specL - Prepare Peptide Spectrum Matches for Use in Targeted Proteomics

Christian Panse*
Christian Trachsel†
Jonas Grossmann‡
Witold Wolski§

May 15, 2016

*cp@fgcz.ethz.ch
†christian.trachsel@fgcz.ethz.ch
‡jg@fgcz.ethz.ch
§wewolski@gmail.com

Contents

1 Introduction

Targeted proteomics is a fast evolving field in proteomics science and was even elected as method of the year in 2012 [1]. Especially targeted methods like SWATH [**?**] open promising perspectives for the identification and quantification of peptides and proteins. All targeted methods have in common the need of precise MS coordinates composed of precursor mass, fragment masses, and retention time. The combination of this information is kept in so called assays or spectra libraries. Here we present an R package able to produce such libraries out of peptide identification results (Mascot (dat), TPP (pep.xml and mzXMLs), ProteinPilot (group), Omssa (omx)). *specL* (see also [**?**]) is an easy to use, versatile and flexible function, which can be integrated into already existing commercial or non commercial analysis pipelines for targeted proteomics data analysis. Some examples of today's pipelines are ProteinPilot combined with Peakview (AB Sciex), Spectronaut (Biognosys) or OpenSwath [**?**].

In the following vignette it is described how the *specL* package can be used for the included data sets `peptideStd` and `peptideStd.redundant`.

2 Workflow

2.1 Prologue - How to get the input for the specL package?

Since peptide identification (using, e.g., Mascot, Sequest, xTandem!, Omssa, ProteinPilot) usually creates result files which are heavily redundant and therefore unsuited for spectral library building, the search results must first be filtered. To create non-redundant input files, we use the BiblioSpec [**?**] algorithm implemented in Skyline [**?**]. A given search result (e.g. Mascot result file) is loaded into the software Skyline and is redundancy filtered. The 'Skyline workflow step' provides two sqlite readable files[2] as output named `'*.blib'` and `'*.redundant.blib'`. These files are used as ideal input for this packages. Note here, that Skyline is very flexible when it comes to peptide identification results. It means with Skyline you can build the spectrum library files for almost all search engines (even from other spectrum library files such as spectraST [**?**]).

The first step which has to be performed on the R shell is loading *specL* library.

```
> library(specL)
> packageVersion('specL')
```

```
[1] âĂŸ1.6.2âĂŹ
```

2.2 Read from redundant plus non-redundant blib files

for demonstration *specL* contains the two data sets namely `peptideStd` and `peptideStd.redundant`. This is a data set which comes from two standard run experiments which are routinely used to check

[1] http://www.nature.com/nmeth/journal/v10/n1/pdf/nmeth.2329.pdf, 2014-09-22
[2] sqlite uses standart SQL as query language.

if the liquid chromatographic system is still working appropriate. The sample consists of a digest of the Fetuin protein (Bos taurus, uniprot id: P12763). 40 femtomole are loaded on column. Mascot was used to search and identify the respective peptides.

