`protViz`: Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics

Christian Panse Jonas Grossmann

November 15, 2012

Vignette for v.0.1.0
t.b.d for the next release:

- assemble proteins from pg feature map using t3pq algorithm
- enable varmods in `fragmentIons` and `peakplot`

Contents

1 Recent changes and updates

None

2 Preliminary Note

`protViz` is an R package to do quality checks, vizualizations and analysis of mass spectrometry data, coming from proteomics experiments. The package is developed, tested and used at the Functional Genomics Center Zurich. We use this package mainly for prototyping, teaching, and having `fun` with proteomics data. But it can also be used to do solid data analysis for small scale data sets.

3 Related Work

The methode of choice in proteomics is mass spectrometry. There are already packages in R which deal with mass spec related data. Some of them are listed here:

- MSnbase package (basic function)
  `http://www.bioconductor.org/packages/release/bioc/html/MSnbase.html`
- plgem – spec counting
  `http://www.bioconductor.org/packages/release/bioc/html/plgem.html`
- synapter – MSe (Top3 Quantification)
  `http://www.bioconductor.org/packages/release/bioc/html/synapter.html`
- mzR
  `http://www.bioconductor.org/packages/release/bioc/html/mzR.html`
- isobar iTRAQ quantification
  `http://www.bioconductor.org/packages/release/bioc/html/isobar.html`
- readMzXmlData
  `http://cran.r-project.org/web/packages/readMzXmlData/`

4 Get Data In – Preprocessing

The most time consuming and challenging part for data analysis and visualization is shaping the data that they can easily be processed.

4.1 In-silico from Proteins to Peptides

For demonstration we use a sequence of peptides derived from a tryptics digest using the Swissprot FETUA_BOVIN Alpha-2-HS-glycoprotein precursor (Fetuin-A) (Asialofetuin) protein.

`fcat` and `tryptic-digest` are commandline programs which are included in the package. `fcat` removes the lines starting with > and all 'new line' character within the protein sequence while `tryptic-digest` is doing the triptic digest of a protein sequence applying the rule: cleave after arginine (R) and lysine (K) except followed by proline(P).

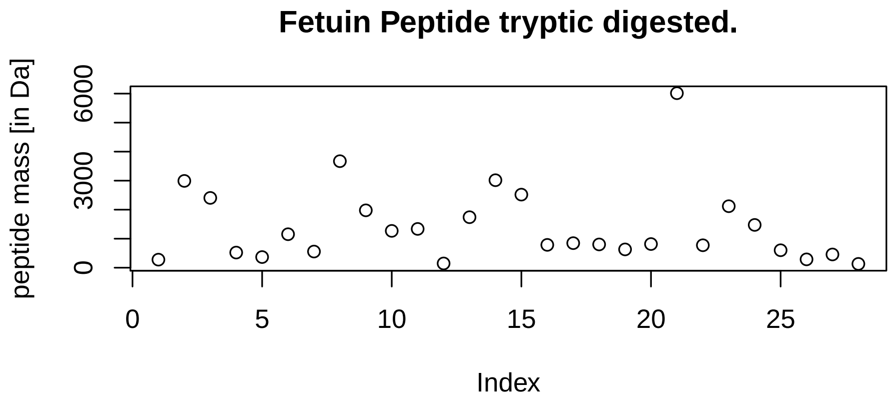
```
$ cat Fetuin.fasta
MKSFVLLFCLAQLWGCHSIPLDPVAGYKEPACDDPDTEQAALAAVDYINKHLPRGYKHTL
NQIDSVKVWPRRPTGEVYDIEIDTLETTCHVLDPTPLANCSVRQQTQHAVEGDCDIHVLK
QDGQFSVLFTKCDSSPDSAEDVRKLCPDCPLLAPLNDSRVVHAVEVALATFNAESNGSYL
QLVEISRAQFVPLPVSVSVEFAVAATDCIAKEVVDPTKCNLLAEKQYGFCKGSVIQKALG
GEDVRVTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAV
PLPLHRAHYDLRHTFSGVASVESSSGEAFHVGKTPIVGQPSIPGGPVRLCPGRIRYFKI

$ cat Fetuin.fasta | fcat | tryptic-digest
MK
SFVLLFCLAQLWGCHSIPLDPVAGYK
EPACDDPDTEQAALAAVDYINK
HLPR
GYK
HTLNQIDSVK
VWPR
RPTGEVYDIEIDTLETTCHVLDPTPLANCSVR
QQTQHAVEGDCDIHVLK
QDGQFSVLFTK
CDSSPDSAEDVR
K
LCPDCPLLAPLNDSR
VVHAVEVALATFNAESNGSYLQLVEISR
AQFVPLPVSVSVEFAVAATDCIAK
EVVDPTK
CNLLAEK
QYGFCK
GSVIQK
ALGGEDVR
VTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR
AHYDLR
HTFSGVASVESSSGEAFHVGK
TPIVGQPSIPGGPVR
LCPGR
IR
YFK
I
```

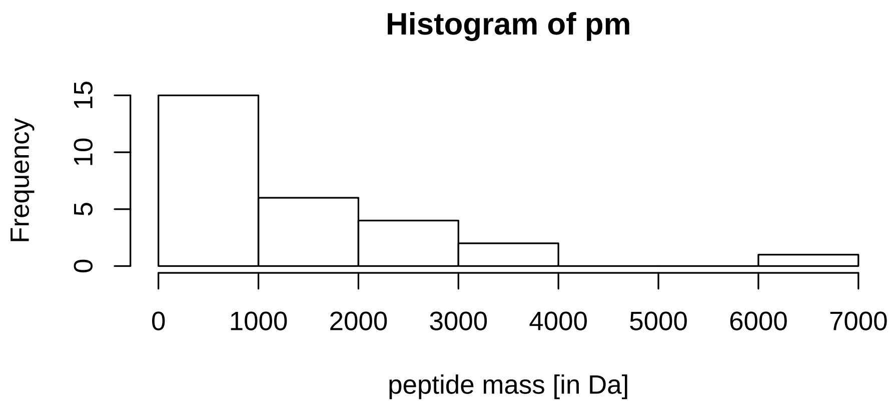
5 Peptide Identification

The currency in proteomics are the peptides. In proteomics, proteins are digested to so-called peptides since peptides are much easier to handle biochemically than proteins. Proteins are very different in nature some are very sticky while others are soluble in aqueous solutions while again are only sitting in membranes. Therefore, proteins are chopped up into peptides because it is fair to assume, that for each protein, there will be a number of peptides behaving well, so that they can actually be measured with the mass spectrometer. This step introduces another problem, the so-called protein inference problem. In this package here, we do not at all touch upon the protein inference.

