

protViz: Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics

Christian Panse

Jonas Grossmann

June 20, 2014

Vignette for package version $\geq v.0.1.48$

Recent changes, updates, and new features:

- 0.1.92 added unit tests.
- 0.1.91 bugfix rt normalization in `genSwathLib`.
- 0.1.89 replace `fragmentIons` by `fragmentIon`.
- 0.1.77 bugfix `xlim` in `peakplot`

Contents

1	Preliminary Note	3
2	Related Work	3
3	Get Data In – Preprocessing	3
3.1	Identification - In-silico from Proteins to Peptides	4
4	Peptide Identification	5
4.1	Computing the Parent Ion Mass	5
4.2	In-silico Peptide Fragmentation	6
4.3	Peptide Sequence – Fragment Ion Matching	8
4.4	Modifications	10
4.5	Labeling Peaklists	11
5	Quantification	15
5.1	Label-free methods on protein level	15
5.2	pgLFQ – LCMS based label-free quantification	17
5.3	iTRAQ – Two Group Analysis	21
5.3.1	Sanity Check	21

5.3.2	On Protein Level	22
5.3.3	On Peptide Level	23
6	Pressure Profiles QC	24

1 Preliminary Note

protViz is an R package to do quality checks, visualizations 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. Nevertheless, if one is patient, it also handles large data sets.

2 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 functions for mass spec data including quant aspect with iTRAQ data)
<http://www.bioconductor.org/packages/release/bioc/html/MSnbase.html>
- plgem – spectral counting quantification, applicable to MudPIT experiments
<http://www.bioconductor.org/packages/release/bioc/html/plgem.html>
- synapter – MSe (Hi3 = Top3 Quantification) for Waters Q-tof data aquired in MSe mode
<http://www.bioconductor.org/packages/release/bioc/html/synapter.html>
- mzR
<http://www.bioconductor.org/packages/release/bioc/html/mzR.html>
- isobar iTRAQ/TMT quantification package
<http://www.bioconductor.org/packages/release/bioc/html/isobar.html>
- readMzXmlData
<http://cran.r-project.org/web/packages/readMzXmlData/>
- msQC
<http://bioconductor.org/packages/3.0/bioc/html/msQC.html>

3 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. In this package, we intentionally left this part away because it is very infrastructure dependent. Moreover we use also commercial tools to analyze data and export the data into R accessible formats. We provide different kind of importers if these formats are available, but with very little effort, one can bring other exports in a similar format which will make it easy to use our package for a variety of tools.

3.1 Identification - 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.

`fc` and `tryptic-digest` are commandline programs which are included in the package. `fc` 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
QDGQFSVLFTKCDSSPDSAEDVRKLCPCPLLAPLNSRVVHAVEVALATFNAESNGSYL
QLVEISR A QFVPLPVSVSVEFAVAATDCIAKEVVDPTKCNLLAEKQYGFKGSGVIQKALG
GEDVRVTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAV
PLPLHRAHYDLRHTFSGVASVESSSGEAFHVGKTPIVGQPSIPGGPVRLCPGRIRYFKI

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

4 Peptide Identification

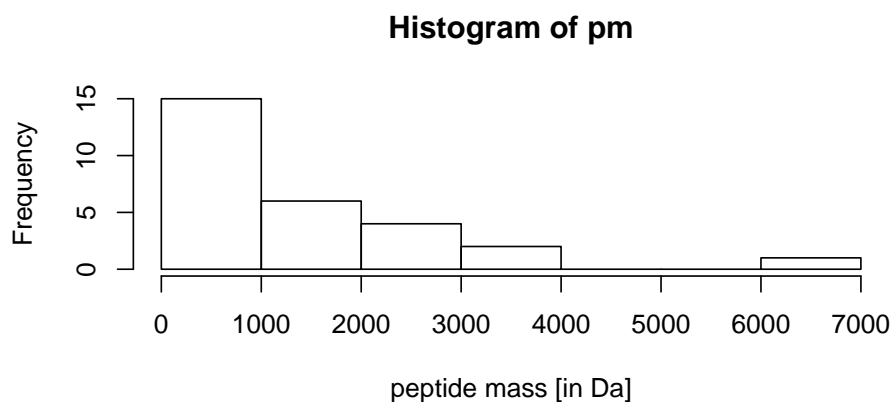
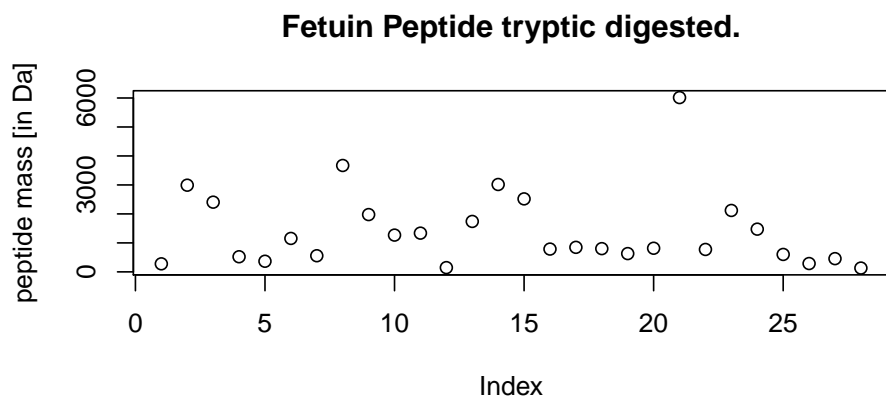
The currency in proteomics are the peptides. In proteomics, proteins are digested to so-called peptides since peptides are much easier to handle biochemically than proteins. Proteins are very different in nature some are very sticky while others are soluble in aqueous solutions while again are only sitting 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.

4.1 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',
+ 'VTCTLFQTQPVIPQPQPDGAEEAPSAPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR',
+ '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]")
```



4.2 In-silico Peptide Fragmentation

The fragment ions of a peptide can be computed following the rules proposed in [4]. Beside the b and y ions the FUN argument of `fragmentIon` defines which ions are computed. the default ions being computed are defined in the function `defaultIon`. There are no limits for defining other forms of fragment ions for ETD (c and z ions) CID (b and y ions).