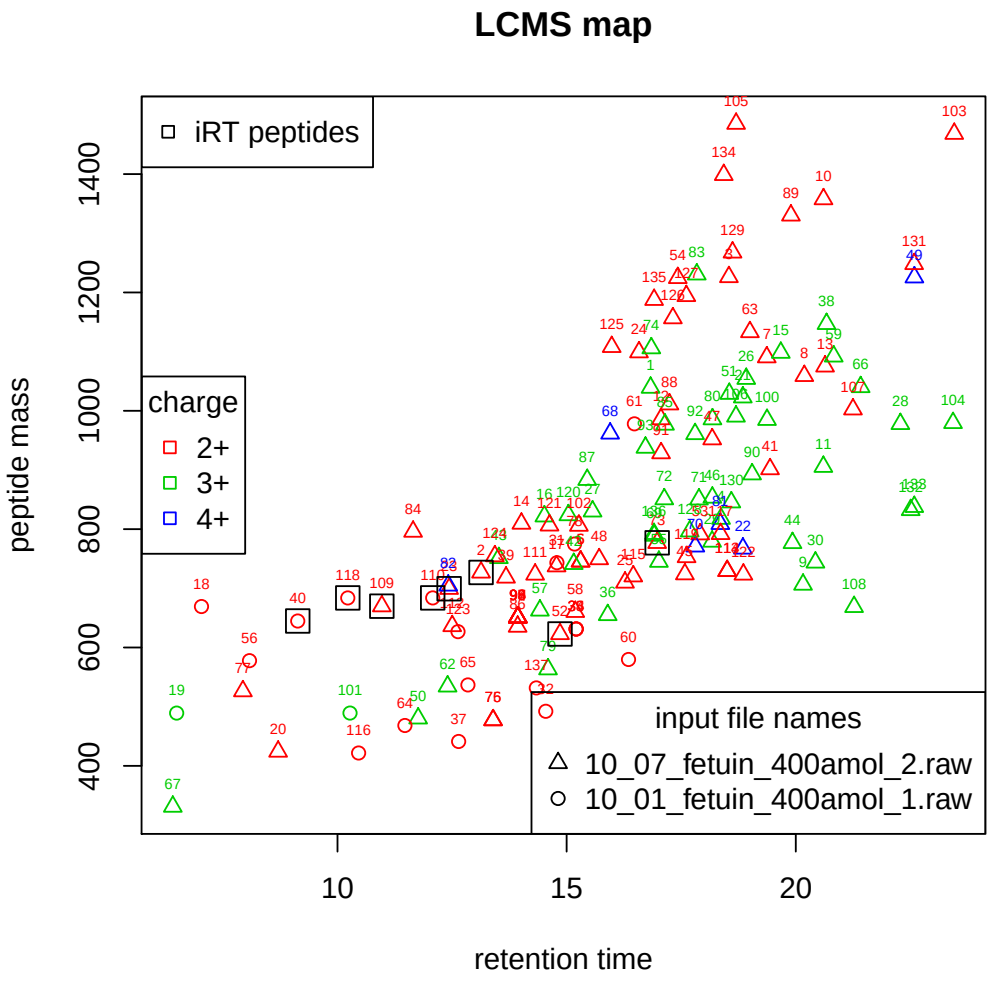
```
> summary(peptideStd)
```

```
Summary of a "psmSet" object.
Number of precursor:
        137
Number of precursors in Filename(s)
        _methods\20140910_01_fetuin_400amol_1.raw        21
        _methods\20140910_07_fetuin_400amol_2.raw        116
Number of annotated precursor:
        0
```

For both `peptideStd`, `peptideStd.redandant` data sets the Skyline software was used to generate the bibliospec files which contain the peptide sequences with the respective peptide spectrum match (PSM). The `specL::read.bibliospec` function was used to read the blib files into R.

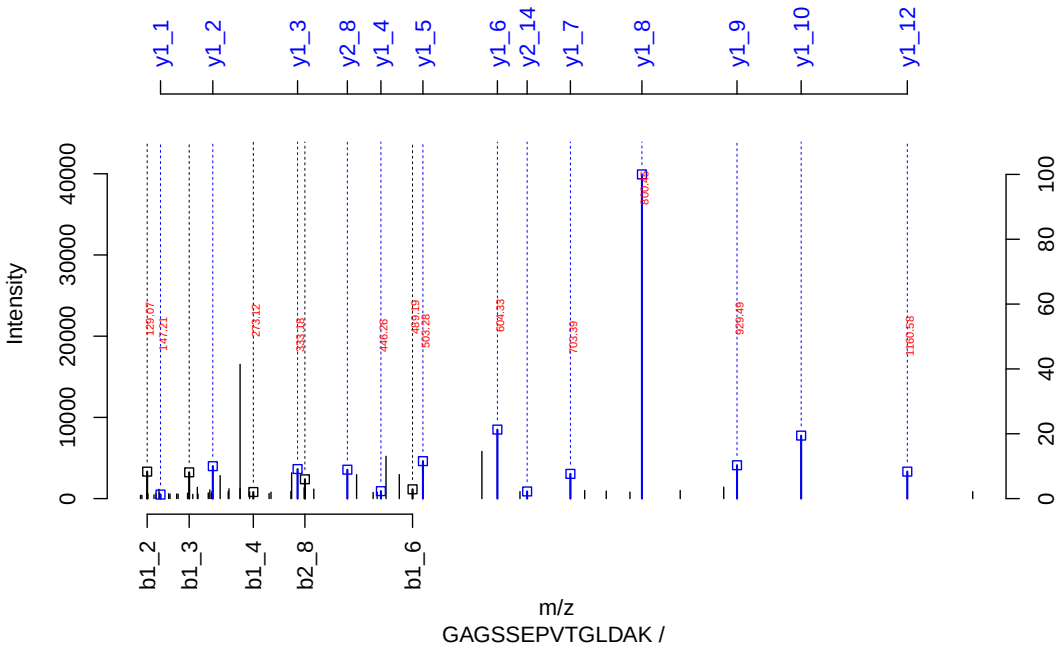
The from `read.bibliospec` generated object has its own plot functions. The LC-MS map graphs peptide mass versus retention time.

```
> plot(peptideStd)
```