5.1 Computing the Parent Ion Mass

```
> library(protViz)
> op<-par(mfrow=c(1,1))
> fetuin<-c('MK', 'SFVLLFCLAQLWGCHSIPLDPVAGYK',
+ 'EPACDDPDTEQAALAAVDYINK',
+ 'HLPR', 'GYK', 'HTLNQIDSVK', 'VWPR',
+ 'RPTGEVYDIEIDTLETTCHVLDPTPLANCSVR',
+ 'QQTQHAVEGDCDIHVLK', 'QDGQFSVLFTK',
+ 'CDSSPDSAEDVR', 'K', 'LCPDCPLLAPLNDSR',
+ 'VVHAVEVALATFNAESNGSYLQLVEISR',
+ 'AQFVPLPVSVSVEFAVAATDCIAK',
+ 'EVVDPTK', 'CNLLAEK', 'QYGFCK',
+ 'GSVIQK', 'ALGGEDVR',
+ 'VTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR',
+ 'AHYDLR', 'HTFSGVASVESSSGEAFHVGK',
+ 'TPIVGQPSIPGGPVR', 'LCPGR', 'IR', 'YFK', 'I')
> (pm<-parentIonMass(fetuin))

 [1]  278.1533 2991.5259 2406.0765  522.3147  367.1976 1154.6164  557.3194
 [8] 3671.7679 1977.9447 1269.6474 1337.5274  147.1128 1740.8407 3016.5738
[15] 2519.3214  787.4196  847.4342  802.3552  631.3773  816.4210 6015.1323
[22]  774.3893 2120.0043 1474.8376  602.3079  288.2030  457.2445  132.1019

> op<-par(mfrow=c(2,1))
> plot(pm, ylab="peptide mass [in Da]",
+     main="Fetuin Peptide tryptic digested.")
> hist(pm, xlab="peptide mass [in Da]")
```



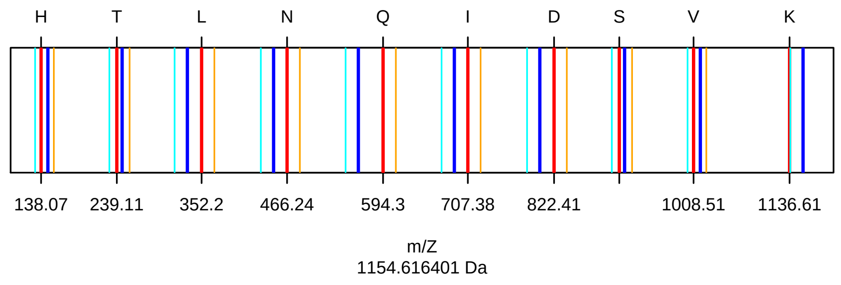


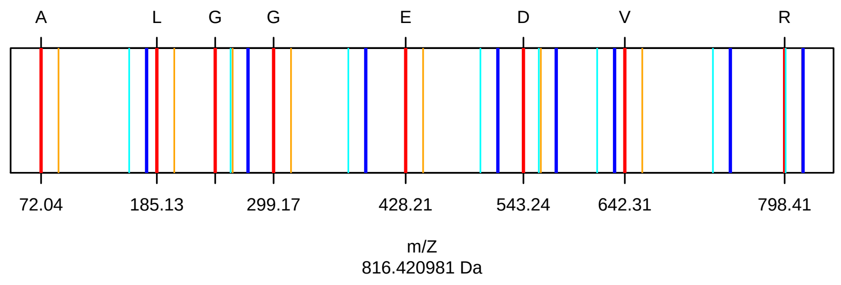
5.2 In-silico Peptide Fragmentation

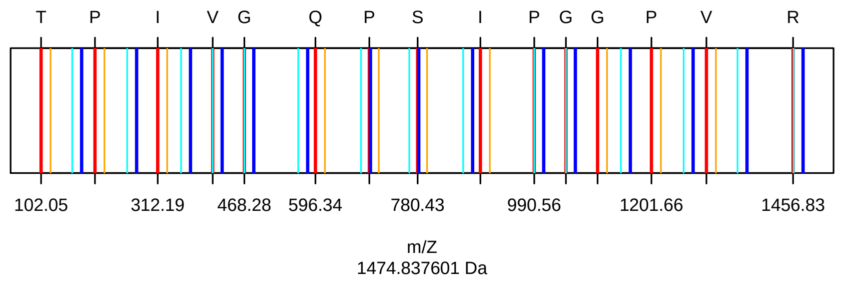
The fragment ions of a peptide can be computed following the rules proposed in [1]. Beside the `b` and `y` ions the `FUN` argument of `fragmentIons` defines which ions are computed. the default ions beeing computed are defined in the function `defaultIons`. The are no limits for defining other forms of fragment ions for ETD (c and z ions) CID (b and y ions).

```
> defaultIons

function (fi)
{
    Hydrogen <- 1.007825
    Oxygen <- 15.994915
    Nitrogen <- 14.003074
    y_0 <- fi$y - Oxygen - Hydrogen - Hydrogen
    c <- fi$b + (Nitrogen + (3 * Hydrogen))
    z <- fi$y - (Nitrogen + (3 * Hydrogen))
    return(cbind(y_0, c, z))
}
<environment: namespace:protViz>
```

```
> peptides<-c('HTLNQIDSVK', 'ALGGEDVR', 'TPIVGQPSIPGGPVR')
> pim<-parentIonMass(peptides)
> fi<-fragmentIons(peptides)
> par(mfrow=c(3,1));
> for (i in 1:length(peptides)){
+     plot(0,0,
+         xlab='m/Z',
+         ylab='',
+         xlim=range(c(fi[i][[1]]$b,fi[i][[1]]$y)),
+         ylim=c(0,1),
+         type='n',
+         axes=FALSE,
+         sub=paste( pim[i], "Da"));
+     box()
+     axis(1,fi[i][[1]]$b,round(fi[i][[1]]$b,2))
+     pepSeq<-strsplit(peptides[i],"")
+     axis(3,fi[i][[1]]$b,pepSeq[[1]])
+
+     abline(v=fi[i][[1]]$b, col='red',lwd=2)
+     abline(v=fi[i][[1]]$c, col='orange')
+     abline(v=fi[i][[1]]$y, col='blue',lwd=2)
+     abline(v=fi[i][[1]]$z, col='cyan')
+ }
```







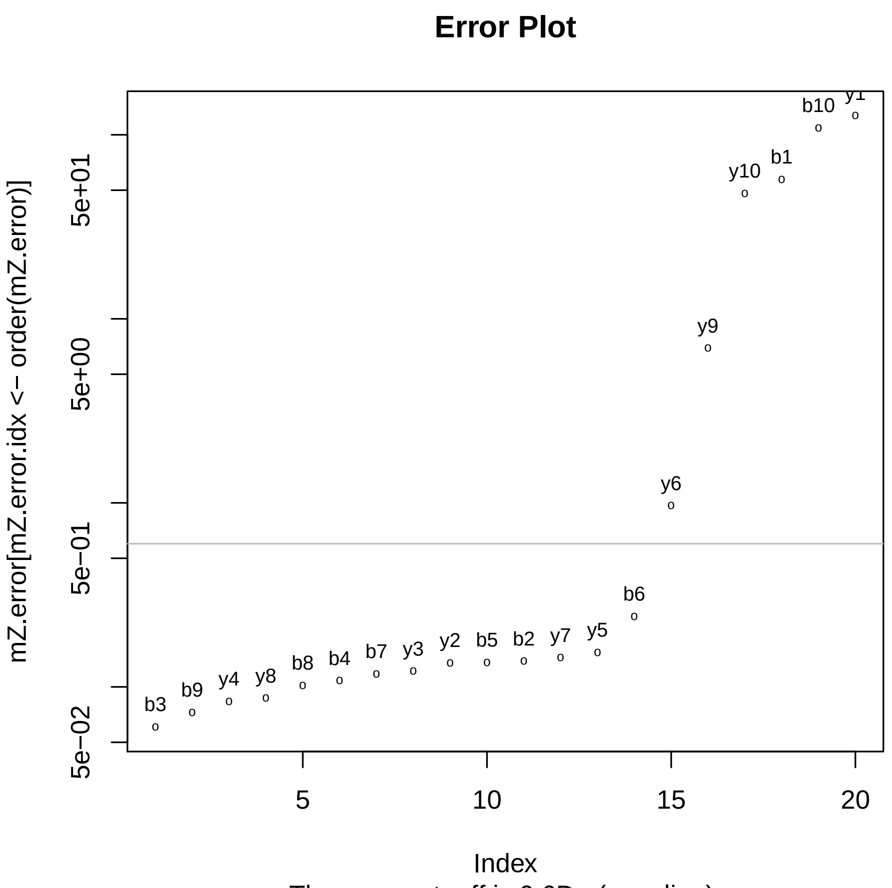
The next lines compute the singly and doubly charged fragment ions of the `HTLNQIDSVK` peptide. Which are usually the ones that can be used to make an identification.

```
> Hydrogen<-1.007825
> (fi.HTLNQIDSVK.1<-fragmentIons('HTLNQIDSVK'))[[1]]

           b         y       y_0         c         z
1   138.0662  147.1128  129.1022  155.0927  130.0863
2   239.1139  246.1812  228.1706  256.1404  229.1547
3   352.1979  333.2132  315.2027  369.2245  316.1867
4   466.2409  448.2402  430.2296  483.2674  431.2136
5   594.2994  561.3242  543.3137  611.3260  544.2977
6   707.3835  689.3828  671.3723  724.4100  672.3563
7   822.4104  803.4258  785.4152  839.4370  786.3992
8   909.4425  916.5098  898.4992  926.4690  899.4833
9  1008.5109 1017.5575  999.5469 1025.5374 1000.5309
10 1136.6058 1154.6164 1136.6058 1153.6324 1137.5899

> (fi.HTLNQIDSVK.2<-(fi.HTLNQIDSVK.1[[1]] + Hydrogen) / 2)

           b         y       y_0         c         z
1   69.53701  74.06031  65.05503  78.05028  65.54704
2  120.06085 123.59452 114.58924 128.57412 115.08124
3  176.60288 167.11053 158.10525 185.11615 158.59726
4  233.62434 224.62400 215.61872 242.13761 216.11073
5  297.65363 281.16603 272.16075 306.16691 272.65276
6  354.19566 345.19532 336.19004 362.70894 336.68205
7  411.70913 402.21679 393.21151 420.22241 393.70351
8  455.22515 458.75882 449.75354 463.73842 450.24554
9  504.75935 509.28266 500.27738 513.27262 500.76938
10 568.80683 577.81211 568.80683 577.32010 569.29884
```

