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

**Christian Panse**
Functional Genomics Center Zurich

**Jonas Grossmann**
Functional Genomics Center Zurich

**Abstract**

**protViz** is an R package for quality checks, visualizations, and analysis of mass spectrometry data from proteomics experiments. The package is developed, tested, and used at the Functional Genomics Center Zurich. We use it mainly for prototyping, teaching, and having *fun* with proteomics data, but it can also be used to analyze small data sets and, with some patience, large ones.

*Keywords*: proteomics, mass spectrometry, fragment-ion.

## Contents

# 1. Related Work

*The method of choice in proteomics is mass spectrometry.* Several R packages already deal with mass spectrometry data. Some of them are listed here:

- **OrgMassSpec**: Organic Mass Spectrometry
- **MSnbase** package (basic functions for mass spec data including quant aspect with iTRAQ data)
  https://bioconductor.org/packages/MSnbase/
- **plgem** – spectral counting quantification, applicable to MudPIT experiments
  https://bioconductor.org/packages/plgem/
- **synapter** – MSe (Hi3 = Top3 Quantification) for Waters Q-TOF data acquired in MSe mode
  https://bioconductor.org/packages/synapter/
- **mzR**
  https://bioconductor.org/packages/mzR/
- isobar iTRAQ/TMT quantification package
  https://bioconductor.org/packages/isobar/
- **readMzXmlData**
  https://CRAN.R-project.org/package=readMzXmlData
- **rawDiag** - an R package supporting rational LC-MS method optimization for bottom-up proteomics on multiple OS platforms (Trachsel, Panse, Kockmann, Wolski, Grossmann, and Schlapbach 2018)

# 2. Get Data In – Preprocessing

The most time-consuming and challenging part of data analysis and visualization is shaping the data so that they can easily be processed further. This package intentionally leaves this part out because it depends very much on the local infrastructure. Moreover, we also use commercial tools to analyze data and export the results into formats accessible to R. We provide importers for some of these formats; with little effort, exports of other tools can be brought into a similar format, which makes the package usable together with a variety of tools.

## 2.1. Identification - In-silico from Proteins to Peptides

For demonstration, we use a sequence of peptides derived from a tryptic digest of the Swiss-Prot `FETUA_BOVIN Alpha-2-HS-glycoprotein` protein (P12763).

The `Fasta` Rcpp module reads a FASTA file and performs the tryptic digest of the protein sequences, applying the rule: cleave after arginine (R) and lysine (K) unless followed by proline (P).

```
R> library(protViz)
R> fname <- system.file("extdata", name = 'P12763.fasta', package = "protViz")
R> fa <- Fasta$new(fname)
```

Print the first 60 characters of P12763.

```
R> substr(fa$getSequences(), 1, 60)

[1] "MKSFVLLFCLAQLWGCHSIPLDPVAGYKEPACDDPDTEQAALAAVDYINKHLPRGYKHTL"

R> (fetuin <- fa$getTrypticPeptides())

 [1] "MK"
 [2] "SFVLLFCLAQLWGCHSIPLDPVAGYK"
 [3] "EPACDDPDTEQAALAAVDYINK"
 [4] "HLPR"
 [5] "GYK"
 [6] "HTLNQIDSVK"
 [7] "VWPR"
 [8] "RPTGEVYDIEIDTLETTCHVLDPTPLANCSVR"
 [9] "QQTQHAVEGDCDIHVLK"
[10] "QDGQFSVLFTK"
[11] "CDSSPDSAEDVR"
[12] "K"
[13] "LCPDCPLLAPLNDSR"
[14] "VVHAVEVALATFNAESNGSYLQLVEISR"
[15] "AQFVPLPVSVSVEFAVAATDCIAK"
[16] "EVVDPTK"
[17] "CNLLAEK"
[18] "QYGFCK"
[19] "GSVIQK"
[20] "ALGGEDVR"
[21] "VTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR"
[22] "AHYDLR"
[23] "HTFSGVASVESSSGEAFHVGK"
[24] "TPIVGQPSIPGGPVR"
[25] "LCPGR"
[26] "IR"
[27] "YFK"
[28] "I"
```

## 3. Peptide Identification

*Peptides are the currency of proteomics.* In proteomics, proteins are digested into so-called peptides because peptides are much easier to handle biochemically than proteins. Proteins differ greatly in nature: some are very sticky, others are soluble in aqueous solutions, and

yet others reside only in membranes. Proteins are therefore chopped into peptides, assuming that each protein yields many well-behaved peptides that can be measured with the mass spectrometer. This step introduces another problem, the so-called protein inference problem, which this package does not address.

### 3.1. Computing Mass and Hydrophobicity of a Peptide Sequence

`parentIonMass` computes the mass of the singly charged peptide ion `[M+H]+` of an amino acid sequence.

```
R> mass <- parentIonMass(fetuin)
```

The `ssrc` function derives a measure for the hydrophobicity based on the method described in (Krokhin, Craig, Spicer, Ens, Standing, Beavis, and Wilkins 2004).

```
R> hydrophobicity <- ssrc(fetuin)
```

The content of `mass` and `hydrophobicity` can be seen in Table 1.

| peptide | mass | hydrophobicity |
|---|---|---|
| MK | 278.15 | |
| SFVLLFCLAQLWGCHSIPLDPVAGYK | 2991.53 | 71.74 |
| EPACDDPDTEQAALAAVDYINK | 2406.08 | 25.81 |
| HLPR | 522.31 | 6.05 |
| GYK | 367.20 | 2.16 |
| HTLNQIDSVK | 1154.62 | 18.37 |
| VWPR | 557.32 | 9.55 |
| RPTGEVYDIEIDTLETTCHVLDPTPLANCSVR | 3671.77 | 46.69 |
| QQTQHAVEGDCDIHVLK | 1977.94 | 21.45 |
| QDGQFSVLFTK | 1269.65 | 32.22 |
| CDSSPDSAEDVR | 1337.53 | 2.08 |
| K | 147.11 | |
| LCPDCPLLAPLNDSR | 1740.84 | 31.62 |
| VVHAVEVALATFNAESNGSYLQLVEISR | 3016.57 | 54.51 |
| AQFVPLPVSVSVEFAVAATDCIAK | 2519.32 | 53.75 |
| EVVDPTK | 787.42 | 7.78 |
| CNLLAEK | 847.43 | 16.51 |
| QYGFCK | 802.36 | 10.05 |
| GSVIQK | 631.38 | 9.83 |
| ALGGEDVR | 816.42 | 10.35 |
| VTCTLFQTQPVIPQPQPDGAEAEAPSAVPDAAGPTPSAAGPPVASVVVGPSVVAVPLPLHR | 6015.13 | 39.37 |
| AHYDLR | 774.39 | 11.42 |
| HTFSGVASVESSSGEAFHVGK | 2120.00 | 27.95 |
| TPIVGQPSIPGGPVR | 1474.84 | 23.26 |
| LCPGR | 602.31 | 3.61 |
| IR | 288.20 | |
| YFK | 457.24 | 7.91 |
| I | 132.10 | |

Table 1: parent ion mass and hydrophobicity values of the tryptic digested protein extttP12763.

Figure 1 graphs the parent ion mass versus the hydrophobicity value of each in-silico tryptic peptide of the `FETUA_BOVIN` (P12763) protein.

```
R> plot(hydrophobicity ~ mass,
+    log = 'xy', pch = 16, col = '#88888888', cex = 2,
```

```
+    main = "sp|P12763|FETUA_BOVIN Alpha-2-HS-glycoprotein",
+    sub = 'tryptic peptides')
R> text(mass, hydrophobicity, fetuin, pos = 2, cex = 0.5, col = '#CCCCCC88')
```

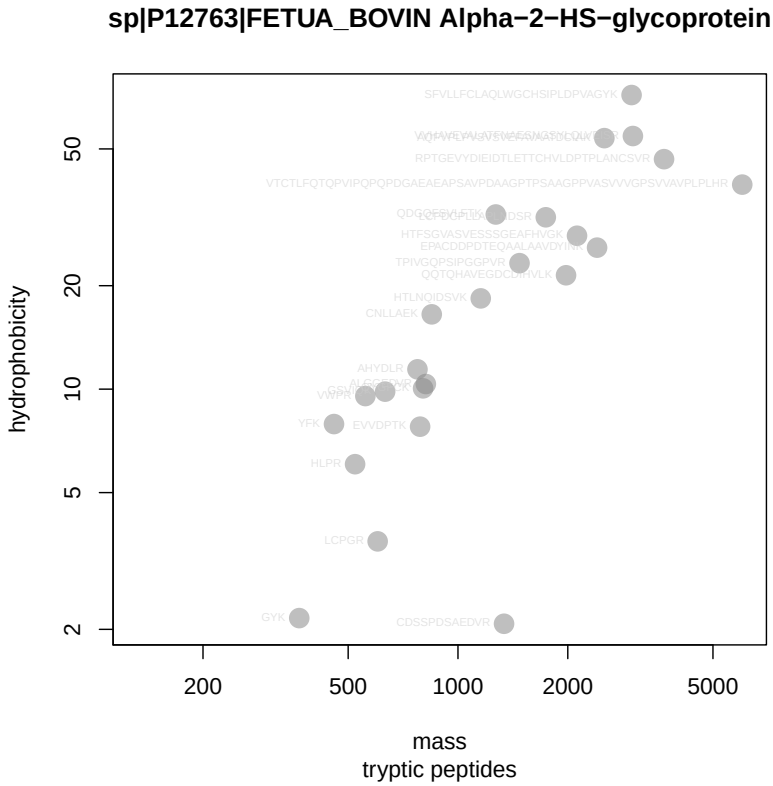


Figure 1: Parent ion mass versus hydrophobicity (`ssrc`) of the in-silico tryptic peptides of P12763 (log scales).

