# msmsTests package
# LC-MS/MS post test filters to improve reproducibility

Josep Gregori, Alex Sanchez, and Josep Villanueva
Vall Hebron Institute of Oncology &
Statistics Dept. Barcelona University
josep.gregori@gmail.com

November 1, 2022

## Contents



## 1 Introduction

The *omics* technologies are in general challenged by a high dimensionality problem, whereby a high number of statistical tests are carried out at the same time on very few samples. The high number of tests that are carried out simultaneously requires a p-value adjustment to control the false discovery rate (FDR). The low number of replicates, combined with a high number of statistical tests easily leads to low reproducibility in the results. Shi et al. addressed this issue in a systematic manner using a large dataset elaborated within the studies of the MicroArray Quality Control Consortium (MAQC) [1]. The MAQC studied the possible sources of the limitations of microarray (MA) studies that precluded obtaining reproducible results across laboratories and platforms. Their conclusions underscored the influence of batch effects and the limitation introduced by

using only the p-value criteria to get ordered lists of differentially expressed features as the main weaknesses in MA studies [2] [3]. This recommendation makes an intuitive appeal to the use of a post-test filter, excluding features that show low p-values but poor effect size or low signal strength, as a mean to guaranty a minimal level of reproducibility.

Label-free differential proteomics is based on comparing the expression of proteins between different biological conditions [4] [5]. The expression of a protein by LC-MS/MS may be obtained by MS peak intensity measures or by the number of spectral counts assigned to that protein in a LC-MS/MS run. Here we concentrate on spectral counts.

In proteomics reproducibility is also of great concern, and the use of such post-test filters may help notably [6].

# 2 An example LC-MS/MS dataset

The example dataset [6] is the result of multiple spiking experiments, showing real LC-MS/MS data . Samples of 500 micrograms of a standard yeast lysate are spiked with 100, 200, 400 and 600fm of a complex mix of 48 equimolar human proteins (UPS1, Sigma-Aldrich). The number of technical replicates vary between 3 to 6.
The dataset consists in an instance of the *MSnSet* class, defined in the MSnbase package [7], a S4 class [8] [9]. This *MSnSet* object contains a spectral counts (SpC) matrix in the *assayData* slot, and a factor treatment in the *phenoData* slot. (See also the expressionSet vignette by vignette("ExpressionSetIntroduction",package="Biobase") [10])

```
> library(msmsTests)
> data(msms.spk)
> msms.spk

MSnSet (storageMode: lockedEnvironment)
assayData: 685 features, 19 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Y500U100_001 Y500U100_002 ... Y500U600_006 (19 total)
  varLabels: treat
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: http://www.ncbi.nlm.nih.gov/pubmed/23770383
Annotation:
- - - Processing information - - -
 MSnbase version: 1.8.0

> dim(msms.spk)

[1] 685  19

> head(pData(msms.spk))
```

```
             treat
Y500U100_001  U100
Y500U100_002  U100
Y500U100_003  U100
Y500U100_004  U100
Y500U200_001  U200
Y500U200_002  U200
```

```
> table(pData(msms.spk)$treat)

U100 U200 U400 U600
   4    6    3    6
```

Although the mix is equimolar the signal strength of each protein is markedly different, allowing to cover the full range of SpC values, what makes it specially worth in this sort of experiments:

```
> msms.spc <- exprs(msms.spk)
> treat <- pData(msms.spk)
> idx <- grep("HUMAN",rownames(msms.spc))
> mSpC <- t( apply(msms.spc[idx,],1,function(x) tapply(x,treat,mean)) )
> apply(mSpC,2,summary)

            U100      U200      U400       U600
Min.    0.000000  0.000000  0.000000  0.1666667
1st Qu. 0.000000  0.250000  1.500000  2.1666667
Median  0.500000  1.000000  2.666667  4.6666667
Mean    1.384615  2.752137  6.837607  9.7521368
3rd Qu. 1.500000  3.333333  7.833333 11.4166667
Max.    9.750000 20.500000 43.333333 60.3333333
```

# 3 Differential expression tests on spectral counts

Spectral counts (SpC) is an integer measure which requires of tests suited to compare counts, as the GML methods based in the Poisson distribution, the negative-binomial, or the quasilikelihood [11]