The individual peptide spectrum match (psm) is displayed by using the *protViz* `peakplot` function.

```
> demoIdx <- 40
> # str(peptideStd[[demoIdx]])
> res <- plot(peptideStd[[demoIdx]], ion.axes=TRUE)
```



2.3 Read from Mascot result files

Alternatively Mascot search result files (dat) can be used by applying *protViz* perl script `protViz_mascotDat2RData.pl`.

The perl script can be found in the exec directory of the *protViz* package. The mascot mod_file can be found in the configurations of the mascot server. An example on our Linux shell looks as follow:

```
$ /usr/local/lib/R/site-library/protViz/exec/protViz_mascotDat2RData.pl \
-d=/usr/local/mascot/data/20130116/F178287.dat \
-m=mod_file
```

`mascotDat2RData.pl` requires the Mascot server `mod_file` keeping all the configured modification.

Once the perl script is finished, the resulting RData file can be read into the R session using `load`.

Next, the variable modifications, and the S3 psmSet object has to be generated. This can be done by using `specL:::.mascot2psmSet`

```
> specL:::.mascot2psmSet

function (dat, mod, mascotScoreCutOff = 40)
{
    res <- lapply(dat, function(x) {
        x$MonoisotopicAAmass <- protViz::aa2mass(x$peptideSequence)[[1]]
        modString <- as.numeric(strsplit(x$modification, "")[[1]])
        modIdx <- which(modString > 0) - 1
        modString.length <- length(modString)
        x$varModification <- mod[modString[c(-1, -modString.length)] +
```

```
            1]
        if (length(modIdx) > 0) {
            warning("modified varModification caused.")
            x$varModification[modIdx] <- x$varModification[modIdx] -
                x$MonoisotopicAAmass[modIdx]
        }
        rt <- x$rtinseconds
        x <- c(x, rt = rt, fileName = "mascot")
        class(x) <- "psm"
        return(x)
    })
    res <- res[which(unlist(lapply(dat, function(x) {
        x$mascotScore > mascotScoreCutOff && length(x$mZ) > 10
    })))]
    class(res) <- "psmSet"
    return(res)
}
<environment: namespace:specL>
```

If you are processing Mascot result files you can continue reading in section **??**.

However, please note, due do the potential high redundancy of peptide spectrum matches in a database search approach, it might not result in useful ion library for targeted data extraction, unless redundancy filtering is handled. However in a future release a redundancy filter algorithm might be proposed to resolve this problem.

2.4 Annotate protein ids using FASTA

The information to which protein a peptide-spectrum-match belongs (PSM) is not stored by BiblioSpec. Therefore *specL* provides the `annotate.protein_id` function which uses R's internal `grep` to 'reassigning' the protein information. Therefore a `fasta` object has to be loaded into the R system using `read.fasta` of the *seqinr* package. For this, not necessarily, the same `fasta` file needs to be provided as in the original database search.