```
> defaultIon
```

```
function (b, y)
{
  Hydrogen <- 1.007825
  Oxygen <- 15.994915
  Nitrogen <- 14.003074
  c <- b + (Nitrogen + (3 * Hydrogen))
  z <- y - (Nitrogen + (3 * Hydrogen))
  return(cbind(b, y, c, z))
}
```

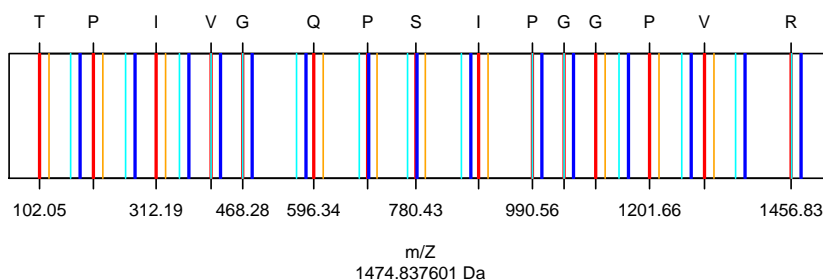
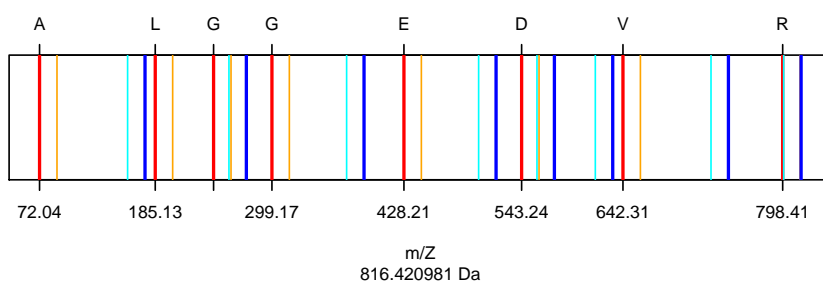
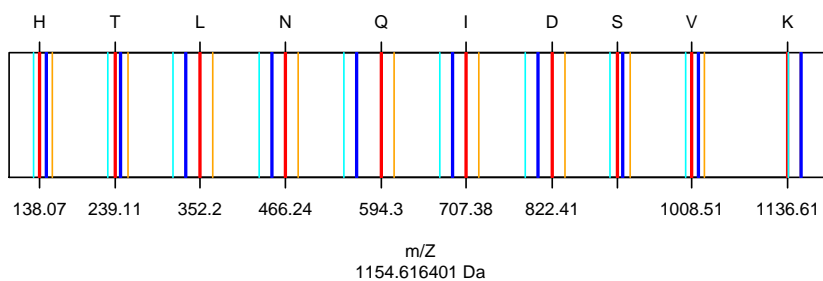
```
<environment: namespace:protViz>
```

```
> peptides<-c('HTLNQIDSVK', 'ALGGEDVR', 'TPIVGQPSIPGGPVR')
> pim<-parentIonMass(peptides)
```

```

> fi<-fragmentIon(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<-fragmentIon('HTLNQIDSVK'))[[1]]
```

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

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

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

4.3 Peptide Sequence – Fragment Ion Matching

Given a peptide sequence and a tandem mass spectrum. For the assignment of a candidate peptide an in-silico fragment ion spectra *fi* is computed. The function *findNN* determines for each fragment ion the closed 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,
```

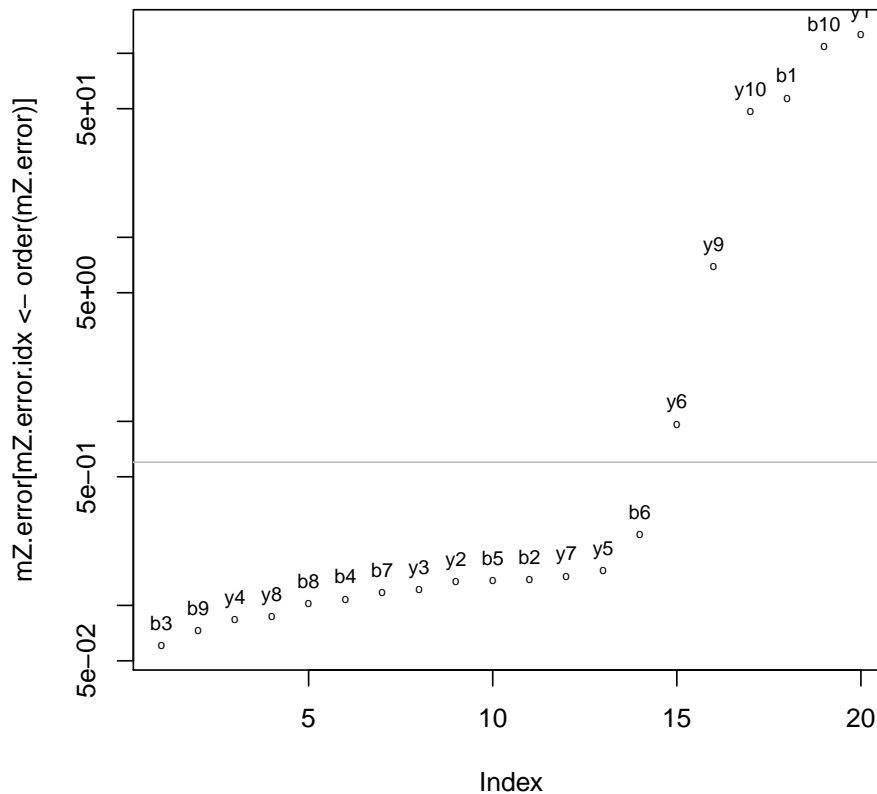


```

+ 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<-fragmentIon(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)

```

Error Plot



The graphic above is showing the mass error of the assignment between the MS2 spec and the singly charged fragment ions of HTLNQIDSVK. The function `psm` is doing the peptide sequence matching. 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)

4.4 Modifications

```
> library(protViz)
> ptm.0<-cbind(AA="-",
+   mono=0.0, avg=0.0, desc="unmodified", unimodAccID=NA)
> ptm.616<-cbind(AA='S',
+   mono=-27.010899, avg=NA, desc="Substitution",
+   unimodAccID=616)
> ptm.651<-cbind(AA='N',
+   mono=27.010899, avg=NA, desc="Substitution",
+   unimodAccID=651)
> m<-as.data.frame(rbind(ptm.0, ptm.616, ptm.651))
> genMod(c('TAFDEAIAELDTLNEESYK', 'TAFDEAIAELDTLSEESYK'), m$AA)

[[1]]
[1] "00000000000000000000" "00000000000000200000" "0000000000000000100"
[4] "00000000000000100200"
```

```
[[2]]
[1] "00000000000000000000" "0000000000000100000" "0000000000000000100"
[4] "0000000000000100100"
```

```
> fi<-fragmentIon(c('TAFDEAIAELDTLSEESYK',
+ 'TAFDEAIAELDTLNEESYK', 'TAFDEAIAELDTLSEESYK',
+ 'TAFDEAIAELDTLNEESYK'),
+ modified=c('0000000000000200000',
+ '0000000000000100000', '0000000000000000000',
+ '0000000000000000000'),
+ modification=m$mono)
>
> #bh<-c('TAFDEAIAELDTLNEESYK', 'TAFDEAIAELDTLSEESYK')
> #fi<-fragmentIon(rep('HTLNQIDSVK',2),
> # modified=c('0000000100','0000000000'),
> # modification=m[,2])
```