Generally speaking no test is superior to the other. They are just more or less indicated in some cases. The Poisson regression requires the estimation of just one parameter, and is indicated when the number of replicates is low, two or three. A drawback of the Poisson distribution is that its variance equals its mean, and it is not able to explain extra sources of variability a part of the sampling. Quasi-likelihood is a distribution independent GLM, but requires of the estimation of two parameters and hence needs a higher number of replicates, i.e. not less than four. The negative-binomial requires two parameters too, but we may use the implementation of this GLM in the edgeR package [12] which uses an empirical Bayes method to share information across features and may be employed with a restricted number of replicates; in the worst case it limits with the Poisson solution.

# 4 Poisson GLM regression example

When using the Poisson distribution we implicitly accept a model not sensitive to biological variability [11]. So it is just recommended in cases where we have very few replicates, if any, and we do not expect a significant biological variability between samples.

```
> ### Subset to the 200 and 600fm spikings
> fl <- treat=="U200" | treat=="U600"
> e <- msms.spk[,fl]
> table(pData(e))

treat
U100 U200 U400 U600
   0    6    0    6

> ### Remove all zero rows
> e <- pp.msms.data(e)
> dim(e)

[1] 664  12

> ### Null and alternative model
> null.f <- "y~1"
> alt.f <- "y~treat"
> ### Normalizing condition
> div <- apply(exprs(e),2,sum)
> ### Poisson GLM
> pois.res <- msms.glm.pois(e,alt.f,null.f,div=div)
> str(pois.res)

'data.frame':        664 obs. of  3 variables:
 $ LogFC  : num  0.12061 -0.18016 0.40229 0.00168 -0.37157 ...
 $ D      : num  5.22 8.93 4.11e+01 7.72e-04 3.10e+01 ...
 $ p.value: num  2.23e-02 2.80e-03 1.41e-10 9.78e-01 2.59e-08 ...

> ### DEPs on unadjusted p-values
> sum(pois.res$p.value<=0.01)

[1] 44

> ### DEPs on multitest adjusted p-values
> adjp <- p.adjust(pois.res$p.value,method="BH")
> sum(adjp<=0.01)

[1] 26

> ### The top features
> o <- order(pois.res$p.value)
> head(pois.res[o,],20)
```

```
                LogFC         D      p.value
TRFL_HUMAN   1.8306054 150.41594 1.406200e-34
ALBU_HUMAN   1.9625212 104.51026 1.563819e-24
TRFE_HUMAN   1.3100859  81.53741 1.719756e-19
YPL037C      0.7416385  79.62566 4.525092e-19
CAH2_HUMAN   2.3041307  63.81053 1.369793e-15
CATA_HUMAN   1.5343714  50.12093 1.445571e-12
CAH1_HUMAN   4.4542343  47.82775 4.653585e-12
MYG_HUMAN    2.1323062  46.76436 8.005636e-12
YOL086C      0.4022934  41.14620 1.412577e-10
KCRM_HUMAN   1.4294782  37.83644 7.693109e-10
YGR192C     -0.3715658  30.99581 2.585863e-08
ANT3_HUMAN   1.4013022  27.39627 1.657502e-07
GSTA1_HUMAN  2.5495822  27.11728 1.914789e-07
PRDX1_HUMAN  1.6359291  26.61692 2.480616e-07
UBE2C_HUMAN  3.0181352  25.68582 4.017692e-07
NQO1_HUMAN   3.7380273  25.66094 4.069826e-07
CATD_HUMAN   1.6748334  24.58937 7.094204e-07
CYC_HUMAN   32.2452233  24.09871 9.152148e-07
ANXA5_HUMAN  2.5156349  22.35721 2.263617e-06
YJL136C      0.7517022  21.60136 3.356135e-06
```

```
> ### How the UPS1 proteins get ordered in the list
> grep("HUMAN",rownames(pois.res[o,]))
```

```
 [1]   1   2   3   5   6   7   8  10  12  13  14  15  16  17  18  19  21  22  23
[20]  29  30  31  32  33  36  37  39  44  45  46  57  62  64  87  88 115 126 236
[39] 240
```

```
> ###  Truth table
> nh <- length(grep("HUMAN",featureNames(e)))
> ny <- length(grep("HUMAN",featureNames(e),invert=TRUE))
> tp <- length(grep("HUMAN",rownames(pois.res)[adjp<=0.01]))
> fp <- sum(adjp<=0.01)-tp
> (tt.pois1 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))
```

```
  TP FP  TN FN
1 19  7 618 20
```