5.3 Peptide Sequence – Fragment Ion Matching

Given a peptide sequence and a tandem mass spectrum. For the assignment of a canditate peptide an in-silico fragment ion spectra `fi` is computed. The function `findNN` determines for each fragment ion the closesed peak in the MS2. If the difference between the in-silico mass and the measured mass is inside the 'accuracy' mass window of the mass spec device the in-silico fragment ion is considered as potential hit.

```
>     peptideSequence<-'HTLNQIDSVK'
>     spec<-list(scans=1138,
+         title="178: (rt=22.3807) [20080816_23_fetuin_160.RAW]",
+         rtinseconds=1342.8402,
+         charge=2,
+         mZ=c(195.139940, 221.211970, 239.251780, 290.221750,
+     316.300770, 333.300050, 352.258420, 448.384360, 466.348830,
+     496.207570, 509.565910, 538.458310, 547.253380, 556.173940,
+      560.358050, 569.122080, 594.435500, 689.536940, 707.624790,
+      803.509240, 804.528220, 822.528020, 891.631250, 909.544400,
+      916.631600, 973.702160, 990.594520, 999.430580, 1008.583600,
+      1017.692500, 1027.605900),
+          intensity=c(931.8, 322.5, 5045, 733.9, 588.8, 9186, 604.6,
+      1593, 531.8, 520.4, 976.4, 410.5, 2756, 2279, 5819, 2.679e+05,
+      1267, 1542, 979.2, 9577, 3283, 9441, 1520, 1310, 1.8e+04,
+      587.5, 2685, 671.7, 3734, 8266, 3309))
>      fi<-fragmentIons(peptideSequence)
>      n<-nchar(peptideSequence)
>      by.mZ<-c(fi[[1]]$b, fi[[1]]$y)
>      by.label<-c(paste("b",1:n,sep=''), paste("y",n:1,sep=''))
>      # should be a R-core function as findInterval!
>      idx<-findNN(by.mZ, spec$mZ)
>      mZ.error<-abs(spec$mZ[idx]-by.mZ)
>      plot(mZ.error[mZ.error.idx<-order(mZ.error)],
+          main="Error Plot",
+          pch='o',
+          cex=0.5,
+          sub='The error cut-off is 0.6Da (grey line).',
+          log='y')
>      abline(h=0.6,col='grey')
>      text(1:length(by.label),
+          mZ.error[mZ.error.idx],
+          by.label[mZ.error.idx],
+          cex=0.75,pos=3)
```



The error cut−off is 0.6Da (grey line).

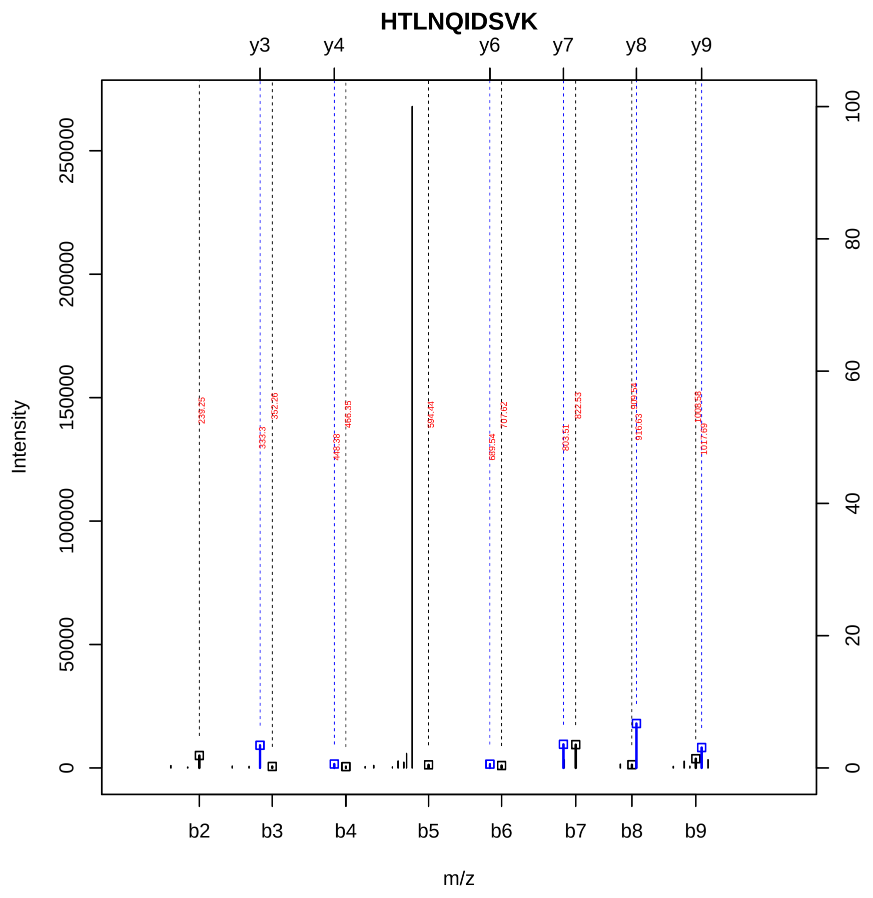
The graphic above is showing the mass error of the assingment between the MS2 `spec` and the singly charged fragment ions of `HTLNQIDSVK`. The function `psm` is doing the peptide sequence assignment. Of course, the more theoretical ions match (up to a small error tolerance, given by the system) the actually measured ion series, the more likely it is, that the measured spectrum indeed is from the inferred peptide (and therefore the protein is identified)