4.5 Labeling Peaklists

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

```
> data(msms)
> op<-par(mfrow=c(2,1))
> peakplot("TAFDEAIAELDTLNEESYK", msms[[1]])
```

```
$mZ.Da.error
[1] 232.331344 161.294234 14.225824 -0.032616 -0.143306 0.032244
[7] 0.054604 -0.004076 -0.071746 -0.084536 -0.097076 -0.038856
[13] -0.061816 0.004554 -0.122336 -0.139626 -1.071256 NA
[19] NA 187.273499 24.210169 0.048669 0.177779 0.027939
[25] 0.049579 0.052379 0.044579 0.036749 0.043189 -0.035101
[31] -0.061011 0.000729 -0.092081 2.011029 -8.412111 7.195579
[37] NA NA 215.304795 144.267685 -2.800725 -17.059165
[43] 2.034875 2.264105 4.008125 1.292875 -0.003965 -13.612585
[49] -0.060925 -17.065405 3.897535 3.000405 -17.148885 -17.166175
[55] -18.097805 NA NA 204.300048 41.236718 17.075218
[61] -0.843372 -1.091812 0.129908 17.078928 -0.372162 -16.539502
[67] -1.044962 -1.000952 -1.409062 -2.995122 16.934468 19.037578
[73] 8.614438 24.222128 NA NA
```

```
$mZ.ppm.error
[1] 2.276532e+06 9.318407e+05 4.443342e+04 -7.494702e+01 -2.539851e+02
[6] 5.075660e+01 7.296574e+01 -4.974443e+00 -7.564705e+01 -7.963713e+01
[11] -8.250960e+01 -3.041352e+01 -4.445040e+01 3.026484e+00 -7.488007e+01
[16] -7.920687e+01 -5.791093e+02 NA NA 1.272993e+06
[21] 7.805297e+04 1.225277e+02 3.378218e+02 4.263587e+01 6.444386e+01
[26] 5.935833e+01 4.532837e+01 3.345395e+01 3.564687e+01 -2.618263e+01
```

[31]	-4.321937e+01	4.781134e-01	-5.770282e+01	1.165934e+03	-4.572174e+03
[36]	3.621478e+03	NA	NA	1.808046e+06	7.588299e+05
[41]	-8.306147e+03	-3.772366e+04	3.500821e+03	3.470990e+03	5.236793e+03
[46]	1.545734e+03	-4.106862e+00	-1.262129e+04	-5.104441e+01	-1.318183e+04
[51]	2.768725e+03	1.971690e+03	-1.038832e+04	-9.644849e+03	-9.694247e+03
[56]	NA	NA	1.570497e+06	1.406678e+05	4.491332e+04
[61]	-1.656190e+03	-1.710589e+03	1.726789e+02	1.973544e+04	-3.850849e+02
[66]	-1.529356e+04	-8.747728e+02	-7.562373e+02	-1.010347e+03	-1.986529e+03
[71]	1.072648e+04	1.114745e+04	4.725878e+03	1.229618e+04	NA
[76]	NA				

\$idx

[1]	1	1	1	3	14	21	38	49	64	87	91	97	102	106	110	113	115	117	117
[20]	1	1	2	12	25	41	53	70	89	94	99	104	107	108	111	114	116	117	117
[39]	1	1	1	3	16	24	41	52	67	88	93	97	104	107	110	113	115	117	117
[58]	1	1	2	11	22	40	53	68	88	93	98	103	106	108	111	114	116	117	117

\$label

[1]	"b1"	"b2"	"b3"	"b4"	"b5"	"b6"	"b7"	"b8"	"b9"	"b10"	"b11"	"b12"
[13]	"b13"	"b14"	"b15"	"b16"	"b17"	"b18"	"b19"	"y1"	"y2"	"y3"	"y4"	"y5"
[25]	"y6"	"y7"	"y8"	"y9"	"y10"	"y11"	"y12"	"y13"	"y14"	"y15"	"y16"	"y17"
[37]	"y18"	"y19"	"c1"	"c2"	"c3"	"c4"	"c5"	"c6"	"c7"	"c8"	"c9"	"c10"
[49]	"c11"	"c12"	"c13"	"c14"	"c15"	"c16"	"c17"	"c18"	"c19"	"z1"	"z2"	"z3"
[61]	"z4"	"z5"	"z6"	"z7"	"z8"	"z9"	"z10"	"z11"	"z12"	"z13"	"z14"	"z15"
[73]	"z16"	"z17"	"z18"	"z19"								

\$score

[1] -1

\$sequence

[1] "TAFDEAIAELDTLNEESYK"

\$fragmentIon

	b	y	c	z
1	102.0550	147.1128	119.0815	130.0863
2	173.0921	310.1761	190.1186	293.1496
3	320.1605	397.2082	337.1870	380.1816
4	435.1874	526.2508	452.2140	509.2242
5	564.2300	655.2933	581.2566	638.2668
6	635.2671	769.3363	652.2937	752.3097
7	748.3512	882.4203	765.3777	865.3938
8	819.3883	983.4680	836.4148	966.4415
9	948.4309	1098.4950	965.4574	1081.4684
10	1061.5149	1211.5790	1078.5415	1194.5525
11	1176.5419	1340.6216	1193.5684	1323.5951
12	1277.5896	1411.6587	1294.6161	1394.6322
13	1390.6736	1524.7428	1407.7002	1507.7162

```

14 1504.7165 1595.7799 1521.7431 1578.7533
15 1633.7591 1724.8225 1650.7857 1707.7959
16 1762.8017 1839.8494 1779.8283 1822.8229
17 1849.8338 1986.9178 1866.8603 1969.8913
18 2012.8971 2057.9549 2029.9236 2040.9284
19 2140.9920 2159.0026 2158.0186 2141.9761

```

```
> peakplot("TAFDEAIAELDTLSEESYK", msms[[2]])
```

```
$mZ.Da.error
```

```

[1] 245.264254 174.227144 27.158734 14.444434 0.021404 -0.111266
[7] -0.039926 -0.021626 -0.121916 -8.079236 -0.158376 -0.153156
[13] -0.094316 -0.022946 -0.186736 -0.092226 -0.120456 NA
[19] NA 200.206409 37.143079 0.078909 0.062269 0.129769
[25] 0.103729 0.060869 -0.051451 -18.048351 -0.027511 -0.025601
[31] -0.006211 0.020529 -0.048781 -0.024771 -9.166311 6.953579
[37] NA NA 228.237705 157.200595 10.132185 -2.582115
[43] 1.626855 2.722405 9.009025 -1.130895 1.216385 13.347315
[49] -3.671525 0.960295 -17.120865 3.020205 -17.213285 -17.118775
[55] -17.147005 NA NA 217.232958 54.169628 17.105458
[61] -0.833452 -1.260332 -0.899352 -3.098942 -1.173512 -1.021802
[67] -0.939162 -1.007752 -1.377062 -3.022622 16.977768 17.001778
[73] 7.860238 23.980128 NA NA