# 5 Quasi-likelihood GLM regression example

The quasi-likelihood is a distribution free model that allows for overdispersion, and could be indicated where an appreciable source of biological variability is expected. In this model, instead of specifying a probability distribution for the data, we just provide a relationship between mean and variance. This relationship takes the form of a function, with a multiplicative factor known as the overdispersion, which has to be estimated from the data [11]. Its use in proteomics has been documented by Li et al. (2010)[13].

```
> ### Quasi-likelihood GLM
> ql.res <- msms.glm.qlll(e,alt.f,null.f,div=div)
> str(ql.res)

'data.frame':        664 obs. of  3 variables:
 $ LogFC  : num  0.12061 -0.18016 0.40229 0.00168 -0.37157 ...
 $ D      : num  5.22 8.93 4.11e+01 7.72e-04 3.10e+01 ...
 $ p.value: num  0.314295 0.001513 0.048548 0.991762 0.000234 ...

> ### DEPs on unadjusted p-values
> sum(ql.res$p.value<=0.01)

[1] 75

> ### DEPs on multitest adjusted p-values
> adjp <- p.adjust(ql.res$p.value,method="BH")
> sum(adjp<=0.01)

[1] 22

> ### The top features
> o <- order(ql.res$p.value)
> head(ql.res[o,],20)

                LogFC         D      p.value
PRDX1_HUMAN  1.6359291  26.61692 4.478445e-09
TRFE_HUMAN   1.3100859  81.53741 7.691259e-09
TRFL_HUMAN   1.8306054 150.41594 2.767215e-08
ALBU_HUMAN   1.9625212 104.51026 4.782251e-08
CYC_HUMAN   32.2452234  24.09871 6.946465e-08
CATA_HUMAN   1.5343714  50.12093 1.533212e-07
CAH2_HUMAN   2.3041307  63.81053 2.984566e-07
GSTA1_HUMAN  2.5495822  27.11728 5.778069e-07
CAH1_HUMAN   4.4542343  47.82775 3.020100e-06
UBE2C_HUMAN  3.0181352  25.68582 7.086311e-06
CATG_HUMAN   1.4749929  21.35193 7.428511e-06
CATD_HUMAN   1.6748334  24.58937 1.287976e-05
ANT3_HUMAN   1.4013022  27.39627 2.087493e-05
SODC_HUMAN   1.3712449  10.38483 2.928947e-05
MYG_HUMAN    2.1323062  46.76436 5.971058e-05
NQO1_HUMAN   3.7380273  25.66094 9.621658e-05
KCRM_HUMAN   1.4294782  37.83644 1.056421e-04
ANXA5_HUMAN  2.5156349  22.35721 2.173514e-04
YGR192C     -0.3715658  30.99581 2.338676e-04
TAU_HUMAN    1.5156349  11.43236 2.481070e-04

> ### How the UPS1 proteins get ordered in the list
> grep("HUMAN",rownames(ql.res[o,]))
```

```
 [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  20
[20]  23  24  27  31  32  34  35  42  44  47  54  58  60  61 122 153 158 248 249
[39] 281
```

```
> ###  Truth table
> tp <- length(grep("HUMAN",rownames(ql.res)[adjp<=0.01]))
> fp <- sum(adjp<=0.01)-tp
> (tt.ql1 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))
```

```
  TP FP  TN FN
1 19  3 622 20
```