The following lines demonstrate a simple sanity check with a single FASTA style formated protein entry. Also it demonstrates the use case how to identify entries in the R-object which are from one or a few proteins of interest.

```
> irtFASTAseq <- paste(">zz|ZZ_FGCZCont0260|",
+ "iRT_Protein_with_AAAAK_spacers concatenated Biognosys\n",
+ "LGGNEQVTRAAAAKGAGSSEPVTGLDAKAAAAKVEATFGVDESNAKAAAAKYILAGVENS",
+ "KAAAAKTPVISGGPYEYRAAAAKTPVITGAPYEYRAAAAKDGLDAASYYAPVRAAAAKAD",
+ "VTPADFSEWSKAAAAKGTFIIDPGGVIRAAAAKGTFIIDPAAVIRAAAAKLFLQFGAQGS",
+ "PFLK\n")
> Tfile <- file();  cat(irtFASTAseq, file = Tfile);
```

```
> fasta.irtFASTAseq <- read.fasta(Tfile, as.string=TRUE, seqtype="AA")
> close(Tfile)
```

As expected the peptideStd data, e.g., our demo Object, does not contain any protein information yet.

```
> peptideStd[[demoIdx]]$proteinInformation
```

```
[1] ""
```

The protein information can be added as follow:

```
> peptideStd <- annotate.protein_id(peptideStd,
+     fasta=fasta.irtFASTAseq)
```

The following lines show now the object indices of those entries which do hav a protein information now.

```
> (idx<-which(unlist(lapply(peptideStd,
+     function(x){nchar(x$proteinInformation)>0}))))
```

```
[1] 1 2 3 4 5 6
```

As expected, there are now a number of peptide sequences annotated with the protein ID.

```
> peptideStd[[demoIdx]]$proteinInformation
```

```
[1] "zz|ZZ_FGCZCont0260|"
```

Please note, that the default digest pattern is defined as

```
> digestPattern = "(([RK])|(^)|(^M))"
```

for tryptic peptides. For other enzymes, the pattern has to be adapted. For example, for semi-tryptic identifications use `digestPattern = ""`.

2.5 Generate the spectral library (assay)

genSwathIonLib is the main contribution of the specL package. It generates the spectral library used in a targeted data extraction workflow from a mass spectrometric measurement. Generating the ion library using iRT peptides is highly recommended as described. However if you have no iRT peptide continue reading in section **??**.

Generation of the spec Library with default (see Table **??**) settings.

```
> res.genSwathIonLib <- genSwathIonLib(data=peptideStd,
+                                      data.fit=peptideStd.redundant)
```

```
> summary(res.genSwathIonLib)
```

```
Summary of a "specLSet" object.

Parameter:
```

parameter	description	value
`max.mZ.Da.error`	max ms2 tolerance	`0.1`
`topN`	the n most intense fragment ion	10
`fragmentIonMzRange`	mZ range filter of fragment ion	c(300, 1800)
`fragmentIonRange`	min/max number of fragment ions	`c(5,100)`
`fragmentIonFUN`	desired fragment ion types	`b1+,y1+,b2+,y2+,b3+,y3+`

Table 1: `genSwathIonLib` default settings

```
        mascotIonScoreCutOFF=20
        proteinIDPattern=
        max.mZ.Da.error=0.1
        ignoreMascotIonScore=TRUE
        topN=10
        fragmentIonMzRange=300
         fragmentIonMzRange=1800
        fragmentIonRange=4
         fragmentIonRange=100

Number of precursor (q1 and peptideModSeq) = 137
Number of unique precursor
(q1.in-silico and peptideModSeq) = 126
Number of iRT peptide(s) = 8
Number of transitions frequency:
        4        1
        5        5
        6        10
        7        7
        8        18
        9        32
        10        64

Number of annotated precursor = 6
Number of file(s)
        2

Number of precursors in Filename(s)
        _methods\20140910_01_fetuin_400amol_1.raw        21
        _methods\20140910_07_fetuin_400amol_2.raw        116

Misc:

Memory usage        =         625928 bytes
```