5.4 Labeling Peaklists

The labeling of the spectra can be done with the `peakplot` function.

```
> peakplot('HTLNQIDSVK', spec)

$mZ.Da.error
 [1]   57.073754    0.137914    0.060494    0.107974    0.136064    0.241294
 [7]    0.117584    0.101934    0.072724 -108.999936   48.027139   -6.929431
[13]    0.086809    0.144179   -0.966191    0.154119    0.083489    0.121789
[19]    0.135009 -127.010501

$mZ.ppm.error
 [1] 413379.66705     576.77124     171.76137     231.58417     228.94856
 [6]     341.10776     142.97484     112.08406      72.11028  -95899.50407
[11] 326464.71737  -28147.68427     260.52086     321.65568   -1721.27075
```

```
[16]     223.56084     103.91626     132.88347     132.67948 -110002.33575

$idx
 [1]  1  3  7  9 17 19 22 24 29 31  1  3  6  8 15 18 20 25 30 31

$label
 [1] "b1"  "b2"  "b3"  "b4"  "b5"  "b6"  "b7"  "b8"  "b9"  "b10" "y1"  "y2"
[13] "y3"  "y4"  "y5"  "y6"  "y7"  "y8"  "y9"  "y10"

$score
[1] -1

$sequence
[1] "HTLNQIDSVK"

$fragmentIons
           b         y       y_0         c         z
1   138.0662  147.1128  129.1022  155.0927  130.0863
2   239.1139  246.1812  228.1706  256.1404  229.1547
3   352.1979  333.2132  315.2027  369.2245  316.1867
4   466.2409  448.2402  430.2296  483.2674  431.2136
5   594.2994  561.3242  543.3137  611.3260  544.2977
6   707.3835  689.3828  671.3723  724.4100  672.3563
7   822.4104  803.4258  785.4152  839.4370  786.3992
8   909.4425  916.5098  898.4992  926.4690  899.4833
9  1008.5109 1017.5575  999.5469 1025.5374 1000.5309
10 1136.6058 1154.6164 1136.6058 1153.6324 1137.5899
```



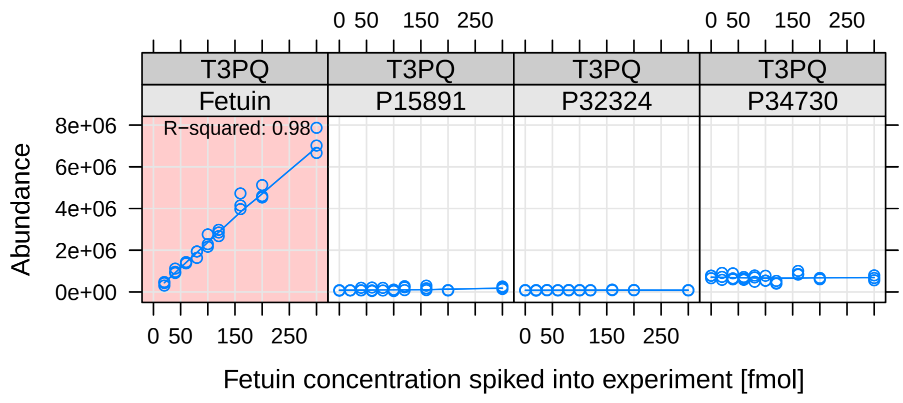
6 Quantification

For an overview on Quantitative Proteomics read [4, 5]. The authors are aware that meaningful statistics usually require much higher number of biological replicates. In almost all cases there are not more than three to six repitions. For the moment there are limited options due to the availabilty of machine time and the limits of the technologies.

6.1 Relative and absolute label-free methods on protein level

The data set `fetuinLFQ` contains a subset of our results descriped in [2]. The example below shows a visualization using trellis plots. It graphs the abundance of four protein in dependency from the fetuin concentration spiked into the sample.

```
> library(lattice)
> data(fetuinLFQ)
> cv<-1-1:7/10
> t<-trellis.par.get("strip.background")
> t$col<-(rgb(cv,cv,cv))
> trellis.par.set("strip.background",t)
> print(xyplot(abundance~conc|prot*method,
+     groups=prot,
+     xlab="Fetuin concentration spiked into experiment [fmol]",
+     ylab="Abundance",
+     aspect=1,
+     data=fetuinLFQ$t3pq[fetuinLFQ$t3pq$prot
+         %in% c('Fetuin', 'P15891', 'P32324', 'P34730'),],
+     panel = function(x, y, subscripts, groups) {
+         if (groups[subscripts][1] == "Fetuin")  {
+             panel.fill(col="#ffcccc")
+         }
+         panel.grid(h=-1,v=-1)
+         panel.xyplot(x, y)
+         panel.loess(x,y, span=1)
+         if (groups[subscripts][1] == "Fetuin")  {
+             panel.text(min(fetuinLFQ$t3pq$conc),
+                 max(fetuinLFQ$t3pq$abundance),
+                 paste("R-squared:",
+                 round(summary(lm(x~y))$r.squared,2)),
+                 cex=0.75,
+                 pos=4)
+         }
+     }
+ ))
```

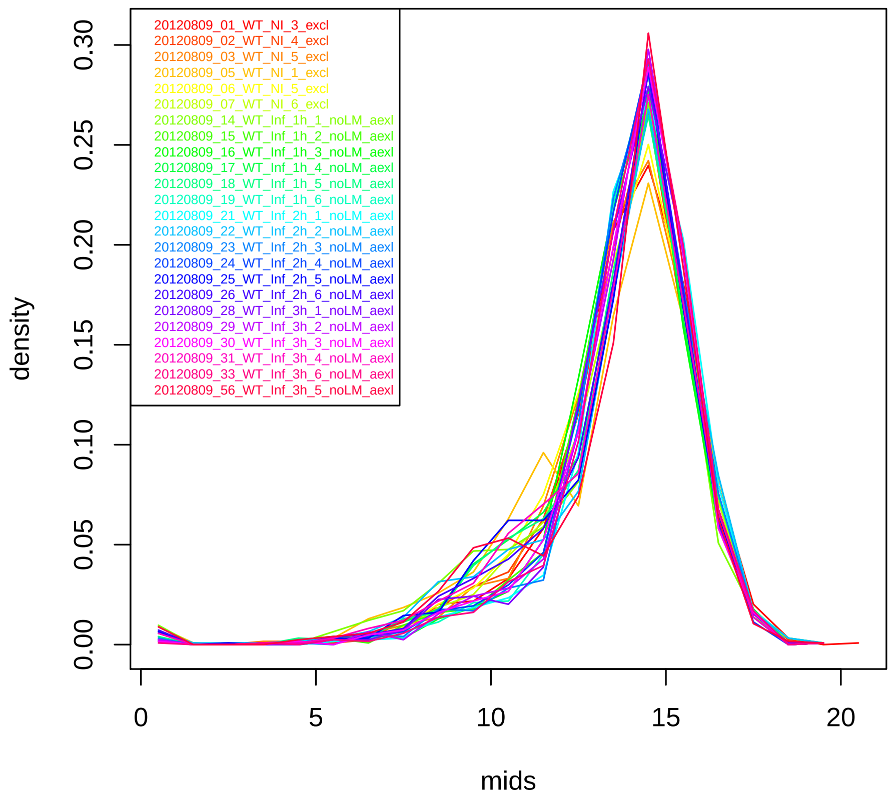


The plot shows the estimated concentration of the four proteins using the top three most intense peptides. The Fetuin peptides are spiked in with increasing concentration while the three other yeast proteins are kept stable in the background.