### 3.2. In-silico Peptide Fragmentation

The computation of the fragment ions of a peptide follows the rules proposed in (Roepstorff and Fohlman 1984). Besides the `b` and `y` ions, the `FUN` argument of `fragmentIon` defines which ions are computed. The ions computed by default are defined in the function `defaultIon`. Any other kind of fragment ion can be defined, e.g., c and z ions for ETD or b and y ions for CID.

```
R> 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))
}
<bytecode: 0x560477b22f20>
<environment: namespace:protViz>
```

Figure 2 shows the in-silico fragment ions of three tryptic peptides.

```
R> peptides <- c('HTLNQIDSVK', 'ALGGEDVR', 'TPIVGQPSIPGGPVR')
R> pims <- parentIonMass(peptides)
R> fis <- fragmentIon(peptides)
R> op <- par(mfrow = c(3, 1))
R> rv <- mapply(FUN = function(fi, pim, peptide){
+      plot(0, 0,
+          xlab = 'm/z', ylab = '',
+          xlim = range(c(fi$b, fi$y)),
+          ylim = c(0, 1),
+          type = 'n', axes = FALSE,
+          sub = paste(peptide, "/", round(pim, 2), "Da"))
+
+      axis(1, fi$b, round(fi$b, 2))
+
+      pepSeq <- strsplit(peptide, "")
+      axis(3, fi$b, pepSeq[[1]])
+
+      abline(v = fi$b, col = 'red', lwd = 2)
+      abline(v = fi$y, col = 'blue', lwd = 2)
+      abline(v = fi$c, col = 'orange')
+      abline(v = fi$z, col = 'cyan')
+    }, fis, pims, peptides)
R> par(op)
```

The next lines compute the singly and doubly charged fragment ions of the `HTLNQIDSVK` peptide, which are usually the ones used for an identification. A doubly charged ion carries one additional proton, i.e., $m/z_{2+} = (m/z_{1+} + m_p)/2$.

```
R> Proton <- 1.007276
R> (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
```

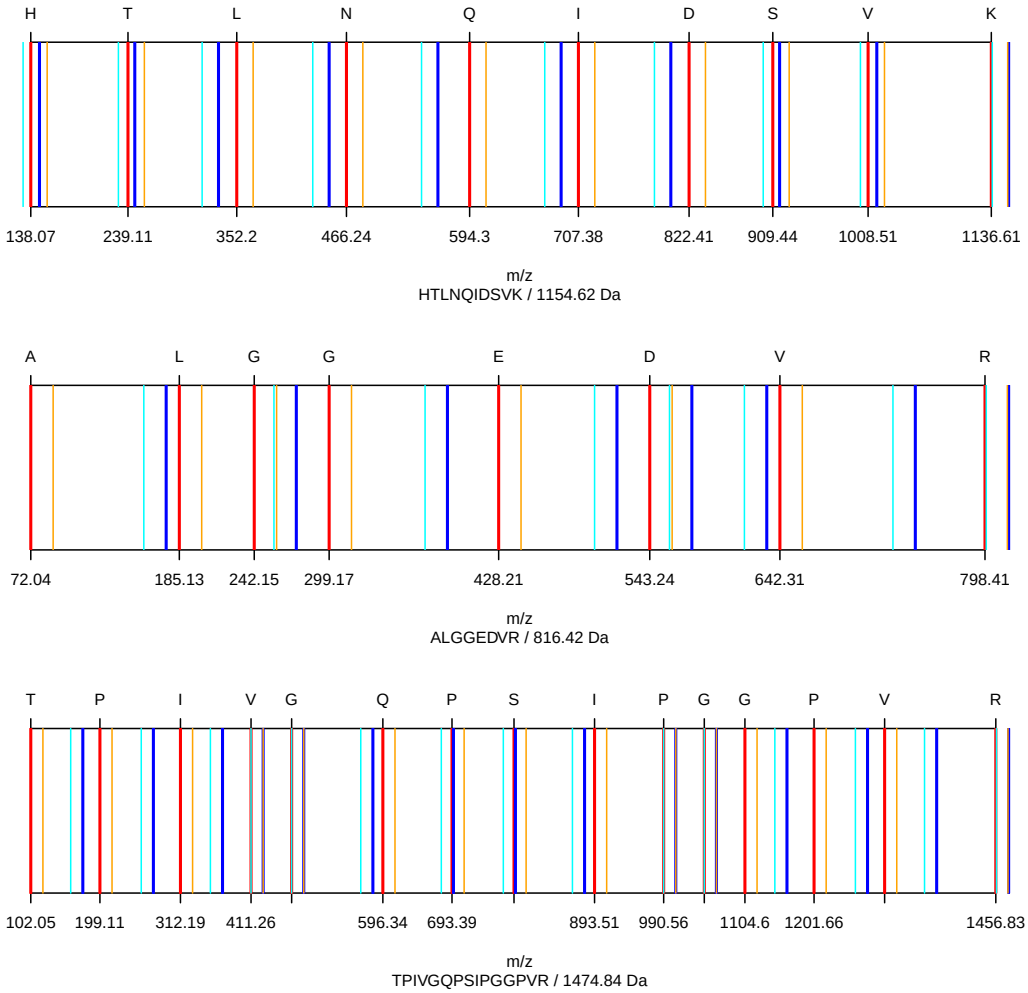


Figure 2: In-silico fragment ions of three tryptic peptides of P12763: b (red), y (blue), c (orange), and z (cyan) ions. The top axis shows the residue at each b ion position.

```
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
```

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

```
           b         y         c         z
1   69.53673  74.06004  78.05001  65.54676
2  120.06057 123.59424 128.57385 115.08097
3  176.60260 167.11026 185.11588 158.59698
4  233.62407 224.62373 242.13734 216.11045
5  297.65336 281.16576 306.16663 272.65248
6  354.19539 345.19505 362.70866 336.68177
7  411.70886 402.21651 420.22213 393.70324
8  455.22487 458.75854 463.73815 450.24527
9  504.75908 509.28238 513.27235 500.76911
10 568.80656 577.81184 577.31983 569.29856
```

### 3.3. Peptide Sequence – Fragment Ion Matching

Given a peptide sequence and a tandem mass spectrum (MS2), the function `psm` computes the in-silico fragment ions of the peptide and determines for each of them the closest peak in the MS2. If the difference between the in-silico mass and the measured mass is within the mass accuracy of the instrument, the fragment ion is considered a hit.

```
R> peptideSequence <- 'HTLNQIDSVK'
R> 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))
```

With `plot = TRUE`, `psm` draws the mass error of each fragment ion (Figure 3).

```
R> op <- par(mfrow = c(1, 2))
R> m <- psm(peptideSequence, spec, plot = TRUE)
R> par(op)
```

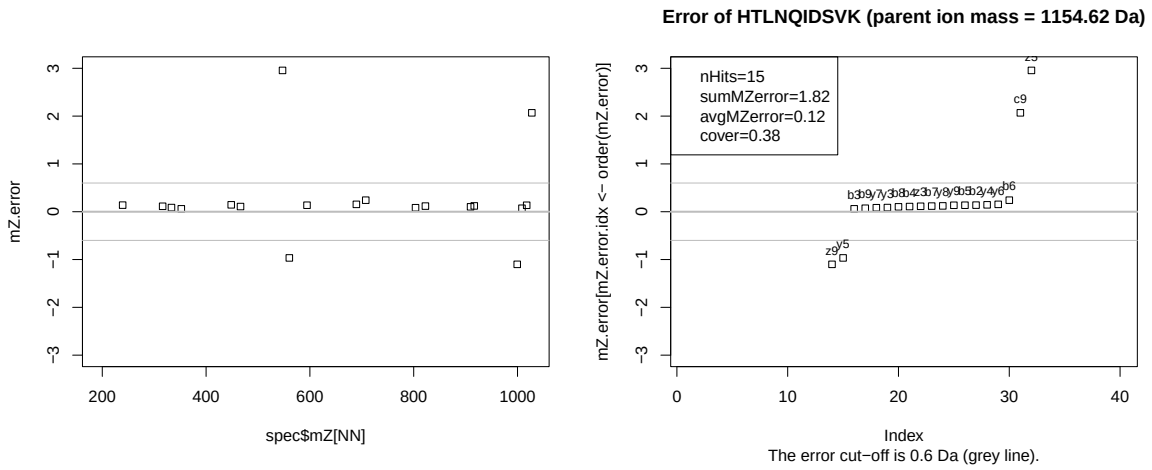


Figure 3: Mass error between the singly charged b, y, c, and z ions (`defaultIon`) of `HTLNQIDSVK` and the closest peaks of the MS2 `spec`: versus the matched peak m/z (left) and sorted by error with ion labels and match statistics (right). The grey lines indicate the default fragment ion tolerance of 0.6 Da.