```

```
$mZ.ppm.error
```

```

[1] 2.403257e+06 1.006558e+06 8.482850e+04 3.319130e+04 3.793488e+01
[6] -1.751484e+02 -5.335196e+01 -2.639286e+01 -1.285450e+02 -7.611043e+03
[11] -1.346114e+02 -1.198789e+02 -6.782037e+01 -1.552813e+01 -1.162198e+02
[16] -5.313198e+01 -6.608212e+01 NA NA 1.360904e+06
[21] 1.197483e+05 1.986591e+02 1.183257e+02 1.980319e+02 1.397352e+02
[26] 7.115774e+01 -5.379332e+01 -1.684426e+04 -2.322450e+01 -1.948903e+01
[31] -4.485617e+00 1.370673e+01 -3.109508e+01 -1.458996e+01 -5.056331e+03
[36] 3.547913e+03 NA NA 1.916651e+06 8.268554e+05
[41] 3.004915e+04 -5.709941e+03 2.798859e+03 4.173588e+03 1.177069e+04
[46] -1.352074e+03 1.259905e+03 1.237534e+04 -3.076091e+03 7.417604e+02
[51] -1.216230e+04 2.020566e+03 -1.060078e+04 -9.766434e+03 -9.319787e+03
[56] NA NA 1.669915e+06 1.847849e+05 4.499286e+04
[61] -1.636709e+03 -1.974616e+03 -1.239974e+03 -3.696333e+03 -1.249174e+03
[66] -9.690310e+02 -8.043928e+02 -7.772361e+02 -1.006903e+03 -2.041339e+03
[71] 1.094110e+04 1.011538e+04 4.376983e+03 1.234257e+04 NA
[76] NA

```

```
$idx
```

```

[1] 1 1 1 3 11 20 39 45 64 90 96 106 116 121 126 129 131 134 134
[20] 1 1 2 7 24 38 49 65 90 97 110 115 122 123 127 130 132 134 134
[39] 1 1 1 3 13 23 40 47 67 91 98 108 116 122 126 129 131 134 134
[58] 1 1 2 6 21 36 47 62 90 95 108 113 121 123 127 130 132 134 134

```

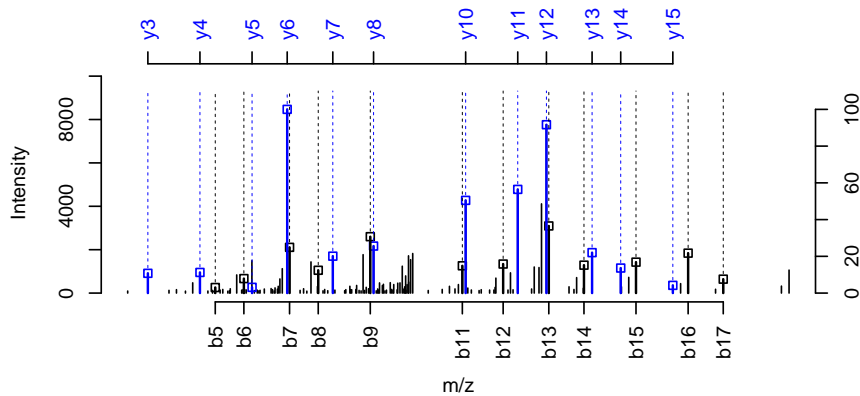
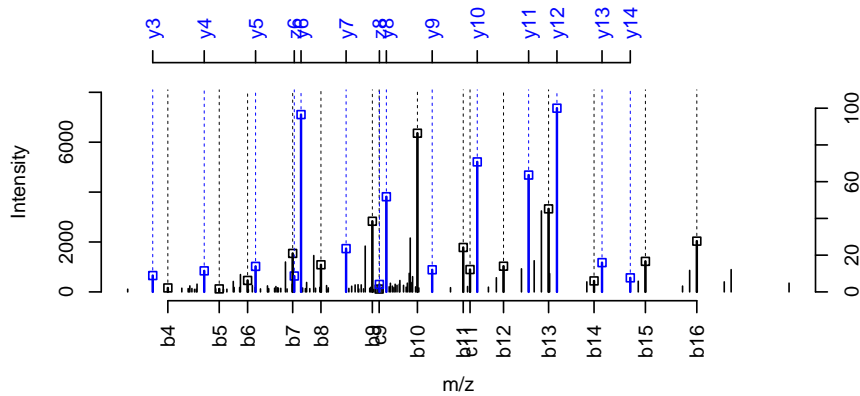
```
$label
[1] "b1" "b2" "b3" "b4" "b5" "b6" "b7" "b8" "b9" "b10" "b11" "b12"
[13] "b13" "b14" "b15" "b16" "b17" "b18" "b19" "y1" "y2" "y3" "y4" "y5"
[25] "y6" "y7" "y8" "y9" "y10" "y11" "y12" "y13" "y14" "y15" "y16" "y17"
[37] "y18" "y19" "c1" "c2" "c3" "c4" "c5" "c6" "c7" "c8" "c9" "c10"
[49] "c11" "c12" "c13" "c14" "c15" "c16" "c17" "c18" "c19" "z1" "z2" "z3"
[61] "z4" "z5" "z6" "z7" "z8" "z9" "z10" "z11" "z12" "z13" "z14" "z15"
[73] "z16" "z17" "z18" "z19"
```

```
$score
[1] -1
```

```
$sequence
[1] "TAFDEAIAELDTLSEESYK"
```

```
$fragmentIon
      b      y      c      z
1  102.0550 147.1128 119.0815 130.0863
2  173.0921 310.1761 190.1186 293.1496
3  320.1605 397.2082 337.1870 380.1816
4  435.1874 526.2508 452.2140 509.2242
5  564.2300 655.2933 581.2566 638.2668
6  635.2671 742.3254 652.2937 725.2988
7  748.3512 855.4094 765.3777 838.3829
8  819.3883 956.4571 836.4148 939.4306
9  948.4309 1071.4841 965.4574 1054.4575
10 1061.5149 1184.5681 1078.5415 1167.5416
11 1176.5419 1313.6107 1193.5684 1296.5842
12 1277.5896 1384.6478 1294.6161 1367.6213
13 1390.6736 1497.7319 1407.7002 1480.7053
14 1477.7056 1568.7690 1494.7322 1551.7424
15 1606.7482 1697.8116 1623.7748 1680.7850
16 1735.7908 1812.8385 1752.8174 1795.8120
17 1822.8229 1959.9069 1839.8494 1942.8804
18 1985.8862 2030.9440 2002.9127 2013.9175
19 2113.9811 2131.9917 2131.0077 2114.9652
```

```
> par(op)
```



5 Quantification

For an overview on Quantitative Proteomics read [1, 2]. 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 repetitions. For the moment there are limited options due to the availability of machine time and the limits of the technologies.