6.2 pgLFQ – LC MS based relative label-free

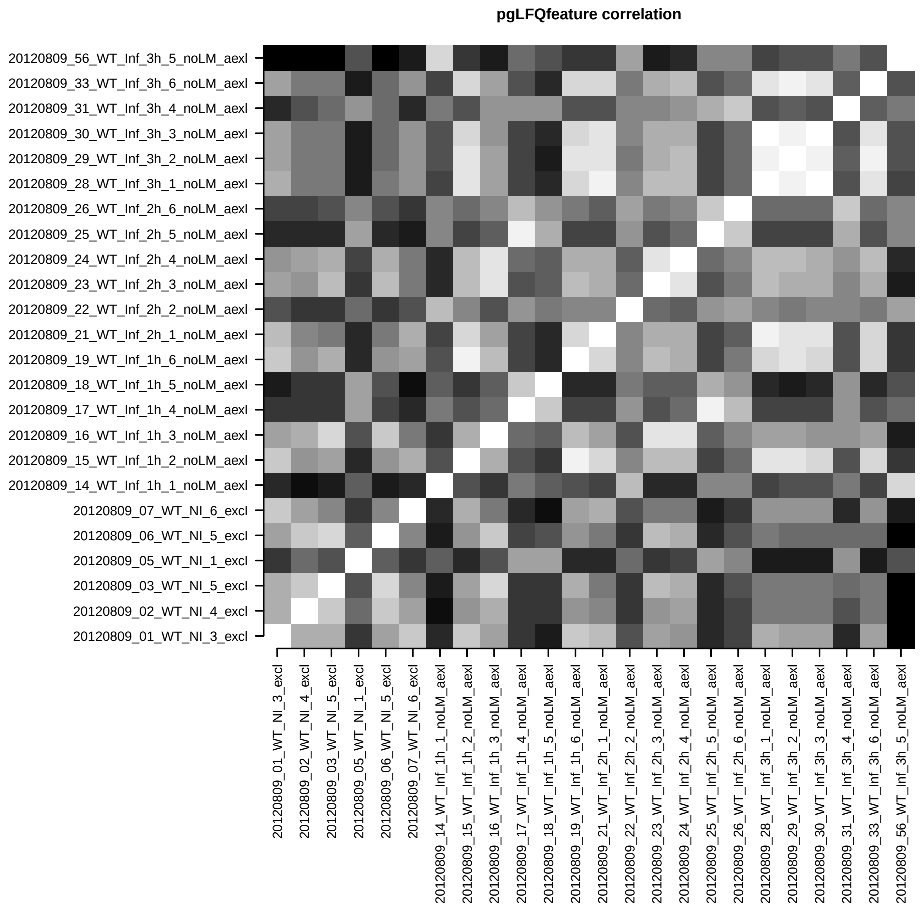
LCMS based label-free quantification is a very popular method to extract relative quantitative information from mass spectrometry experiments. At the FGCZ we use the software ProgenesisLCMS for this workflow `http://www.nonlinear.com/products/progenesis/lc-ms/overview/`. Progenesis is a graphical software which does the aligning and extracts signal intensities from LCMS maps.

```
> data(pgLFQfeature)
> data(pgLFQprot)
> featureDensityPlot<-function(data, n=ncol(data), nbins=30){
+     my.col<-rainbow(n);
+     mids<-numeric()
+     density<-numeric()
+     for (i in 1:n) {
+         h<-hist(data[,i],nbins, plot=F)
+         mids<-c(mids, h$mids)
+         density<-c(density, h$density)
+     }
+     plot(mids,density, type='n')
+     for (i in 1:n) {
+         h<-hist(data[,i],nbins, plot=F)
+         lines(h$mids,h$density, col=my.col[i])
+     }
+     legend("topleft", names(data), cex=0.5,
+         text.col=my.col
+     )
+ }
> par(mfrow=c(1,1));
> featureDensityPlot(asinh(pgLFQfeature$"Normalized abundance"),
+     nbins=25)
```



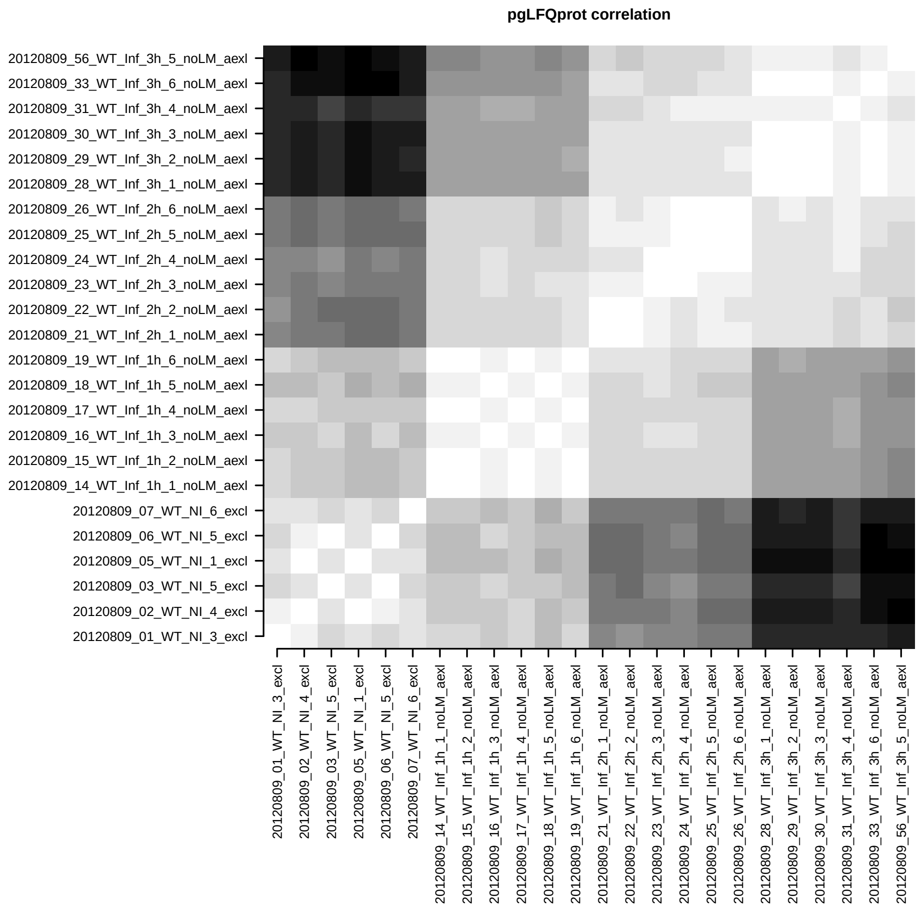
The `featureDensityPlot` shows the normalized signal intensity distribution (asinh transformed) over the 24 LCMS runs aligned in this experiment.

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



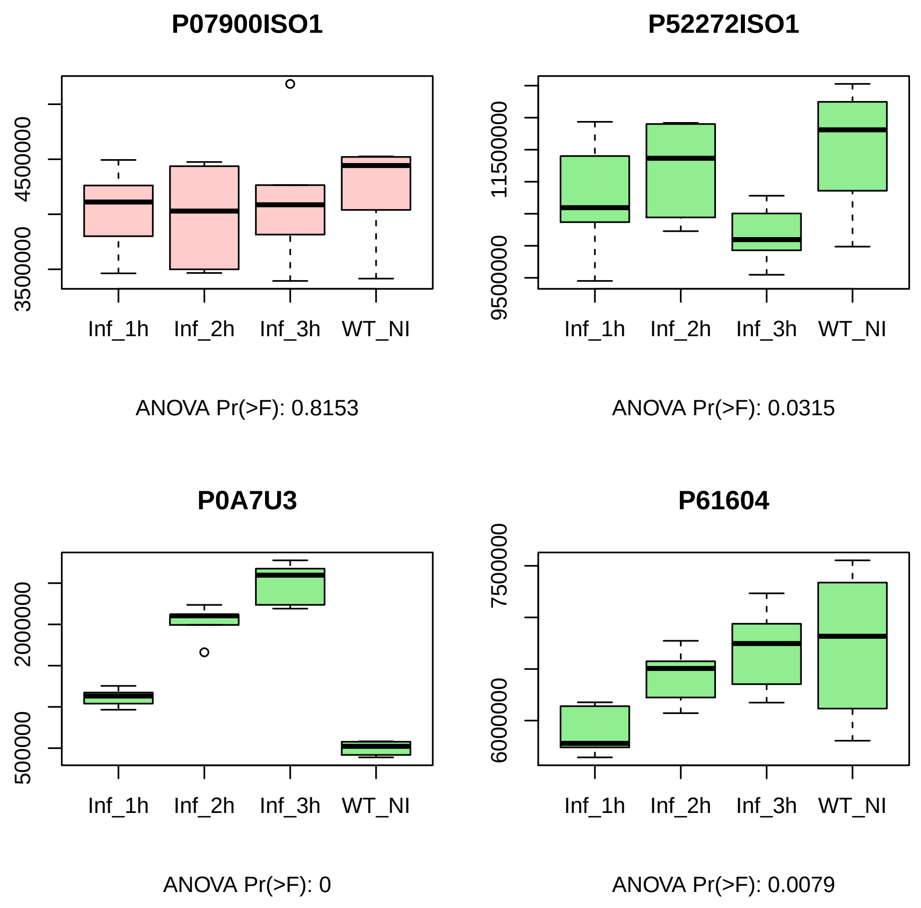
This image plot shows the correlation between runs on feature level (values are asinh transformed). White is perfect correlation while black indicates a poor correlation.