`psm` returns, among others, the mass error of each fragment ion (`mZ.Da.error`), the index of the matched peak (`idx`), and the ion label (`label`). The matched ions are:

```
R> m$label[abs(m$mZ.Da.error) < 0.6]
```

```
 [1] "b2" "b3" "b4" "b5" "b6" "b7" "b8" "b9" "y3" "y4" "y6" "y7" "y8"
[14] "y9" "z3"
```

Internally, `psm` determines the closest peaks with a fast binary search for the nearest neighbor in the sorted `mZ` vector, as provided by `findNN`. The fragment ions are considered column by column of the fragment ion table:

```
R> fi <- fragmentIon(peptideSequence)[[1]]
R> idx <- findNN(unlist(fi), spec$mZ)
R> all.equal(idx, m$idx)
```

```
[1] TRUE
```

The more theoretical ions match the measured spectrum (within the instrument's error tolerance), the more likely it is that the spectrum indeed originates from the inferred peptide, and therefore that the protein is identified.

## 3.4. Modifications

`genMod` enumerates the possible modification sites of peptide sequences. The modification pattern is a vector of one-letter amino acid codes; its first element is a placeholder for "unmodified", and element $i$ defines modification id $i - 1$, i.e., the residue it can be attached to.

The result is encoded as one string of digits per variant, with one digit per residue giving the modification id (0 = unmodified). Below, we consider the substitutions serine to asparagine and vice versa, which differ by ±27.0109 Da.

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

[[1]]
[1] "0000000000000000000" "0000000000000200000" "0000000000000000100"
[4] "0000000000000100200"

[[2]]
[1] "0000000000000000000" "0000000000000100000" "0000000000000000100"
[4] "0000000000000100100"
```

The digit strings can be passed to `fragmentIon` using the `modified` argument, while `modification` gives the mass shift of each modification id. Adding 27.0109 Da to the serine at position 14 of `TAFDEAIAELDTLSEESYK` gives the same fragment ions as the unmodified `TAFDEAIAELDTLNEESYK`, and vice versa:

```
R> fi <- fragmentIon(c('TAFDEAIAELDTLSEESYK',
+      'TAFDEAIAELDTLNEESYK', 'TAFDEAIAELDTLSEESYK',
+      'TAFDEAIAELDTLNEESYK'),
+          modified = c('0000000000000200000',
+          '0000000000000100000', '0000000000000000000',
+          '0000000000000000000'),
+      modification = as.numeric(m$mono))
R> all.equal(fi[[1]], fi[[4]], check.attributes = FALSE)

[1] TRUE

R> all.equal(fi[[2]], fi[[3]], check.attributes = FALSE)

[1] TRUE
```

### 3.5. Labeling Peaklists

The `peakplot` function (Panse, Gerrits, and Schlapbach 2009) labels tandem mass spectra with the matching fragment ions (Figure 4).

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

$mZ.Da.error
 [1]  232.331344  161.294234   14.225824   -0.032616   -0.143306
 [6]    0.032244    0.054604   -0.004076   -0.071746   -0.084536
[11]   -0.097076   -0.038856   -0.061816    0.004554   -0.122336
[16]   -0.139626   -1.071256  -18.783686 -146.878646  187.273499
[21]   24.210169    0.048669    0.177779    0.027939    0.049579
[26]    0.052379    0.044579    0.036749    0.043189   -0.035101
[31]   -0.061011    0.000729   -0.092081    2.011029   -8.412111
[36]    7.195579  -63.841531 -164.889211  215.304795  144.267685
[41]   -2.800725  -17.059165    2.034875    2.264105    4.008125
[46]    1.292875   -0.003965  -13.612585   -0.060925  -17.065405
[51]    3.897535    3.000405  -17.148885  -17.166175  -18.097805
[56]  -35.810235 -163.905195  204.300048   41.236718   17.075218
[61]   -0.843372   -1.091812    0.129908   17.078928   -0.372162
[66]  -16.539502   -1.044962   -1.000952   -1.409062   -2.995122
[71]   16.934468   19.037578    8.614438   24.222128  -46.814982
[76] -147.862662

$mZ.ppm.error
 [1]  2.276532e+06  9.318407e+05  4.443342e+04 -7.494702e+01
 [5] -2.539851e+02  5.075660e+01  7.296574e+01 -4.974443e+00
 [9] -7.564705e+01 -7.963713e+01 -8.250960e+01 -3.041352e+01
[13] -4.445040e+01  3.026484e+00 -7.488007e+01 -7.920687e+01
[17] -5.791093e+02 -9.331667e+03 -6.860308e+04  1.272993e+06
[21]  7.805297e+04  1.225277e+02  3.378218e+02  4.263587e+01
[25]  6.444386e+01  5.935833e+01  4.532837e+01  3.345395e+01
[29]  3.564687e+01 -2.618263e+01 -4.321937e+01  4.781134e-01
[33] -5.770282e+01  1.165934e+03 -4.572174e+03  3.621478e+03
[37] -3.102183e+04 -7.637286e+04  1.808046e+06  7.588299e+05
[41] -8.306147e+03 -3.772366e+04  3.500821e+03  3.470990e+03
[45]  5.236793e+03  1.545734e+03 -4.106862e+00 -1.262129e+04
[49] -5.104441e+01 -1.318183e+04  2.768725e+03  1.971690e+03
[53] -1.038832e+04 -9.644849e+03 -9.694247e+03 -1.764117e+04
[57] -7.595171e+04  1.570497e+06  1.406678e+05  4.491332e+04
[61] -1.656190e+03 -1.710589e+03  1.726789e+02  1.973544e+04
[65] -3.850849e+02 -1.529356e+04 -8.747728e+02 -7.562373e+02
[69] -1.010347e+03 -1.986529e+03  1.072648e+04  1.114745e+04
[73]  4.725878e+03  1.229618e+04 -2.293808e+04 -6.903096e+04

$idx
 [1]   1   1   1   3  14  21  38  49  64  87  91  97 102 106 110 113
[17] 115 116 116   1   1   2  12  25  41  53  70  89  94  99 104 107
[33] 108 111 114 116 116 116   1   1   1   3  16  24  41  52  67  88
```

```
[49]  93  97 104 107 110 113 115 116 116   1   1   2  11  22  40  53
[65]  68  88  93  98 103 106 108 111 114 116 116 116

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

*R> peakplot("TAFDEAIAELDTLSEESYK", msms[[2]])*

```
$mZ.Da.error
 [1]  245.264254  174.227144   27.158734   14.444434    0.021404
 [6]   -0.111266   -0.039926   -0.021626   -0.121916   -8.079236
[11]   -0.158376   -0.153156   -0.094316   -0.022946   -0.186736
[16]   -0.092226   -0.120456   -0.151686 -128.246646  200.206409
```

```
[21]   37.143079    0.078909    0.062269    0.129769    0.103729
[26]    0.060869   -0.051451  -18.048351   -0.027511   -0.025601
[31]   -0.006211    0.020529   -0.048781   -0.024771   -9.166311
[36]    6.953579  -45.209531 -146.257211  228.237705  157.200595
[41]   10.132185   -2.582115    1.626855    2.722405    9.009025
[46]   -1.130895    1.216385   13.347315   -3.671525    0.960295
[51]  -17.120865    3.020205  -17.213285  -17.118775  -17.147005
[56]  -17.178235 -145.273195  217.232958   54.169628   17.105458
[61]   -0.833452   -1.260332   -0.899352   -3.098942   -1.173512
[66]   -1.021802   -0.939162   -1.007752   -1.377062   -3.022622
[71]   16.977768   17.001778    7.860238   23.980128  -28.182982
[76] -129.230662