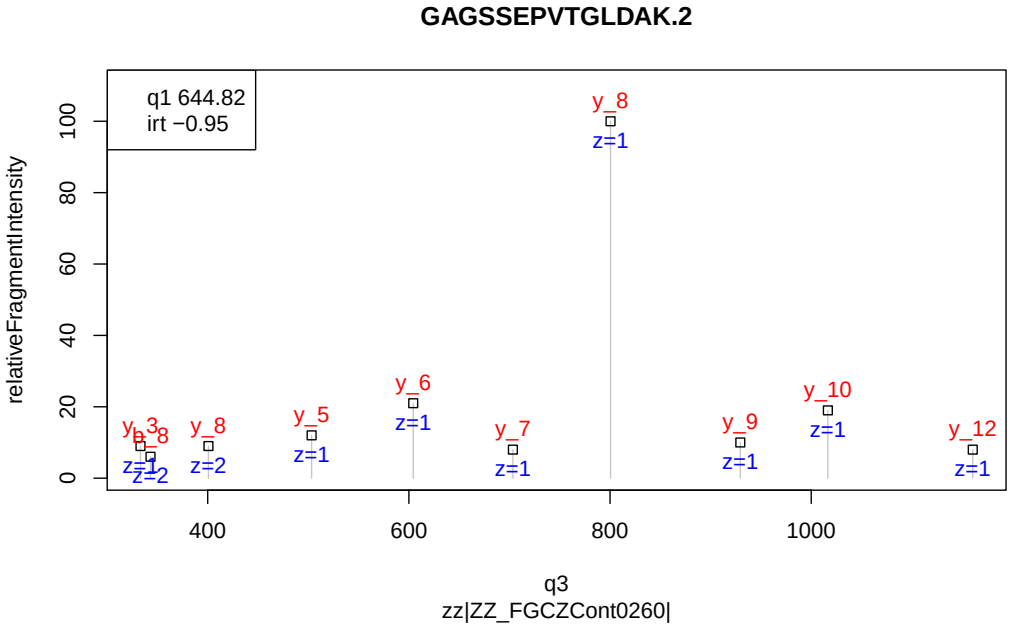
The determined mass spec coordinates of the selected tandem mass spectrum `demoIdx` look like this:

```
> res.genSwathIonLib@ionlibrary[[demoIdx]]
```

```
An "specL" object.


content:
group_id = GAGSSEPVTGLDAK.2
peptide_sequence = GAGSSEPVTGLDAK
proteinInformation = zz|ZZ_FGCZCont0260|
q1 = 644.8219
q1.in_silico = 1288.638
q3 = 800.4497 604.3285 1016.522 503.2805 929.4925 400.7282 333.176 1160.581
703.3948 343.1235
q3.in_silico = 800.4512 604.3301 1016.526 503.2824 929.4938 400.7295 333.1769
1160.579 703.3985 343.1615
decoy = NA NA NA NA NA NA NA NA NA NA
prec_z = 2
frg_type = y y y y y y y y y b
frg_nr = 8 6 10 5 9 8 3 12 7 8
frg_z = 1 1 1 1 1 2 1 1 1 2
relativeFragmentIntensity = 100 21 19 12 10 9 9 8 8 6
irt = -0.95
peptideModSeq = GAGSSEPVTGLDAK
mZ.error = 0.001514 0.00156 0.003685 0.001914 0.001318 0.001313 0.000856
0.001846 0.003686 0.0380015
\ctrachse_20140910_Nuclei_diff_extraction_methods\20140910_01_fetuin_400amol_1.raw
size:
Memory usage: 4672 bytes
```