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



This figure shows the correlation between runs on protein level (values are asinh transformed). White is perfect correlation while black indicates a poor correlation. Stricking is the fact that the six biological replicates for each condition cluster very well.

```
> par(mfrow=c(2,2),mar=c(6,3,4,1))
> ANOVA<-pgLFQaov(pgLFQprot$"Normalized abundance",
+      groups=as.factor(pgLFQprot$grouping),
+      names=pgLFQprot$output$Accession,
+      idx=c(15,16,196,107),
+      plot=TRUE)
```



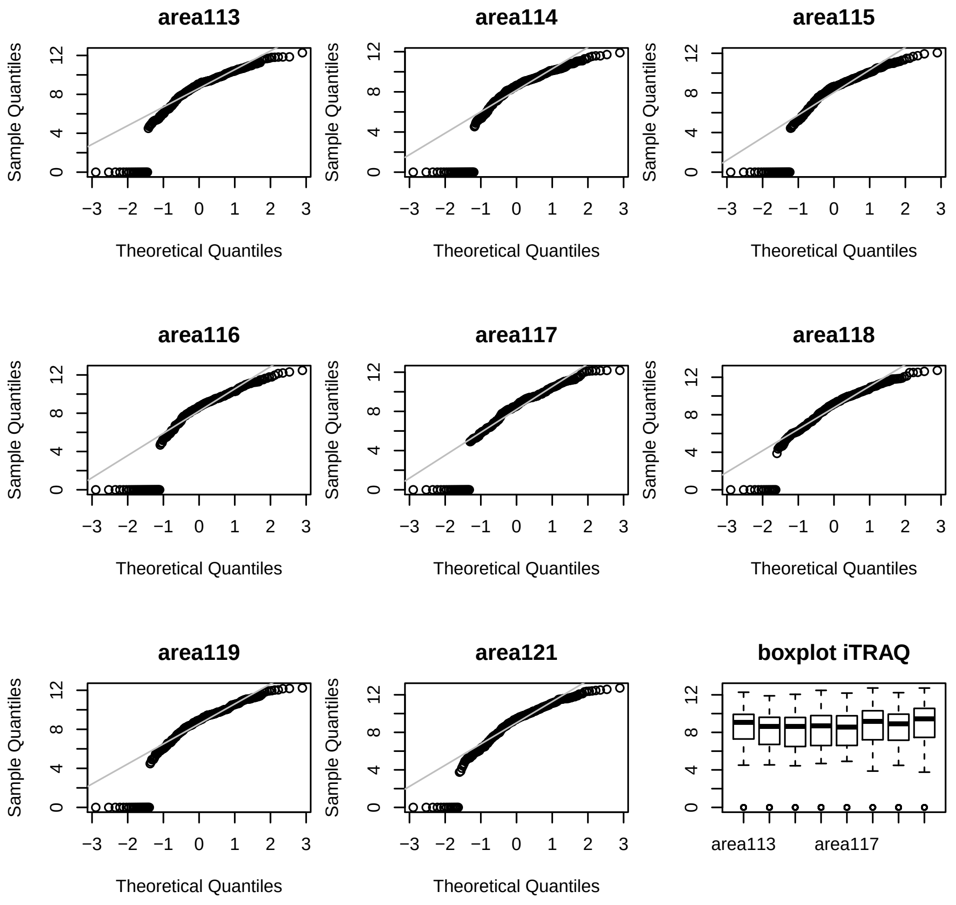
This figure shows the result for four proteins which either differ significantly in expression accross conditions (green boxplots) using an analysis of variance test, or non differing protein expression (red boxplot).

6.3 iTRAQ – Two Group Analysis

The data for the next section is an iTRAQ-8-plex experiment where two conditions are compared (each condition has 4 biological replicates)

6.3.1 Sanity Check

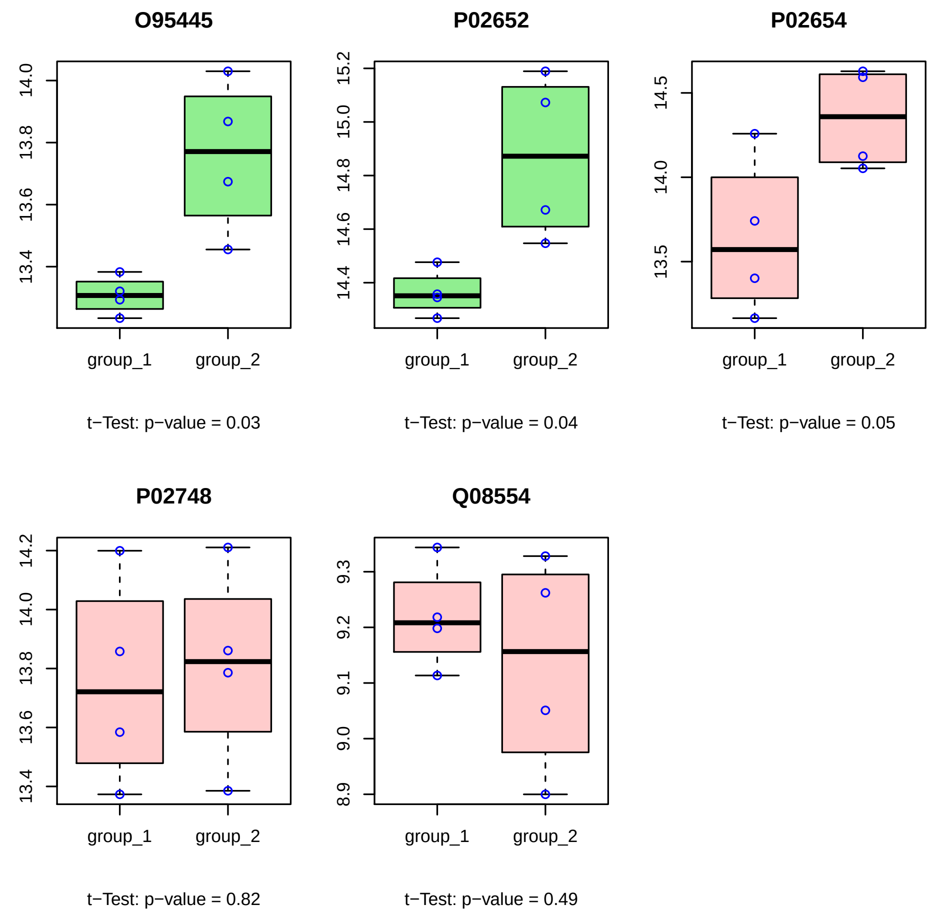
```
> data(iTRAQ)
> x<-rnorm(100)
> par(mfrow=c(3,3),mar=c(6,4,3,0.5));
> for (i in 3:10){
+     qqnorm(asinh(iTRAQ[,i]),
+         main=names(iTRAQ)[i])
+     qqline(asinh(iTRAQ[,i]), col='grey')
+ }
> b<-boxplot(asinh(iTRAQ[,c(3:10)]), main='boxplot iTRAQ')
```



A first check to see if all reporter ion channels are having the same distributions. Shown in the figure are Q-Q plots of the individual reporter channels against a normal distribution. The last is a boxplot for all individual channels.