$mZ.ppm.error
 [1]  2.403257e+06  1.006558e+06  8.482850e+04  3.319130e+04
 [5]  3.793488e+01 -1.751484e+02 -5.335196e+01 -2.639286e+01
 [9] -1.285450e+02 -7.611043e+03 -1.346114e+02 -1.198789e+02
[13] -6.782037e+01 -1.552813e+01 -1.162198e+02 -5.313198e+01
[17] -6.608212e+01 -7.638202e+01 -6.066594e+04  1.360904e+06
[21]  1.197483e+05  1.986591e+02  1.183257e+02  1.980319e+02
[25]  1.397352e+02  7.115774e+01 -5.379332e+01 -1.684426e+04
[29] -2.322450e+01 -1.948903e+01 -4.485617e+00  1.370673e+01
[33] -3.109508e+01 -1.458996e+01 -5.056331e+03  3.547913e+03
[37] -2.226035e+04 -6.860121e+04  1.916651e+06  8.268554e+05
[41]  3.004915e+04 -5.709941e+03  2.798859e+03  4.173588e+03
[45]  1.177069e+04 -1.352074e+03  1.259905e+03  1.237534e+04
[49] -3.076091e+03  7.417604e+02 -1.216230e+04  2.020566e+03
[53] -1.060078e+04 -9.766434e+03 -9.319787e+03 -8.576627e+03
[57] -6.817113e+04  1.669915e+06  1.847849e+05  4.499286e+04
[61] -1.636709e+03 -1.974616e+03 -1.239974e+03 -3.696333e+03
[65] -1.249174e+03 -9.690310e+02 -8.043928e+02 -7.772361e+02
[69] -1.006903e+03 -2.041339e+03  1.094110e+04  1.011538e+04
[73]  4.376983e+03  1.234257e+04 -1.399411e+04 -6.110297e+04

$idx
 [1]   1   1   1   3  11  20  39  45  64  90  96 106 116 121 126 129
[17] 131 133 133   1   1   2   7  24  38  49  65  90  97 110 115 122
[33] 123 127 130 132 133 133   1   1   1   3  13  23  40  47  67  91
[49]  98 108 116 122 126 129 131 133 133   1   1   2   6  21  36  47
[65]  62  90  95 108 113 121 123 127 130 132 133 133

$label
 [1] "b1"  "b2"  "b3"  "b4"  "b5"  "b6"  "b7"  "b8"  "b9"  "b10" "b11"
[12] "b12" "b13" "b14" "b15" "b16" "b17" "b18" "b19" "y1"  "y2"  "y3"
[23] "y4"  "y5"  "y6"  "y7"  "y8"  "y9"  "y10" "y11" "y12" "y13" "y14"
[34] "y15" "y16" "y17" "y18" "y19" "c1"  "c2"  "c3"  "c4"  "c5"  "c6"
[45] "c7"  "c8"  "c9"  "c10" "c11" "c12" "c13" "c14" "c15" "c16" "c17"
```

```
[56] "c18" "c19" "z1"  "z2"  "z3"  "z4"  "z5"  "z6"  "z7"  "z8"  "z9"
[67] "z10" "z11" "z12" "z13" "z14" "z15" "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

R> par(op)
```

### 3.6. A Simple Peptide Search Engine

The following code snippet combines the functions above into a simple peptide search engine. Its arguments are a mass spectrum `x`, i.e., a list of `mZ` and `intensity` vectors, and a character vector of candidate peptide sequences. The best candidate is the one with the most matched fragment ions; the fraction of the total ion intensity explained by the matched peaks is reported as well.

```
R> .peptideFragmentIonSpectrumMatch <- function (x,
+                              peptideSet,
+                              fragmentIonMassToleranceDa = 0.1)
+  {
+    ## Here we ignore the peptide mass
```

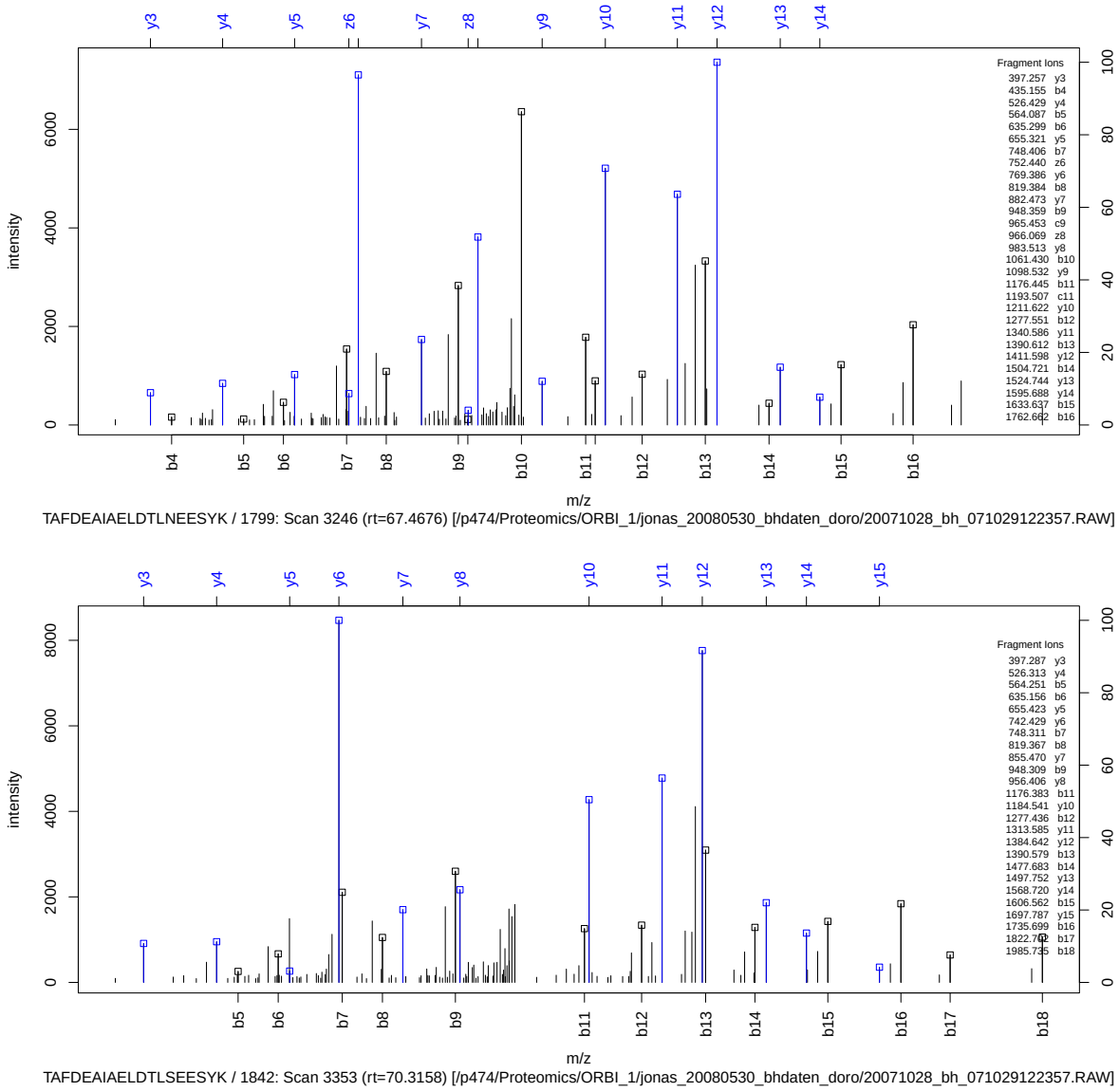


Figure 4: Annotated tandem mass spectra of the `msms` data set: `TAFDEAIAELDTLNEESYK` (top) and `TAFDEAIAELDTLSEESYK` (bottom).