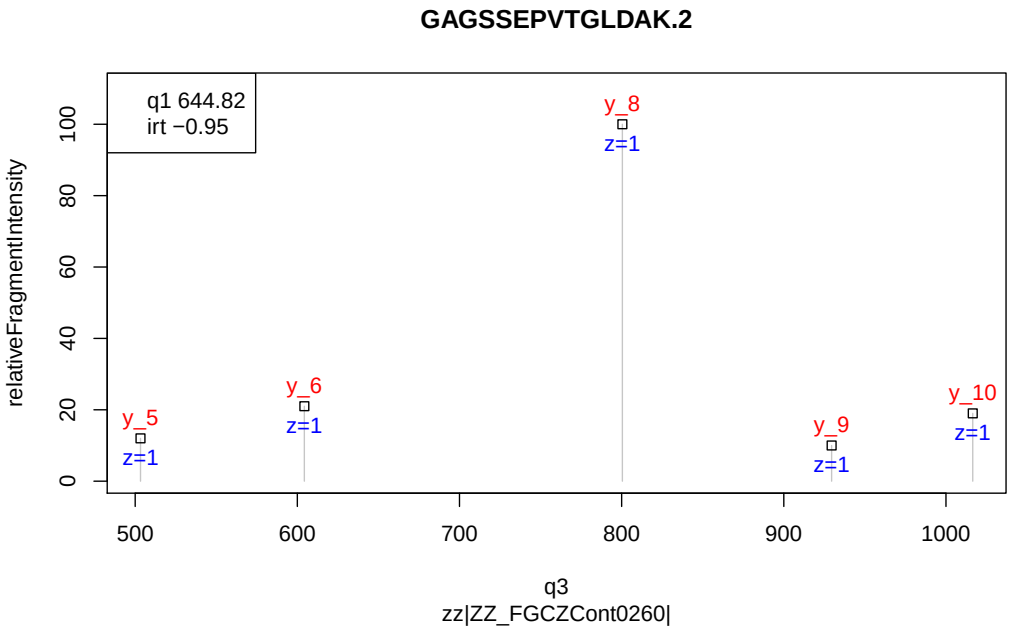
It can be displayed using the `specL::plot` function.

```
> plot(res.genSwathIonLib@ionlibrary[[demoIdx]])
```



The following code considers only the top five y ions.

```
> # define customized fragment ions
> # for demonstration lets consider only the top five singly charged y ions.
>
> r.genSwathIonLib.top5 <- genSwathIonLib(peptideStd,
+     peptideStd.redundant, topN=5,
+     fragmentIonFUN=function (b, y) {
+       return( cbind(y1_=y) )
+       }
+     )
> plot(r.genSwathIonLib.top5@ionlibrary[[demoIdx]])
>
```



2.6 Normalizing the retention time using iRT peptides

Retention time is a very important parameter in targeted data extraction. However retention times are not easy to transfer between different reverse phase columns or HPLC systems. To make transfer applicable and account for inter run shift in retention time Biognosys [**?**] invented the iRT normalization based on iRT / HRM peptides. For this, a set of well behaving peptides (good flying properties, good fragmention characteristics, completely artifical) which cover the whole rt-gradient are spiked into each sample. For this set of peptide an idependent retention time (dimension less) is suggested by Biognosys. With this at hand, the set of peptides can later be used to apply a linear regression model to adapt all measured retention times into an independent retention time scale.
If the identification results contain iRT peptides the package supports the conversion to the iRT scale. For this (if the identification result are based on multiple input files) the redundant BiblioSpec file is required where all iRT peptides from all measurements are stored. For the most representative spectrum in the non-redundant R-object the original filename is identified and the respective linear model for this one particular MS experiment is applied to normalize the retention time to the iRT scale. The iRT peptides as well as their independent retention times are stored in the `iRTpeptides` object.

specL uses by default the iRT peptide table to normalize into the independent retention time but could also be extended or changed to custom iRT peptides if available.

```
> iRTpeptides
```

The method genSwathIonLib uses:

```
> fit <- lm(formula = rt ~ aggregateInputRT * fileName, data=m)
>
```

to build the linear models for each MS measurement individually. For defining `m` both data sets were aggregated over the attributes `peptide` and `fileName` using the `mean` operator.