5.1 Label-free methods on protein level

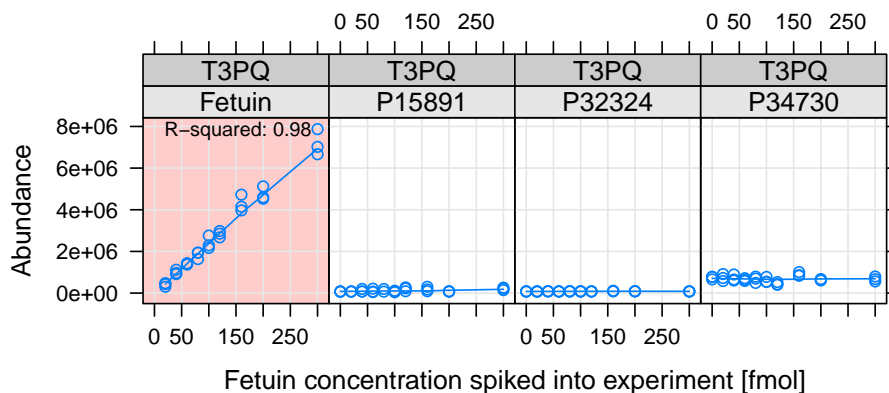
The data set fetuinLFQ contains a subset of our results described in [3]. The example below shows a visualization using trellis plots. It graphs the abundance of four protein in dependency from the fetuin concentration spiked into the sample.

```
> library(lattice)
> data(fetuinLFQ)
> cv<-1:7/10
> 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)
+     }
+   }
+ })

```

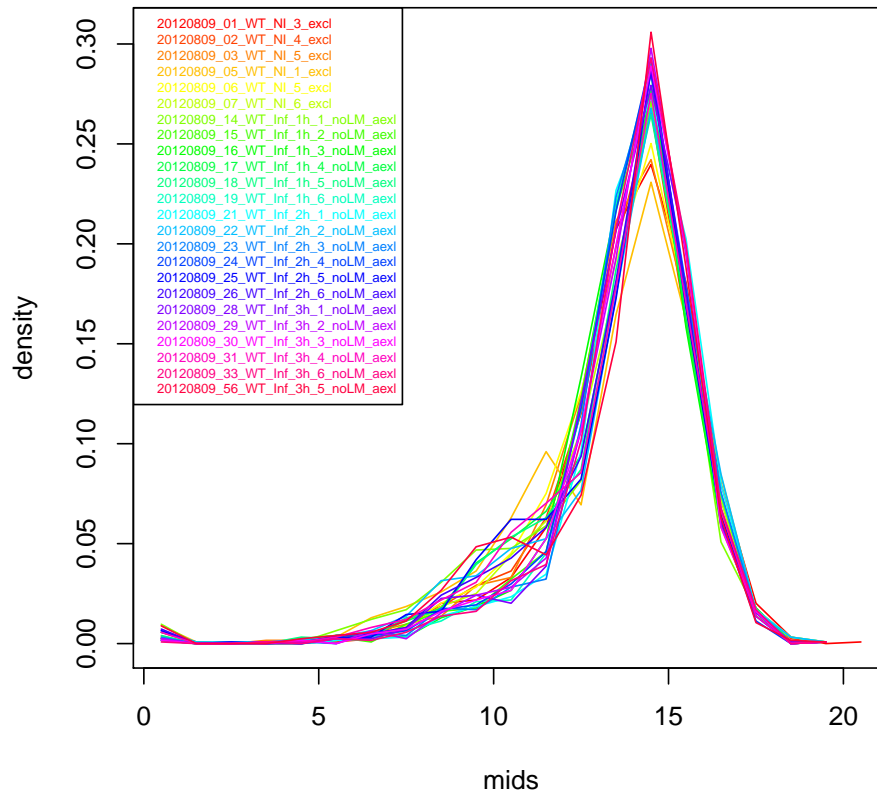


The plot shows the estimated concentration of the four proteins using the top three most intense peptides. The Fetuin peptides are spiked in with increasing concentration while the three other yeast proteins are kept stable in the background.

5.2 pgLFQ – LCMS based label-free quantification

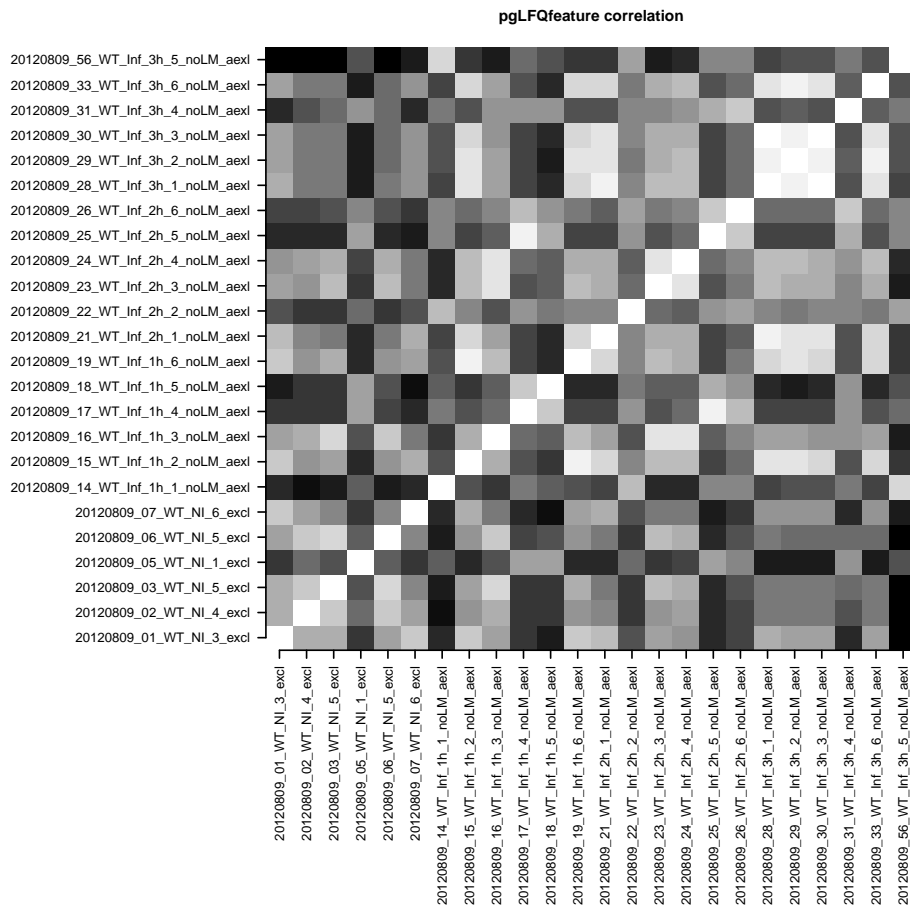
LCMS based label-free quantification is a very popular method to extract relative quantitative information from mass spectrometry experiments. At the FGCZ we use the software ProgenesisLCMS for this workflow <http://www.nonlinear.com/products/progenesis/lc-ms/overview/>. Progenesis is a graphical software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

```
> 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=FALSE)
+     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=FALSE)
+     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)
```