```
+    # peptideMassTolerancePPM = 5
+    # query.mass <- ((x$pepmass[1] * x$charge)) - (1.007825 * (x$charge - 1))
+    # eps <- query.mass * peptideMassTolerancePPM * 1e-06
+    # pimIdx <- parentIonMass(peptideSequence)
+    # lower <- findNN(query.mass - eps, pimIdx)
+    # upper <- findNN(query.mass + eps, pimIdx)
+
+    rv <- lapply(peptideSet, FUN = psm, spec = x, plot = FALSE) |>
+      lapply(FUN = function(p) {
+        ## fragment ions considered as hits
+        hit <- abs(p$mZ.Da.error) < fragmentIonMassToleranceDa
+        ## peaks matched by these fragment ions
+        peakIdx <- unique(p$idx[hit])
+        intensityRatio <- sum(x$intensity[peakIdx]) / sum(x$intensity)
+
+        ## derive objectives for a good match
+        data.frame(nHits = sum(hit), intensityRatio = intensityRatio)
+      }) |>
+      Reduce(f = rbind)
+
+    idx.tophit <- which(rv$nHits == max(rv$nHits))[1]
+
+    data.frame(peptideMatch = peptideSet[idx.tophit],
+               nHits = max(rv$nHits),
+               nPeaks = length(x$mZ),
+               intensityRatio = rv$intensityRatio[idx.tophit]
+    )
+  }
```

Define a set of peptide sequences:

```
R> peptideSet <- c("ELIVSK", 'TAFDEAIAELDTLNEESYK', 'TAFDEAIAELDTLSEESYK')
```

Generate an in-silico tandem mass spectrum with random intensities:

```
R> set.seed(1)
R> mZ <- fragmentIon("TAFDEAIAELDTLNEESYK")[[1]] |>
+    unlist() |> sort()
R> intensity <- mZ |> length() |> sample()
R> msms.insilico <- list(mZ = mZ, intensity = intensity)
```

Generate reversed peptide sequences as decoys:

```
R> peptideSet.rev <- peptideSet |>
+    sapply(FUN = function(x){
+      strsplit(x, "")[[1]] |> rev() |> paste0(collapse = "")
+    })
```

The output is the best-matching peptide for each spectrum.

```
R> lapply(list(msms[[1]], msms[[2]], msms.insilico),
+         FUN = .peptideFragmentIonSpectrumMatch,
+         peptideSet = c(peptideSet, peptideSet.rev),
+         fragmentIonMassToleranceDa = 0.05) |>
+    Reduce(f = rbind)

         peptideMatch nHits nPeaks intensityRatio
1 TAFDEAIAELDTLNEESYK    14    116      0.3028307
2 TAFDEAIAELDTLSEESYK    10    133      0.2602073
3 TAFDEAIAELDTLNEESYK    76     76      1.0000000
```

# 4. Quantification

For an overview of quantitative proteomics, read Bantscheff, Lemeer, Savitski, and Kuster (2012); Cappadona, Baker, Cutillas, Heck, and van Breukelen (2012). The authors are aware that meaningful statistics usually require a much higher number of biological replicates. In almost all cases, there are no more than three to six replicates, since machine time is limited and the technologies have their limits.

## 4.1. Label-free methods on protein level

The data set `fetuinLFQ` contains a subset of the results described in Grossmann, Roschitzki, Panse, Fortes, Barkow-Oesterreicher, Rutishauser, and Schlapbach (2010). The example below uses trellis plots to show the abundance of four proteins as a function of the fetuin concentration spiked into the sample (Figure 5).

```
R> library(lattice)
R> data(fetuinLFQ)
R> cv <- 1 - 1:7 / 10
R> print(xyplot(abundance ~ conc | prot * method,
+      groups = prot,
+      xlab = "Fetuin concentration spiked into experiment [fmol]",
+      ylab = "Abundance",
+      aspect = 1,
+      par.settings = list(strip.background = list(col = rgb(cv, cv, cv))),
+      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)
+          }
+      }
+  ))
```

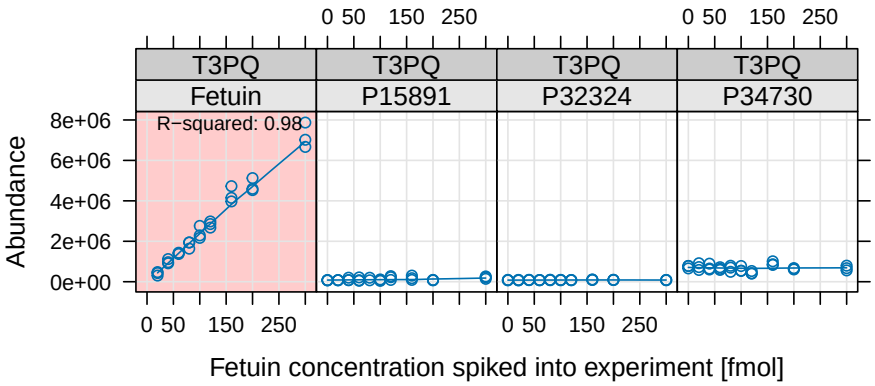


Figure 5: Abundance of fetuin (red panel) and three yeast background proteins estimated with the top three most intense peptides (T3PQ), versus the spiked-in fetuin concentration.

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 yeast proteins are kept stable in the background.

### 4.2. pgLFQ – LCMS based label-free quantification

LC-MS based label-free quantification (LFQ) 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 aligns several LC-MS experiments, extracts signal intensities from the LC-MS maps, and annotates the master map with peptide and protein labels.

```
R> data(pgLFQfeature)
R> data(pgLFQprot)
R> featureDensityPlot <- function(data, n = ncol(data), nbins = 30){
+      my.col <- hcl.colors(n, palette = "Dark 3")
+      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
+      )
+  }
R> featureDensityPlot(asinh(pgLFQfeature$"Normalized abundance"),
+      nbins = 25)
```

The `featureDensityPlot` (Figure 6) shows the normalized signal intensity distribution (asinh transformed) over the 24 LC-MS runs aligned in this experiment.

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

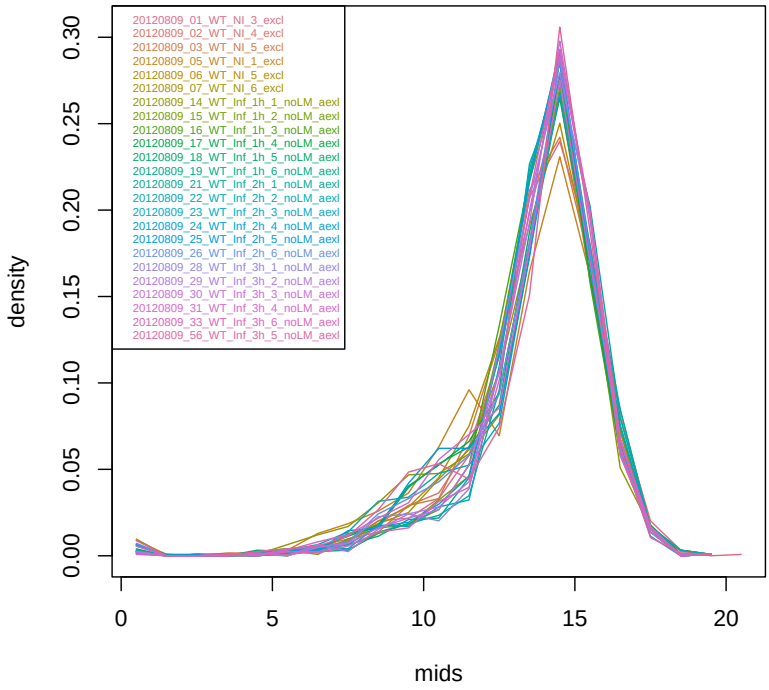


Figure 6: Distribution of the normalized feature intensities (`asinh` transformed) of the 24 aligned LC-MS runs.

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

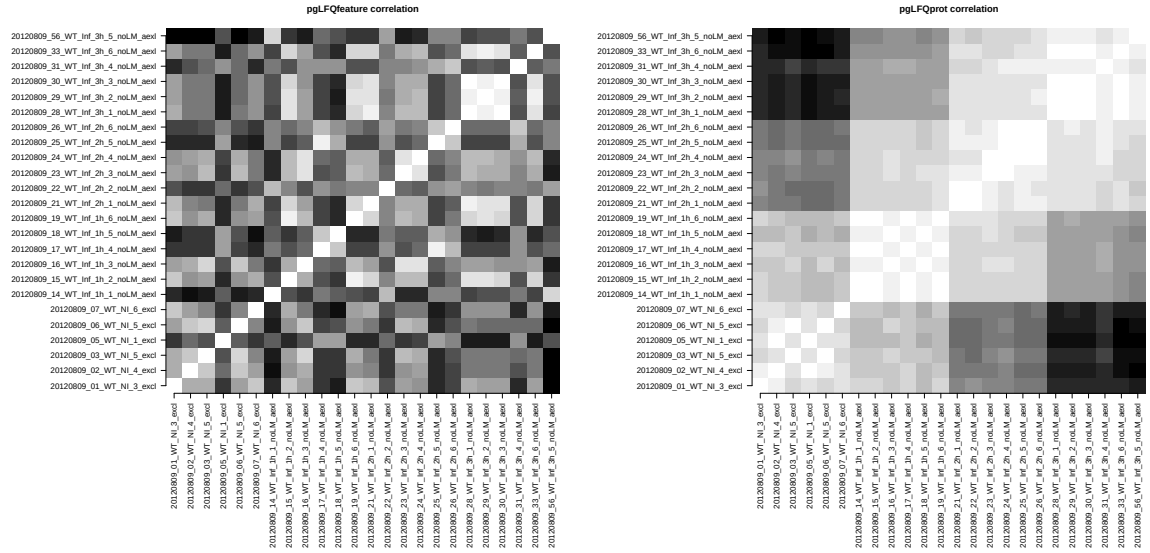


Figure 7: Correlation between runs on feature level (left) and protein level (right); values are `asinh` transformed. White is perfect correlation, black indicates poor correlation.

Figure 7 shows the correlation between runs on feature and protein level. Strikingly, the six biological replicates of each condition cluster very well.

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

Figure 8 shows the result for four proteins whose expression either differs significantly across conditions according to an analysis of variance (green boxplots) or does not differ (red boxplots).

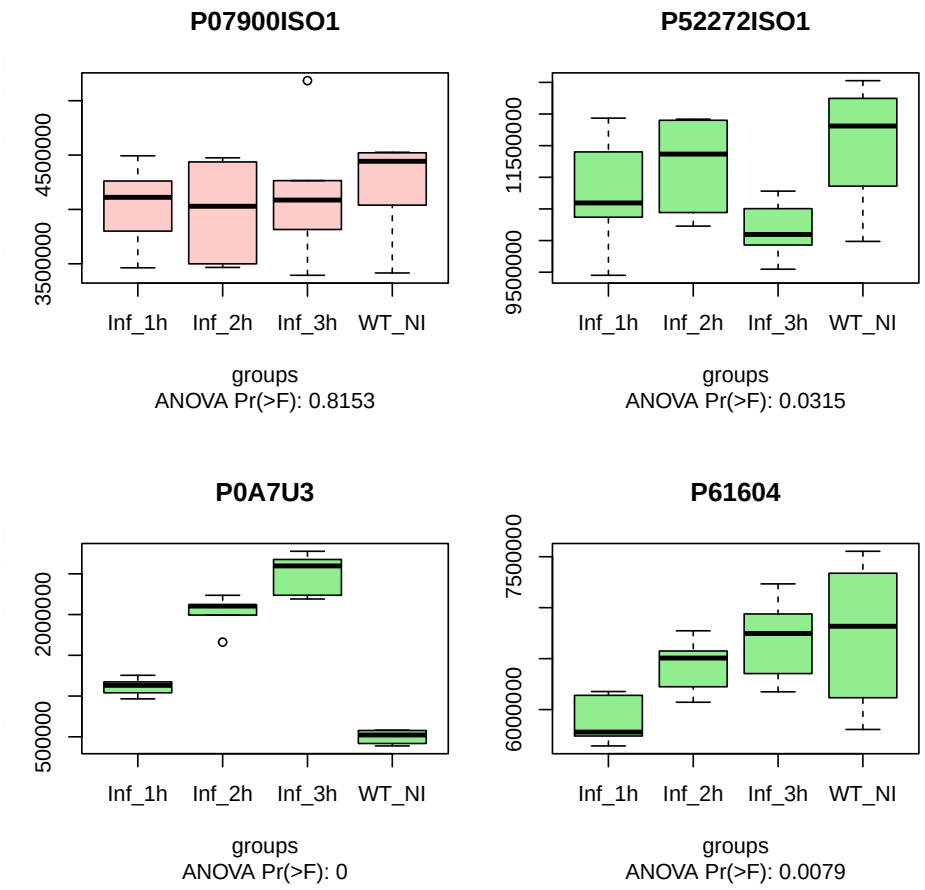


Figure 8: Protein abundance per condition for four proteins; green boxplots indicate a significant difference between conditions (ANOVA, $p < 0.05$), red boxplots no significant difference.

### 4.3. iTRAQ – Two Group Analysis

The data for the next section is an iTRAQ 8-plex experiment in which two conditions are compared, each with four biological replicates.

*Sanity Check*