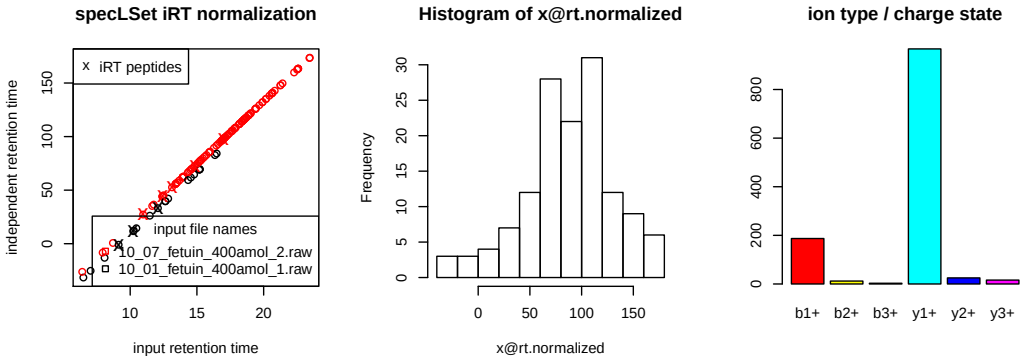
```
> data<-aggregate(df$rt, by=list(df$peptide, df$fileName), FUN=mean)
> data.fit<-aggregate(df.fit$rt,
+                     by=list(df.fit$peptide, df.fit$fileName),
+                     FUN=mean)
```

Afterwards the following join operator was applied.

```
> m <- merge(iRT, data.fit, by.x='peptide', by.y='peptide')
```

The following graph displays the normalized retention time versus the measured retention time after applying the calculated model to the two data sets.

```
> # calls the plot method for a specLSet object
> op<-par(mfrow=c(2,3))
> plot(res.genSwathIonLib)
> par(op)
```



Shown are original retention time (in minutes) and iRT (dimensionless) for two standard run experiments (color black and red). Indicated with black **X** are the iRT peptides which are the base for the regression.

2.7 Generate the spectral library having no iRTs

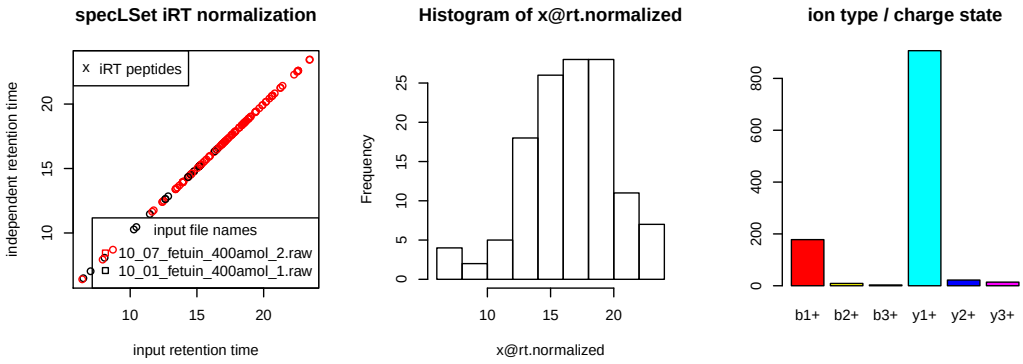
If no iRT peptides are contained in the data not iRT normalization is applied. The scatter plot below shows on the y axis that there were not iRT transformation.

```
> idx.iRT<-which(unlist(lapply(peptideStd, function(x){
+   if(x$peptideSequence %in% iRTpeptides$peptide){0}else{1}
+   })) == 0)
> # remove all iRTs and compute ion library
> res.genSwathIonLib.no_iRT <- genSwathIonLib(peptideStd[-idx.iRT])
> summary(res.genSwathIonLib.no_iRT)
```

```
Summary of a "specLSet" object.

Parameter:
        mascotIonScoreCutOFF=20
        proteinIDPattern=
        max.mZ.Da.error=0.1
        ignoreMascotIonScore=TRUE
        topN=10
        fragmentIonMzRange=300
         fragmentIonMzRange=1800
        fragmentIonRange=4
         fragmentIonRange=100

Number of precursor (q1 and peptideModSeq) = 129
Number of unique precursor
(q1.in-silico and peptideModSeq) = 118
Number of iRT peptide(s) = 0
Number of transitions frequency:
        4       1
        5       5
        6       10
        7       7
        8       17
        9       31
        10       58

Number of annotated precursor = 0
Number of file(s)
        2

Number of precursors in Filename(s)
        _methods\20140910_01_fetuin_400amol_1.raw       18
        _methods\20140910_07_fetuin_400amol_2.raw       111
```

```
Misc:

Memory usage         =         580968 bytes
```

```
> op<-par(mfrow=c(2,3))
> plot(res.genSwathIonLib.no_iRT)
> par(op)
```