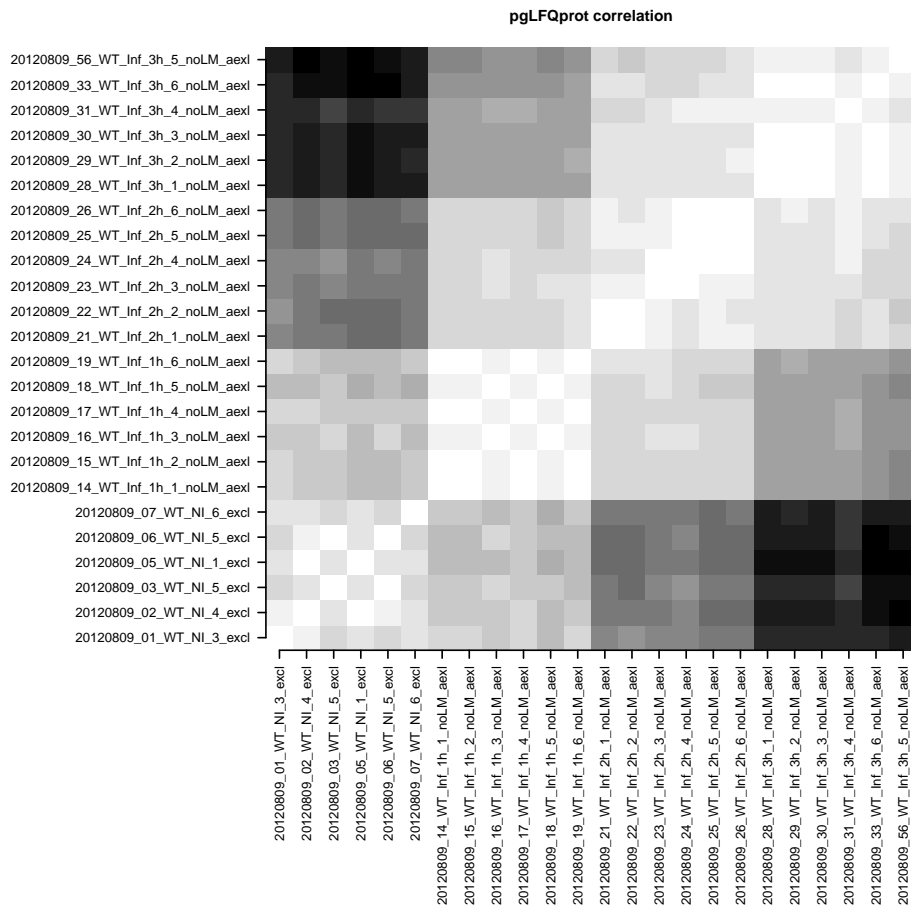
The featureDensityPlot shows the normalized signal intensity distribution (asinh transformed) over 24 LCMS runs which are aligned in this experiment.

```
> 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)
```



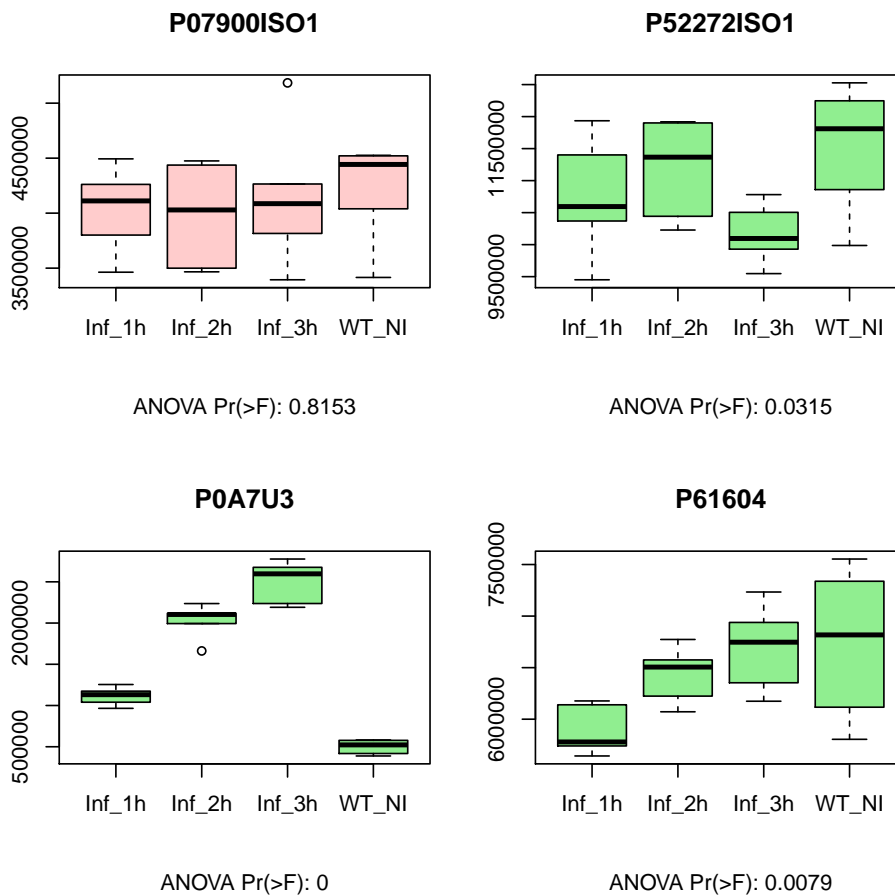
This image plot shows the correlation between runs on feature level (values are asinh transformed). White is perfect correlation while black indicates a poor correlation.

```
> op<-par(mfrow=c(1,1),mar=c(18,18,4,1),cex=0.5)
> image(cor(asinh(pgLFFQprot$"Normalized abundance")),
+       main='pgLFFQprot 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. Striking 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)
```