```
R> data(iTRAQ)
R> channels <- grep("^area", names(iTRAQ), value = TRUE)
R> op <- par(mfrow = c(3, 3), mar = c(6, 4, 3, 0.5))
R> for (ch in channels){
+      qqnorm(asinh(iTRAQ[, ch]), main = ch)
+      qqline(asinh(iTRAQ[, ch]), col = 'grey')
+  }
R> b <- boxplot(asinh(iTRAQ[, channels]), main = 'boxplot iTRAQ', las = 2)
R> par(op)
```

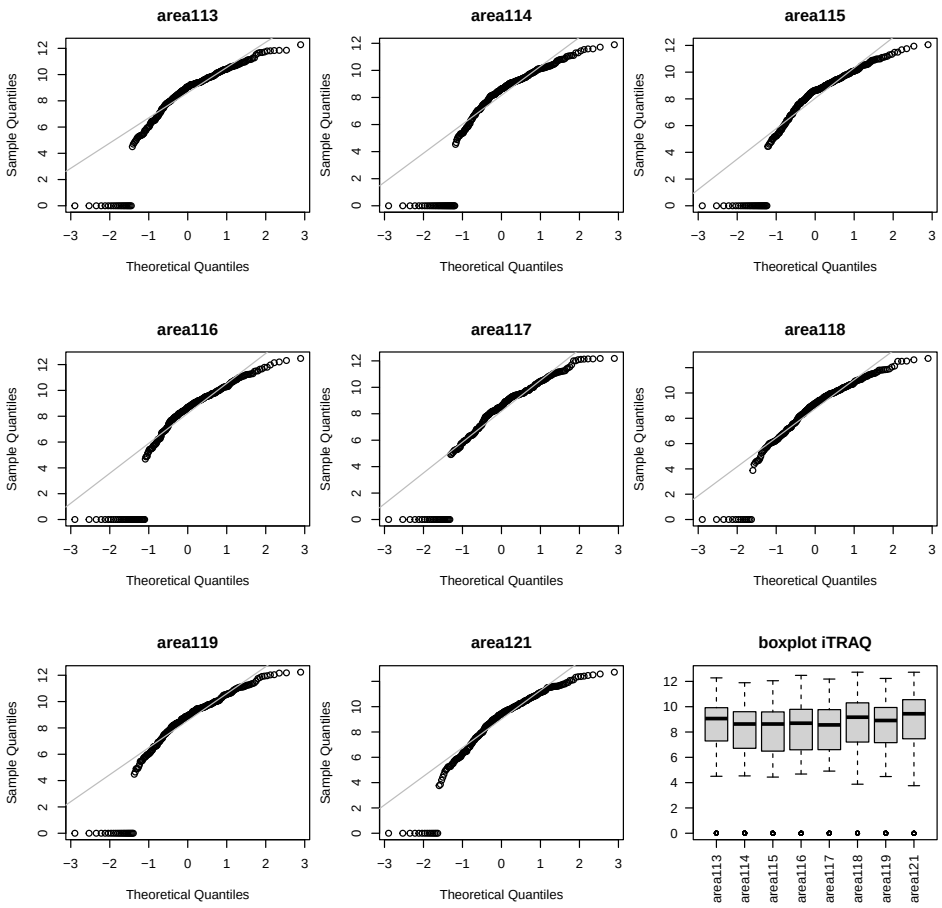


Figure 9: Q-Q plots of the `asinh` transformed reporter ion areas of each iTRAQ channel against a normal distribution, and a boxplot of all channels.

A first quality check is whether all reporter ion channels have the same distribution. Figure 9 shows Q-Q plots of the individual reporter channels against a normal distribution, followed by a boxplot of all channels.

*On Protein Level*

`iTRAQ2GroupAnalysis` sums the reporter ion areas of each protein (`INDEX`), transforms them using `FUN`, and compares the two groups with a $t$ test (Figure 10).