2.8 Write output to file

The output can be written as an ASCII text file.

```
> write.spectronaut(res.genSwathIonLib, file="specL-Spectronaut.txt")
```

2.9 Epilogue - What can I do with that library now?

The specL output text file can directly be used as input (assay) for the Spectronaut software from Biognosys or with minimal reshaping for Peakview. Alternatively it can be used as a basis for script based construction of SRM/MRM assays.

2.10 Benchmark

The benchmark were processed on a 12 core XEON Server (X5650 @ 2.67GHz) running Linux Debian wheezy having R version 3.1.1 (2014-07-10) , specL 1.1.2, and BiocParallel 1.0.0 installed. The default setting of BiocParallel has used eight cores. As FASTA we used a TAIR10 retrived from http://www.arabidopsis.org/ and Human Swissprot.

fasta=TAIR10			blib [unpublished]		runtime	
#proteins	#tryptic peptides	file size	#specs	file size	annotate	generate
71032	3423196	39M	39648/118268	51M	79min	19sec
71032	3423196	39M	65018/136963	120M	130min	30sec
fasta=HUMAN			blib [**?**, Rosenberger]			
88969	3997085	43M	256908/3060421	4.4G	≈7h	≈5min

The following parameter settings were given to the `genSwathIonLib` function:

```
> res <- genSwathIonLib(data, data.fit,
+   topN=10,
+   fragmentIonMzRange=c(200,2000),
+   fragmentIonRange=c(2,100))
```

3 Acknowledgement

The authors thank all colleagues of the Functional Genomics Center Zuerich (FGCZ), and especial thank goes to our test users Sira Echevarría Zomeño (Swiss Federal Institute of Technology in Zurich (ETHZ)), Tobias Kockmann (Swiss Federal Institute of Technology in Zurich (ETHZ)), Lukas von Ziegler (Brain Research Institute, UZH|ETH Zurich), and Stephan Michalik (Ernst-Moritz-Arndt-Universität Greifswald, Germany).

4 To Do for next releases

- importer for peakview csv format; enable `compare.specLSet(object0, object1)`
- new option for `specL::genSwathIonLib`; Exclude fragment ions from precursor `window = TRUE, FALSE`
- new option for `specL::genSwathIonLib`; Predict transitions for heavy labeled peptides using information from light peptides `predictHeavy = TRUE,FALSE, LabelFile = "fileWithHeavyAA"`
- new export function into TraML format for compatibility with OpenSWATH [**?**]
- replace *seqinr* `read.fasta` by using *Biostrings* `readAAStringSet` to handle fasta files
- add varMods to specL class
- replace Mascot score by a generic score
- in-silico rt ion map plot (`plot.specLSet`) split window into SWATH windows (one plot per, e.g., 25Da window)
- assay refinement - replace contaminated fragment ion in library

5 Session information

An overview of the package versions used to produce this document are shown below.

- R version 3.3.0 (2016-05-03), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: BiocStyle 2.0.2, DBI 0.4-1, RSQLite 1.0.0, ade4 1.7-4, lattice 0.20-33, msqc1 1.0.0, protViz 0.2.15, seqinr 3.1-3, specL 1.6.2
- Loaded via a namespace (and not attached): Rcpp 0.12.5, digest 0.6.9, evaluate 0.9, formatR 1.4, grid 3.3.0, htmltools 0.3.5, knitr 1.13, magrittr 1.5, parallel 3.3.0, rmarkdown 0.9.6, stringi 1.0-1, stringr 1.0.0, tools 3.3.0, yaml 2.1.13