use 'experimentData(object)'
Annotation:
- - - Processing information - - -
 MSnbase version: 1.11.9
```

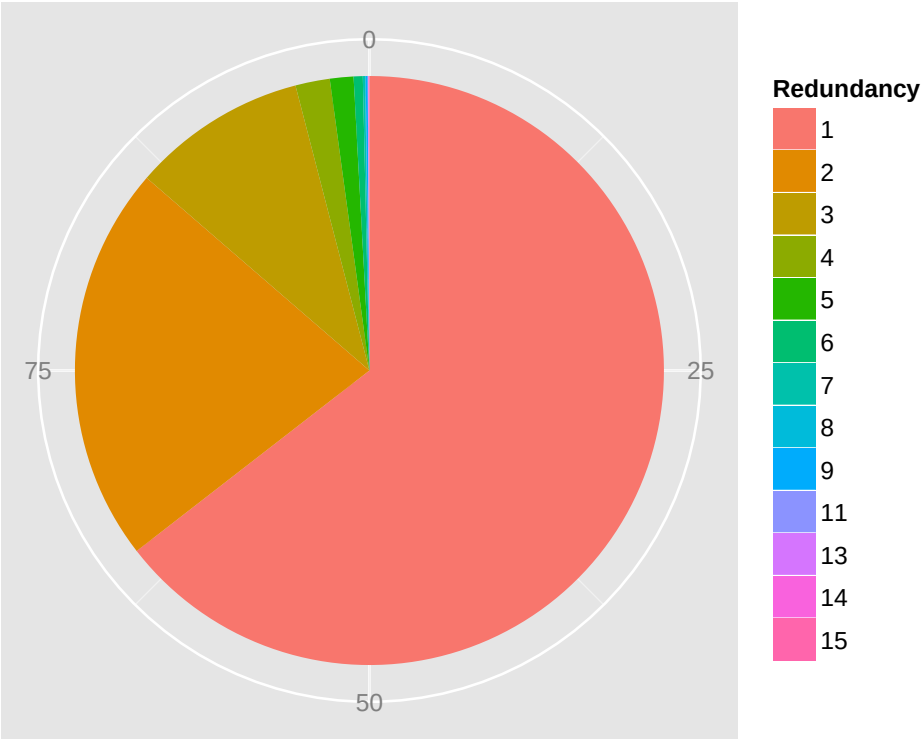
```
> # prepare pheno data
> rownames(meta) <- meta$PNNL.Dataset.Name
> meta <- meta[,c("Letter.Replicate","Daf.16.type")]
> pData(msnset) <- meta[sampleNames(msnset),]
> validObject(msnset)
```

```
[1] TRUE
```

Currently the features in the *MSnSet* object are peptides. However, we plan to quantify differences at level of relative protein abundances. Peptides can be rolled-up to protein level using `combineFeatures` function from *MSnbase*. Ambiguity in peptide-to-protein assignment is handled by `redundancy.handler` argument. It can allow ambiguity of peptide assignment to *multiple* proteins or *unique* mapping only by discarding peptide mapping two or more proteins.

```
> # assessing the extent of peptide/protein mapping redundancy problem
> redundancy <- table(sapply(fData(msnset)$Accession, length), dnn="redundancy")
> redundancy <- 100 * prop.table(redundancy)
> ggplot(as.data.frame(redundancy), aes(x=factor(1), y=Freq, fill=redundancy)) +
+    geom_bar(stat='identity', width=1) +
+    coord_polar(theta='y') +
+    xlab('') + ylab('') +
+    labs(fill='Redundancy') +
+    scale_x_discrete(breaks = NULL)
> # original number of features/peptides in the data
> length(featureNames(msnset))
```

```
[1] 6409
```



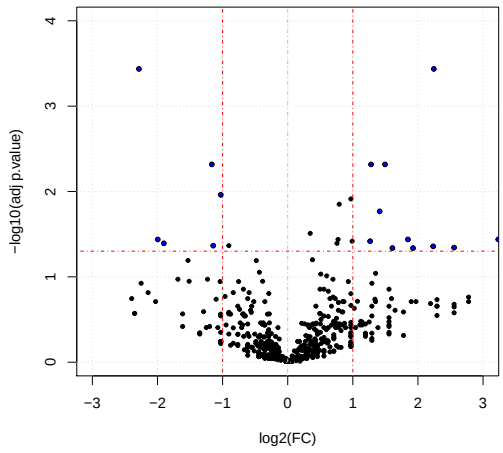

```
> # summing allowing peptide to contribute multiple proteins
> msnset.prot.1 <- combineFeatures(msnset, fData(msnset)$Accession,
+                                  redundancy.handler="multiple",
+                                  fun="sum", cv=FALSE)
> # summing using only non-redundant peptides.
> # Note, in this case 1/3 of peptides will be discarded
> msnset.prot.2 <- combineFeatures(msnset, fData(msnset)$Accession,
+                                  redundancy.handler="unique",
+                                  fun="sum", cv=FALSE)
> # Subset to proteins that present in at least 6 samples out of 10.
> msnset <- msnset.prot.2[rowSums(exprs(msnset.prot.2) > 0) >= 6,]
> # check how many proteins left
> length(featureNames(msnset))
```

```
[1] 567
```

For statistical testing we will leverage functionality available in *msmsTests*. Here we are comparing the difference between *glp-4; daf-2* long-living strain with a strain that bears compensatory mutation within *daf-16*

transcription factor. For statistical testing of the count data we will use quasi-Poisson model `msms.glm.qlll`.

```
> library("msmsTests")
> alt.f <- "y ~ Daf.16.type + 1"
> null.f <- "y ~ 1"
> div <- colSums(exprs(msnset))
> res <- msms.glm.qlll(msnset, alt.f, null.f, div=div)
> sum(p.adjust(res$p.value, "BH") < 0.05)

[1] 26

> # Post-test filter
> lst <- test.results(res,msnset,pData(msnset)$Daf.16.type,"wt","mut",div,
+                     alpha=0.05,minSpC=0,minLFC=1,
+                     method="BH")
> res.volcanoplot(lst$tres, min.LFC=1, max.pval=0.05, ylbls=NULL, maxy=4)
```



References

Geert Depuydt, Fang Xie, Vladislav A. Petyuk, Nilesh Shanmugam, Arne Smolders, Ineke Dhondt, Heather M. Brewer, David G Camp, 2nd, Richard D. Smith, and Bart P. Braeckman. Reduced insulin/insulin-like growth factor-1 signaling and dietary restriction inhibit translation but preserve muscle mass in caenorhabditis elegans. *Mol Cell Proteomics*, 12(12):3624–3639, Dec 2013. doi: 10.1074/mcp.M113.027383. URL `http://dx.doi.org/10.1074/mcp.M113.027383`.

Geert Depuydt, Fang Xie, Vladislav A. Petyuk, Arne Smolders, Heather M. Brewer, David G Camp, 2nd, Richard D. Smith, and Bart P. Braeckman. Lc-ms proteomics analysis of the insulin/igf-1-deficient caenorhabditis elegans daf-2(e1370) mutant reveals extensive restructuring of intermediary metabolism. *J Proteome Res*, Mar 2014. doi: 10.1021/pr401081b. URL `http://dx.doi.org/10.1021/pr401081b`.