6.3.2 On Protein Level

```
> data(iTRAQ)
> group1Protein<-numeric()
> group2Protein<-numeric()
> for (i in c(3,4,5,6))
+     group1Protein<-cbind(group1Protein,
+         asinh(tapply(iTRAQ[,i], paste(iTRAQ$prot), sum, na.rm=TRUE)))
> for (i in 7:10)
+     group2Protein<-cbind(group2Protein,
+         asinh(tapply(iTRAQ[,i], paste(iTRAQ$prot), sum, na.rm=TRUE)))
> par(mfrow=c(2,3),mar=c(6,3,4,1))
> for (i in 1:nrow(group1Protein)){
+     boxplot.color="#ffcccc"
+     tt.p_value<-t.test(as.numeric(group1Protein[i,]),
+         as.numeric(group2Protein[i,]))$p.value
+
+     if (tt.p_value < 0.05)
+           boxplot.color='lightgreen'
+
+       b<-boxplot(as.numeric(group1Protein[i,]),
+           as.numeric(group2Protein[i,]),
+           main=row.names(group1Protein)[i],
+           sub=paste("t-Test: p-value =", round(tt.p_value,2)),
+           col=boxplot.color,
+           axes=F)
+       axis(1, 1:2, c('group_1','group_2')); axis(2); box()
+
+       points(rep(1,b$n[1]), as.numeric(group1Protein[i,]), col='blue')
+       points(rep(2,b$n[2]), as.numeric(group2Protein[i,]), col='blue')
+ }
```



This figure shows five proteins which are tested if they differ accross conditions using the four biological replicates with a t-test.

6.3.3 On Peptide Level

The same can be done on peptide level using the `protViz` function `iTRAQ2GroupAnalysis`.

```
> data(iTRAQ)
> q<-iTRAQ2GroupAnalysis(data=iTRAQ,
+     group1=c(3,4,5,6),
+     group2=7:10,
+     INDEX=paste(iTRAQ$prot,iTRAQ$peptide),
+     plot=F)
> q[1:10,]

                          name p_value Group1.area113 Group1.area114
1               O95445 AFLLTPR   0.056        1705.43        1459.10
2               O95445 DGLCVPR   0.161        2730.41        1852.90
3             O95445 MKDGLCVPR   0.039       28726.38       15409.81
4            O95445 NQEACELSNN   0.277        4221.31        4444.28
5             O95445 SLTSCLDSK   0.036       20209.66       14979.02
6    P02652 AGTELVNFLSYFVELGTQPA   0.640        4504.97        4871.88
7   P02652 AGTELVNFLSYFVELGTQPAT   0.941       67308.30       46518.21
8  P02652 AGTELVNFLSYFVELGTQPATQ   0.338        4661.54        3971.82
9    P02652 EPCVESLVSQYFQTVTDYGK   0.115        4544.56        4356.51
10             P02652 EQLTPLIK   0.053       24596.42       22015.94
   Group1.area115 Group1.area116 Group2.area117 Group2.area118 Group2.area119
1          770.65        3636.40        3063.48        4046.73        2924.49
2         1467.65        2266.88        2269.57        3572.32        2064.82
3        19050.13       58185.02       51416.05       70721.05       38976.42
4         2559.23        6859.71        5545.12       11925.66        6371.50
5        12164.94       37572.56       30687.57       39176.99       34417.66
6         2760.53        9213.41        6728.62       14761.96        7796.29
7        33027.14      111629.30       94531.76      168775.00       83526.72
8         2564.39        8269.73        6045.30       13724.92        7426.84
9         2950.48        6357.90        6819.99       10265.84        7012.92
10       18424.56       49811.91       33197.47       67213.62       40030.86
   Group2.area121
1         5767.87
2         2208.92
3        60359.72
4        15656.92
5        54439.22
6        18681.60
7       168032.50
8        17214.87
9        14279.22
10       87343.38
```

7 Pressure Profiles QC

A common problem with mass spec setup is the pure reliability of the high pressure pump. The following graphics provide visualizations for quality control.

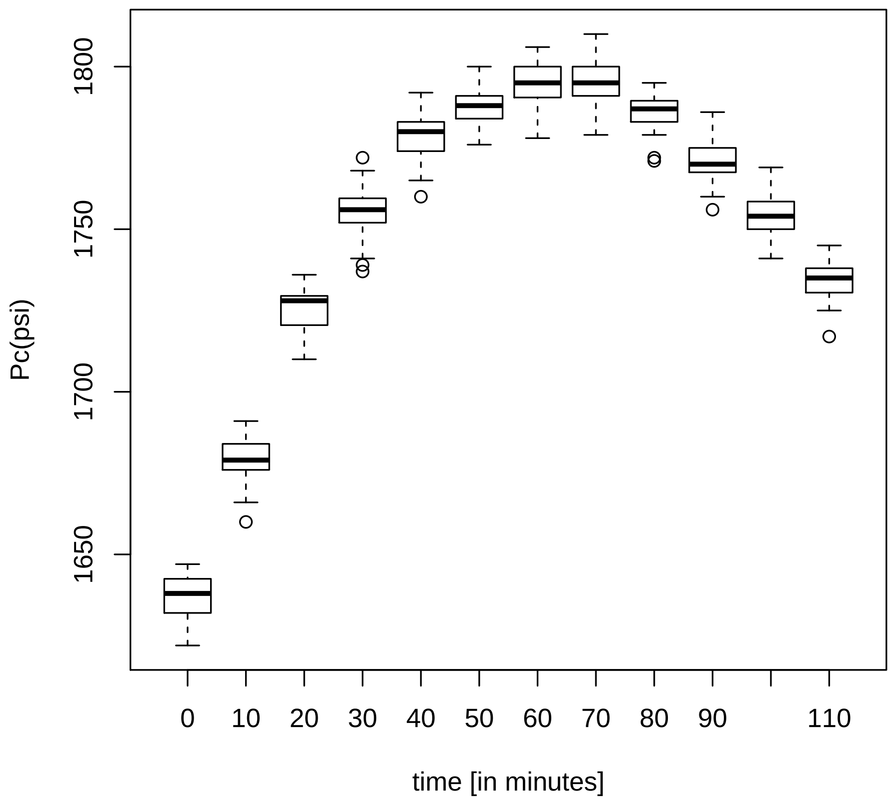
On overview of the pressure profile data can be seen by using the `pressureProfilePlot` function.

```
> data(pressureProfile)
> pressureProfilePlot(pressureProfile)
```

The lines plots the pressure profiles data on a scatter plot 'Pc' versus 'time' grouped by time range (no figure because of too many data items).