```
R> group1 <- channels[1:4]
R> group2 <- channels[5:8]
R> op <- par(mfrow = c(2, 3), mar = c(6, 3, 4, 1))
R> qProt <- iTRAQ2GroupAnalysis(data = iTRAQ,
+       group1 = group1,
+       group2 = group2,
+       INDEX = iTRAQ$prot,
+       FUN = asinh,
+       plot = TRUE)
R> par(op)
```

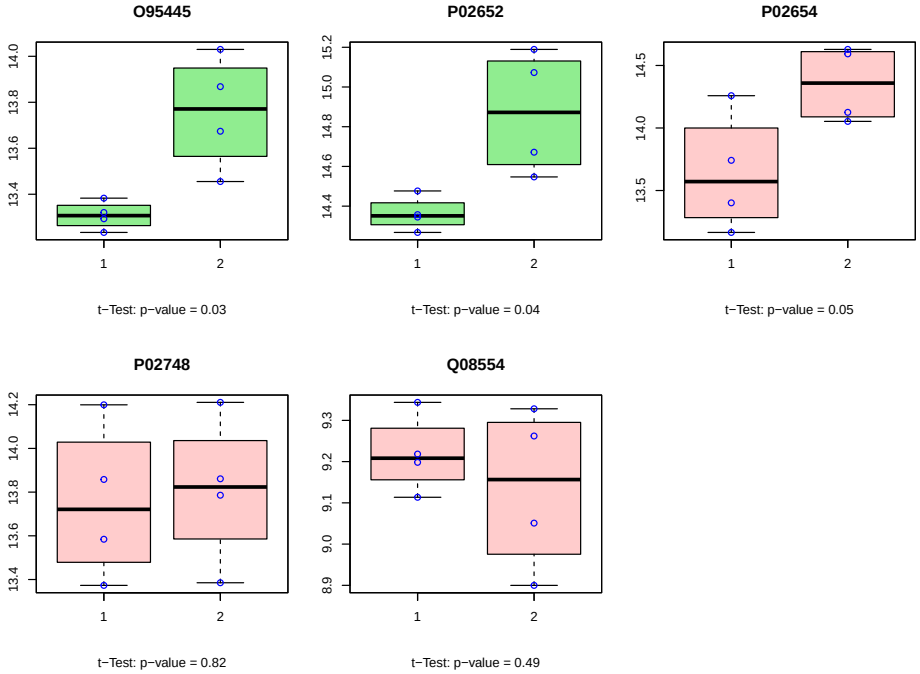


Figure 10: Protein abundance (summed and `asinh` transformed reporter ion areas) of the five proteins in group 1 (`area113`–`area116`) and group 2 (`area117`–`area121`). Green boxplots indicate a significant difference ($t$ test, $p < 0.05$).

```
R> qProt[, c("name", "p_value")]
```

```
    name p_value
1 O95445   0.032
2 P02652   0.041
3 P02654   0.052
4 P02748   0.825
5 Q08554   0.485
```

*On Peptide Level*

The same can be done on peptide level by using the protein and peptide sequence as `INDEX`.

```
R> qPeptide <- iTRAQ2GroupAnalysis(data = iTRAQ,
+      group1 = group1,
+      group2 = group2,
+      INDEX = paste(iTRAQ$prot, iTRAQ$peptide),
+      FUN = asinh,
+      plot = FALSE)
R> qPeptide[1:10, c("name", "p_value")]
```

```
                          name p_value
1                 O95445 AFLLTPR   0.035
2                 O95445 DGLCVPR   0.137
3                O95445 MKDGLCVPR   0.020
4               O95445 NQEACELSNN   0.173
5                O95445 SLTSCLDSK   0.020
6    P02652 AGTELVNFLSYFVELGTQPA   0.938
7   P02652 AGTELVNFLSYFVELGTQPAT   0.837
8  P02652 AGTELVNFLSYFVELGTQPATQ   0.238
9    P02652 EPCVESLVSQYFQTVTDYGK   0.090
10               P02652 EQLTPLIK   0.028
```

## 5. Pressure Profiles QC

A common problem with mass spectrometry setups is the poor reliability of the high-pressure pump. The following graphics provide visualizations for quality control.

The `ppp` function plots the column pressure `Pc` versus time of a sequence of runs, grouped by gradient length. Figure 11 shows a single run.

```
R> data(pressureProfile)
R> ppp(pressureProfile[pressureProfile$filename == "F01", ])
```

The Trellis `xyplot` in Figure 12 shows the pressure `Pc` of each instrument run at specified points in time $(25, 30, \ldots)$.

```
R> pp.data <- pps(pressureProfile, time = seq(25, 40, by = 5))
R> print(xyplot(Pc ~ as.factor(file) | paste("time =",
```

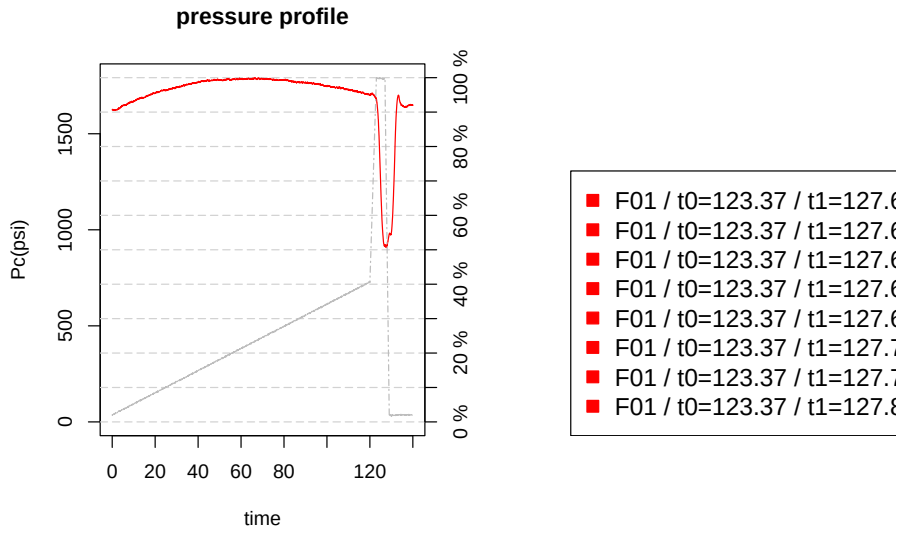


Figure 11: Pressure profile of run `F01` drawn with `ppp`.

```
+       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 red lines indicate the mean +/- 5% of the pressure range.',
+       scales = list(x = list(rot = 45)),
+       data = pp.data))
```

While each panel of the `xyplot` above shows the data at a given point in time, the level plot in Figure 13 gives an overview of the whole pressure profile data.