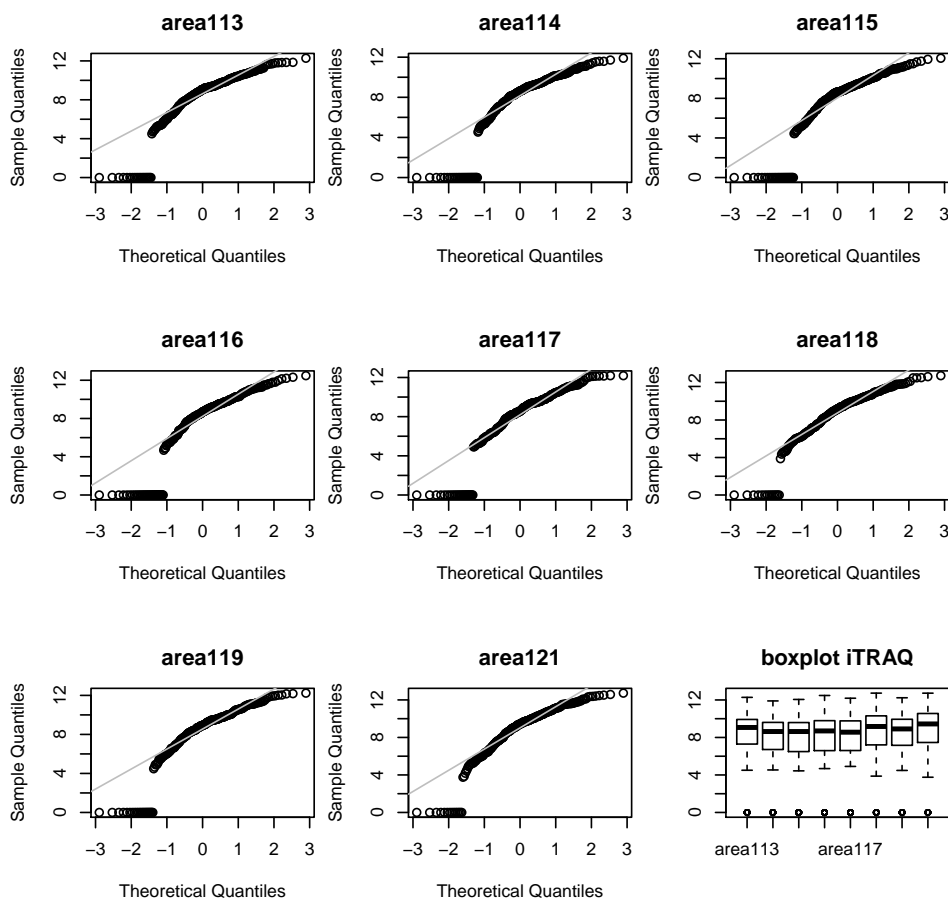
This figure shows the result for four proteins which either differ significantly in expression accross conditions (green boxplots) using an analysis of variance test, or non differing protein expression (red boxplot).

5.3 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)

5.3.1 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 quality 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.

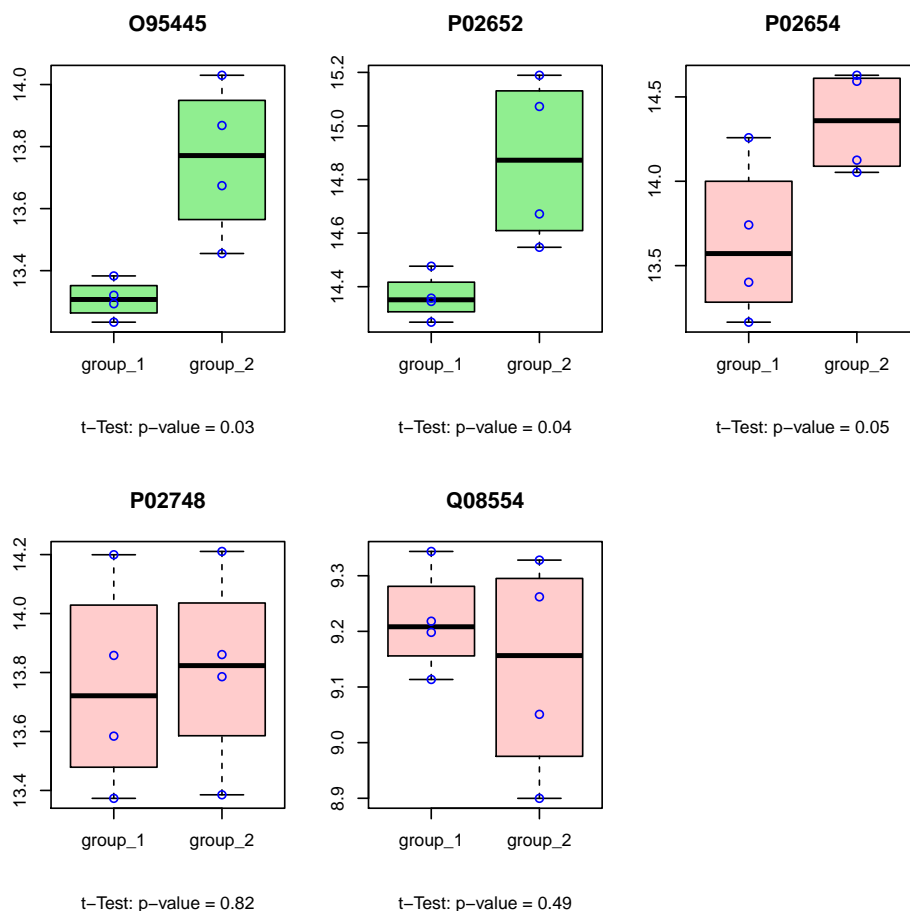
5.3.2 On Protein Level

```
> 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=FALSE)
+       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.

5.3.3 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=FALSE)
> q[1:10,]

```

	name	p_value	Group1.area113	Group1.area114	
1	095445 AFLTPR	0.056	1705.43	1459.10	
2	095445 DGLCVPR	0.161	2730.41	1852.90	
3	095445 MKDGLCVPR	0.039	28726.38	15409.81	
4	095445 NQEACELSNN	0.277	4221.31	4444.28	
5	095445 SLTSCLDISK	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				