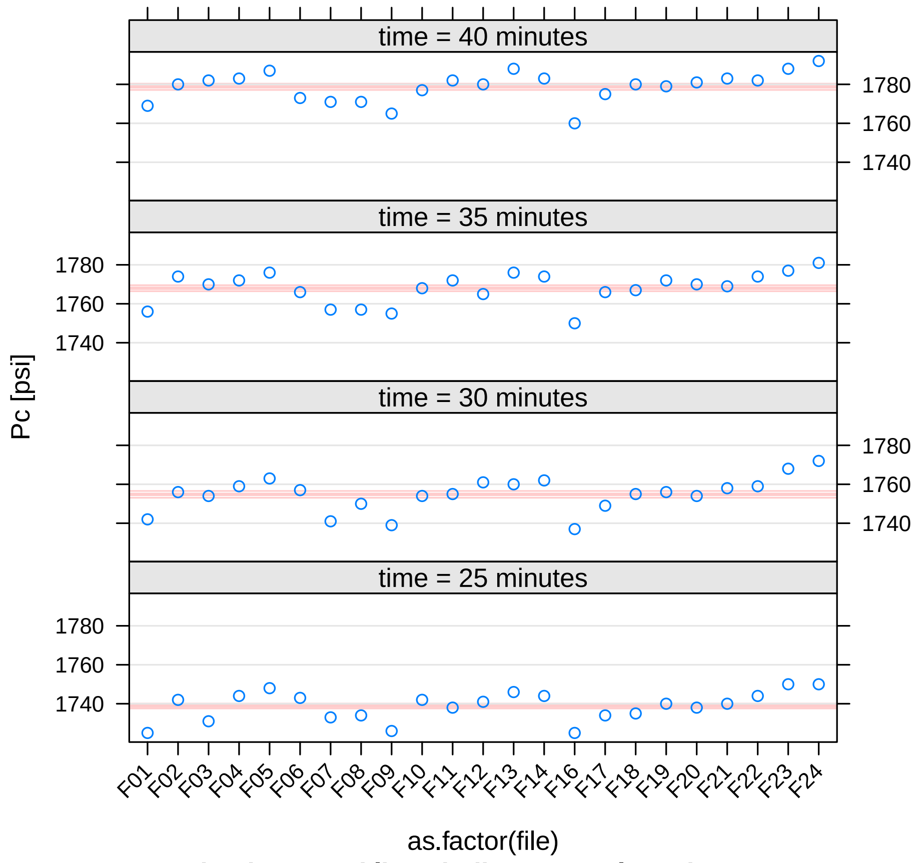
The following boxplots display how the Pc values are distributed over several points in time. For determine the plotting data the `pressureProfileSummary` has to be used.

```
> data(pressureProfile)
> par(mfrow=c(1,1))
> pp<-pressureProfileSummary(pressureProfile, time=seq(0,110,by=10))
> boxplot(Pc~time, data=pp, xlab='time [in minutes]', ylab='Pc(psi)')
```



The Trellis `xyplot` shows the Pc development over each instrument run to a specified relative run time (25,30,...).

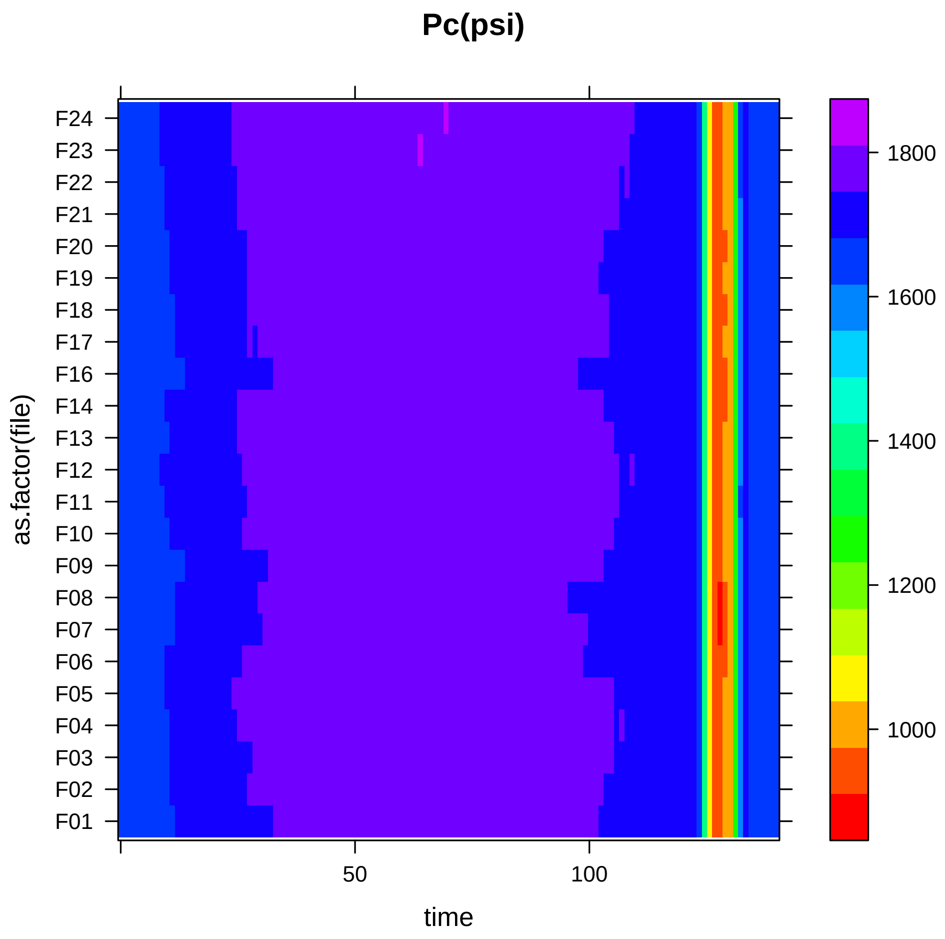
```
> pp<-pressureProfileSummary(pressureProfile, time=seq(25,40,by=5))
> print(xyplot(Pc ~ as.factor(file) | paste("time =",
+     as.character(time), "minutes"),
+     panel = function(x, y){
+         m<-sum(y)/length(y)
+         m5<-(max(y)-min(y))*0.05
+         panel.abline(h=c(m-m5,m,m+m5),
+             col=rep("#ffcccc",3),lwd=c(1,2,1))
+         panel.grid(h=-1, v=0)
+         panel.xyplot(x, y)
+     },
+     ylab='Pc [psi]',
+     layout=c(1,4),
+     sub='The three read lines indicate avg plus min 5%.',
+     scales = list(x = list(rot = 45)),
+     data=pp))
```



The three read lines indicate avg plus min 5%.

While each panel in the `xyplot` above shows the data to a given point in time, we try to use the levelplot to get an overview of the whole pressure profile data.

```
> pp<-pressureProfileSummary(pressureProfile, time=seq(0,140,length=128))
> print(levelplot(Pc ~ time * as.factor(file),
+     main='Pc(psi)',
+     data=pp,
+     col.regions=rainbow(100)[1:80]))
```



References

[1] Roepstorff P, Fohlman J., Proposal for a common nomenclature for sequence ions in mass spectra of peptides. Biomed Mass Spectrom. 1984 Nov;11(11):601 (pubmed ID:6525415).

[2] Grossmann J, Roschitzki B, Panse C, Fortes C, Barkow-Oesterreicher S, Rutishauser D, Schlapbach R., Implementation and evaluation of relative and absolute quantification in shotgun proteomics with label-free methods. J Proteomics. 2010 Aug 5;73(9):1740-6. Epub 2010 May 31 (pubmed ID:20576481).

[3] Ortea I, Roschitzki B, Ovalles JG, Longo JL, de la Torre I, González I, Gómez-Reino JJ, González A., Discovery of serum proteomic biomarkers for prediction of response to infliximab (a monoclonal anti-TNF antibody) treatment in rheumatoid arthritis: An exploratory analysis., J Proteomics. 2012 Sep 20. pii: S1874-3919(12)00655-0. doi: 10.1016/j.jprot.2012.09.011

[4] Bantscheff M, Lemeer S, Savitski MM, Kuster B., Quantitative mass spectrometry in proteomics: critical review update from 2007 to the presen., Anal Bioanal Chem. 2012 Sep;404(4):939-65. doi: 10.1007/s00216-012-6203-4

[5] Cappadona S, Baker PR, Cutillas PR, Heck AJ, van Breukelen B, Current challenges in software solutions for mass spectrometry-based quantitative proteomics., Amino Acids. 2012 Sep;43(3):1087-108. Epub 2012 Jul 22. (pubmed ID: 22821268)