```
R> pp.data <- pps(pressureProfile, time = seq(0, 140, length = 128))
R> print(levelplot(Pc ~ time * as.factor(file),
+       main = 'Pc [psi]',
+       xlab = 'time [min]',
+       ylab = 'run',
+       data = pp.data,
+       col.regions = hcl.colors(100, palette = "viridis")))
```

The **protViz** package has also been used in Grossmann *et al.* 2010; Nanni, Panse, Gehrig, Mueller, Grossmann, and Schlapbach 2013; Panse, Trachsel, Grossmann, and Schlapbach

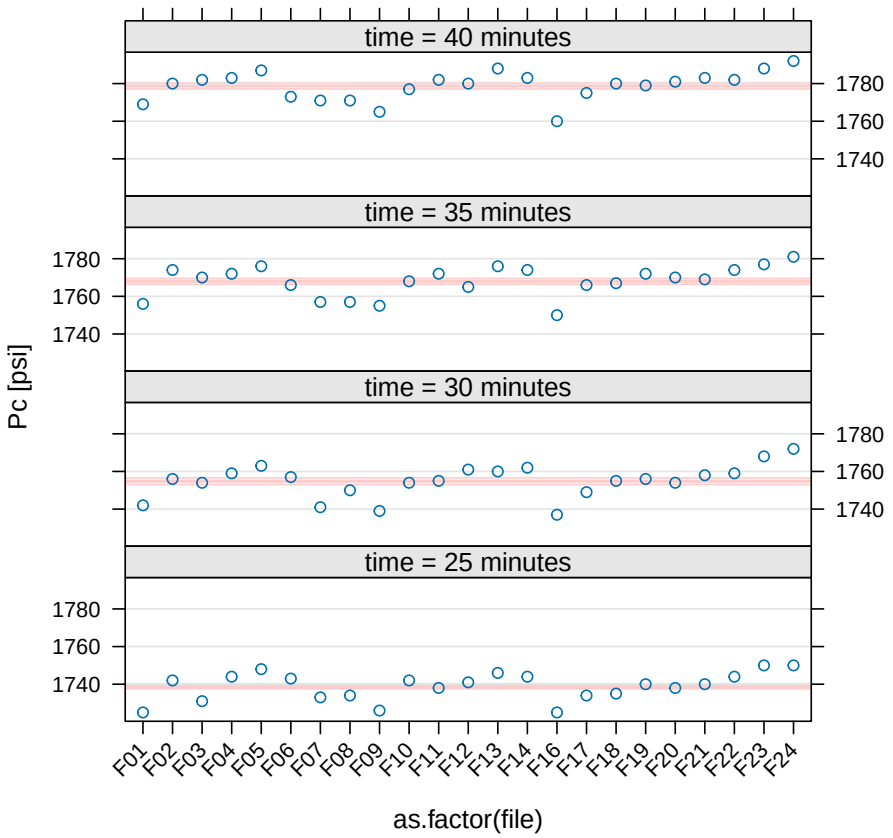


Figure 12: Column pressure of each run at 25, 30, 35, and 40 minutes. The red lines indicate the mean $\pm$ 5 % of the pressure range.

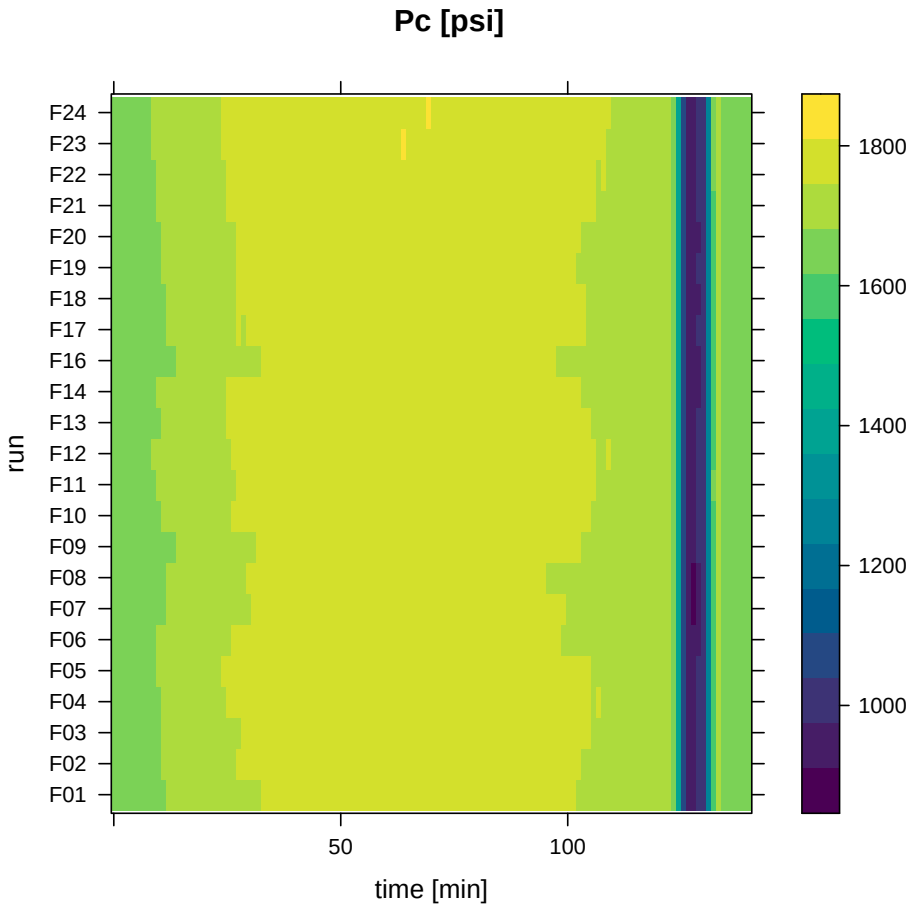


Figure 13: Column pressure of all runs (rows) over time.

2015; Kockmann, Trachsel, Panse, Wahlander, Selevsek, Grossmann, Wolski, and Schlapbach 2016; Bilan, Leutert, Nanni, Panse, and Hottiger 2017; Egloff, Zimmermann, Arnold, Hutter, Morger, Opitz, Poveda, Keserue, Panse, Roschitzki, and Seeger 2018; Gehrig, Nowak, Panse, Leutert, Grossmann, Schlapbach, and Hottiger 2020; Kockmann and Panse 2021.

## References

Bantscheff M, Lemeer S, Savitski MM, Kuster B (2012). "Quantitative mass spectrometry in proteomics: critical review update from 2007 to the present." *Anal Bioanal Chem*, **404**(4), 939–965. doi:10.1007/s00216-012-6203-4.

Bilan V, Leutert M, Nanni P, Panse C, Hottiger MO (2017). "Combining Higher-Energy Collision Dissociation and Electron-Transfer/Higher-Energy Collision Dissociation Fragmentation in a Product-Dependent Manner Confidently Assigns Proteomewide ADP-Ribose Acceptor Sites." *Anal. Chem.*, **89**(3), 1523–1530. doi:10.1021/acs.analchem.6b03365.

Cappadona S, Baker PR, Cutillas PR, Heck AJ, van Breukelen B (2012). "Current challenges in software solutions for mass spectrometry-based quantitative proteomics." *Amino Acids*, **43**(3), 1087–1108. doi:10.1007/s00726-012-1289-8.

Egloff P, Zimmermann I, Arnold FM, Hutter CA, Morger D, Opitz L, Poveda L, Keserue HA, Panse C, Roschitzki B, Seeger M (2018). "Engineered Peptide Barcodes for In-Depth Analyses of Binding Protein Ensembles." doi:10.1101/287813. URL https://doi.org/10.1101/287813.

Gehrig PM, Nowak K, Panse C, Leutert M, Grossmann J, Schlapbach R, Hottiger MO (2020). "Gas-Phase Fragmentation of ADP-Ribosylated Peptides: Arginine-Specific Side-Chain Losses and Their Implication in Database Searches." *Journal of the American Society for Mass Spectrometry*, **32**(1), 157–168. doi:10.1021/jasms.0c00040. URL https://doi.org/10.1021/jasms.0c00040.

Grossmann J, Roschitzki B, Panse C, Fortes C, Barkow-Oesterreicher S, Rutishauser D, Schlapbach R (2010). "Implementation and evaluation of relative and absolute quantification in shotgun proteomics with label-free methods." *J Proteomics*, **73**(9), 1740–1746. doi:10.1016/j.jprot.2010.05.011.

Kockmann T, Panse C (2021). "The rawrr R Package: Direct Access to Orbitrap Data and Beyond." *Journal of Proteome Research*. doi:10.1021/acs.jproteome.0c00866. URL https://doi.org/10.1021/acs.jproteome.0c00866.

Kockmann T, Trachsel C, Panse C, Wahlander A, Selevsek N, Grossmann J, Wolski WE, Schlapbach R (2016). "Targeted proteomics coming of age - SRM, PRM and DIA performance evaluated from a core facility perspective." *Proteomics*, **16**(15-16), 2183–2192. doi:10.1002/pmic.201500502.

Krokhin OV, Craig R, Spicer V, Ens W, Standing KG, Beavis RC, Wilkins JA (2004). "An improved model for prediction of retention times of tryptic peptides in ion pair reversed-phase HPLC: its application to protein peptide mapping by off-line HPLC-MALDI MS." *Mol. Cell Proteomics*, **3**(9), 908–919. doi:10.1074/mcp.M400031-MCP200.

Nanni P, Panse C, Gehrig P, Mueller S, Grossmann J, Schlapbach R (2013). "PTM MarkerFinder, a software tool to detect and validate spectra from peptides carrying post-translational modifications." *Proteomics*, **13**(15), 2251–2255. doi:10.1002/pmic.201300036.

Panse C, Gerrits B, Schlapbach R (2009). "PEAKPLOT: Visualizing Fragmented Peptide Mass Spectra in Proteomics." UseR!2009 conference, Rennes, F, URL https://www.r-project.org/conferences/useR-2009/abstracts/pdf/Panse+Gerrits+Schlapbach.pdf.

Panse C, Trachsel C, Grossmann J, Schlapbach R (2015). "specL–an R/Bioconductor package to prepare peptide spectrum matches for use in targeted proteomics." *Bioinformatics*, **31**(13), 2228–2231. doi:10.1093/bioinformatics/btv105.

Roepstorff P, Fohlman J (1984). "Proposal for a common nomenclature for sequence ions in mass spectra of peptides." *Biomed. Mass Spectrom.*, **11**(11), 601. doi:10.1002/bms.1200111109.

Trachsel C, Panse C, Kockmann T, Wolski WE, Grossmann J, Schlapbach R (2018). "rawDiag - an R package supporting rational LC-MS method optimization for bottom-up proteomics." doi:10.1101/304485. URL https://doi.org/10.1101/304485.

## A. Session information

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

- R version 4.6.2 (2026-10-29), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.utf-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.utf-8`, `LC_COLLATE=en_US.utf-8`, `LC_MONETARY=en_US.utf-8`, `LC_MESSAGES=en_US.utf-8`, `LC_PAPER=en_US.utf-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.utf-8`, `LC_IDENTIFICATION=C`
- Time zone: `Europe/Zurich`
- TZcode source: `system (glibc)`
- Running under: `Debian GNU/Linux forky/sid`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.34.so` ; LAPACK version3.12.0
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: lattice 0.23-1, protViz 0.7.14, xtable 1.8-8

- Loaded via a namespace (and not attached): codetools 0.2-20, compiler 4.6.2, grid 4.6.2, Rcpp 1.1.2, tools 4.6.2

**Affiliation:**

Jonas Grossmann and Christian Panse
Functional Genomics Center Zurich, UZH|ETHZ
Winterthurerstr. 190
CH-8057, Zürich, Switzerland
Telephone: +41-44-63-53912
E-mail: jg@fgcz.ethz.ch, cp@fgcz.ethz.ch
URL: https://fgcz.ch