6 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 ppp function.

```

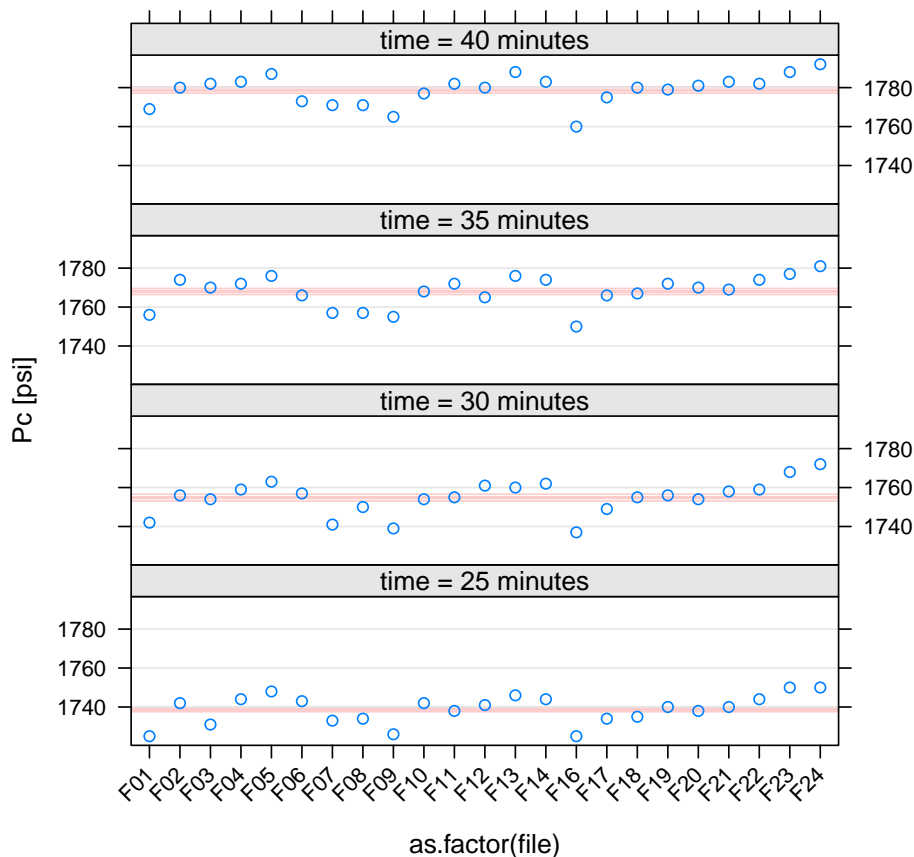
> data(pressureProfile)
> ppp(pressureProfile)

```


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

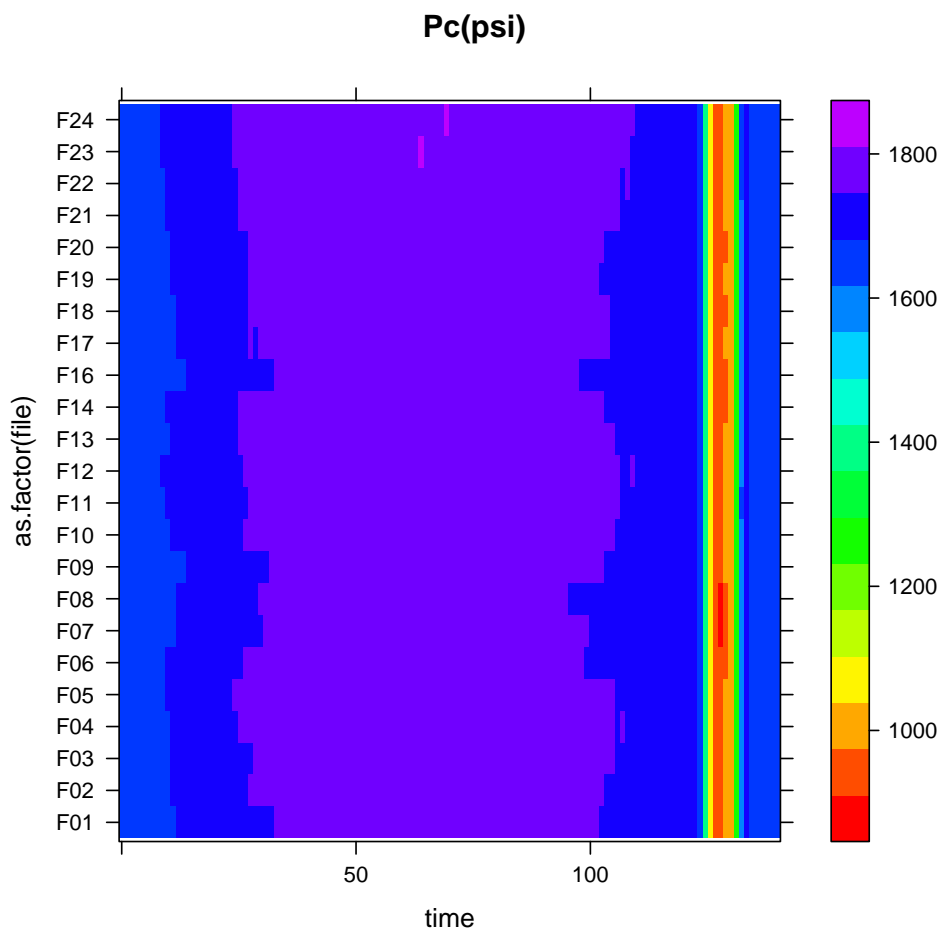
```
> pp.data<-pps(pressureProfile, time=seq(25,40,by=5))
> print(xyplot(Pc ~ as.factor(file) | paste("time =",
+   as.character(time), "minutes"),
+   panel = function(x, y){
+     m<-sum(y)/length(y)
+     m5<-(max(y)-min(y))*0.05
+     panel.abline(h=c(m-m5,m,m+m5),
+       col=rep("#ffcccc",3),lwd=c(1,2,1))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



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



References

- [1] M. Bantscheff, S. Lemeer, M. M. Savitski, and B. Kuster. Quantitative mass spectrometry in proteomics: critical review update from 2007 to the present. *Anal Bioanal Chem*, 404(4):939–965, Sep 2012.
- [2] S. Cappadona, P. R. Baker, P. R. Cutillas, A. J. Heck, and B. van Breukelen. Current challenges in software solutions for mass spectrometry-based quantitative proteomics. *Amino Acids*, 43(3):1087–1108, Sep 2012.
- [3] J. Grossmann, B. Roschitzki, C. Panse, C. Fortes, S. Barkow-Oesterreicher, D. Rutishauser, and R. Schlapbach. Implementation and evaluation of relative and absolute quantification in shotgun proteomics with label-free methods. *J Proteomics*, 73(9):1740–1746, Aug 2010. [DOI:10.1016/j.jpro.2010.05.011] [PubMed:20576481].

- [4] P. Roepstorff and J. Fohlman. Proposal for a common nomenclature for sequence ions in mass spectra of peptides. *Biomed. Mass Spectrom.*, 11(11):601, Nov 1984. [DOI:10.1002/bms.1200111109] [PubMed:6525415].