# 6 edgeR: negative binomial GLM regression example

The negative-binomial provides another model that allows for overdispersion. The implementation adopted in this package is entirely based in the solution provided by the package edgeR [12] which includes empirical Bayes methods to share information among features, and thus may be employed even when the number of replicates is as low as two. The negative-binomial is downward limited, when no overdispersion is observed, by the Poisson distribution.

```
> ### Negative-binomial
> nb.res <- msms.edgeR(e,alt.f,null.f,div=div,fnm="treat")
> str(nb.res)
```

```
'data.frame':        664 obs. of  3 variables:
 $ LogFC  : num  0.1279 -0.1801 0.4264 0.0122 -0.3724 ...
 $ LR     : num  1.31411 8.79606 6.94419 0.00671 27.21513 ...
 $ p.value: num  2.52e-01 3.02e-03 8.41e-03 9.35e-01 1.82e-07 ...
```

```
> ### DEPs on unadjusted p-values
> sum(nb.res$p.value<=0.01)
```

```
[1] 40
```

```
> ### DEPs on multitest adjusted p-values
> adjp <- p.adjust(nb.res$p.value,method="BH")
> sum(adjp<=0.01)
```

```
[1] 22
```

```
> ### The top features
> o <- order(nb.res$p.value)
> head(nb.res[o,],20)
```

```
                LogFC        LR      p.value
TRFL_HUMAN   1.8228068 149.96454 1.764860e-34
ALBU_HUMAN   1.9486266 104.31880 1.722466e-24
TRFE_HUMAN   1.3049775  81.29444 1.944730e-19
CAH2_HUMAN   2.2665908  63.75511 1.408876e-15
CATA_HUMAN   1.5209383  50.05210 1.497178e-12
CAH1_HUMAN   4.0281168  47.49330 5.519227e-12
MYG_HUMAN    2.0923677  45.65439 1.410699e-11
KCRM_HUMAN   1.4146192  36.29769 1.693646e-09
ANT3_HUMAN   1.3827898  27.30596 1.736750e-07
YGR192C     -0.3723619  27.21513 1.820290e-07
GSTA1_HUMAN  2.4316185  26.92061 2.119859e-07
PRDX1_HUMAN  1.6054234  26.57310 2.537524e-07
UBE2C_HUMAN  2.8079615  25.49229 4.441551e-07
NQO1_HUMAN   3.3269851  25.47243 4.487505e-07
CATD_HUMAN   1.6393037  24.37063 7.947113e-07
CYC_HUMAN    4.6109517  24.00359 9.615641e-07
ANXA5_HUMAN  2.3800320  22.17538 2.488447e-06
CATG_HUMAN   1.4473996  21.18451 4.171211e-06
SYH_HUMAN    3.1111352  20.73578 5.272154e-06
IL8_HUMAN    3.2058569  15.17682 9.789756e-05

> ### How the UPS1 proteins get ordered in the list
> grep("HUMAN",rownames(nb.res[o,]))

 [1]   1   2   3   4   5   6   7   8   9  11  12  13  14  15  16  17  18  19  20
[20]  23  25  26  27  28  31  32  34  40  41  42  53  58  60  81  82 111 123 229
[39] 230

> ###  Truth table
> tp <- length(grep("HUMAN",rownames(nb.res)[adjp<=0.01]))
> fp <- sum(adjp<=0.01)-tp
> (tt.nb1 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))

  TP FP  TN FN
1 19  3 622 20
```

# 7 Reproducibility

In the *omics* field, reproducibility is of biggest concern. Very low p-values for a protein in an experiment are not enough to declare that protein as of interest as a biomarker. A good biomarker should give as well a reproducible signal, and posses a biologically significant effect size. According to our experience, a protein giving less than three counts in the most abundant condition results of poor reproducibility, to be declared as statistically significant, experiment after experiment. On the other hand most of the false positives in an spiking experiment show log fold changes below 1. These two observations [6] allow

to improve the results obtained in the previous sections. The trick is to flag as relevant those proteins which have low p-values, high enough signal, and good effect size. In performing this relevance filter we may even accept higher adjusted p-values than usual. This flagging is provided by the function `test.results`.

```
> ###  Cut-off values for a relevant protein as biomarker
> alpha.cut <- 0.05
> SpC.cut <- 2
> lFC.cut <- 1
> ###  Relevant proteins according to the Poisson GLM
> pois.tbl <- test.results(pois.res,e,pData(e)$treat,"U600","U200",div,
                           alpha=alpha.cut,minSpC=SpC.cut,minLFC=lFC.cut,
                           method="BH")$tres
> (pois.nms <- rownames(pois.tbl)[pois.tbl$DEP])

 [1] "TRFL_HUMAN"  "ALBU_HUMAN"  "TRFE_HUMAN"  "CAH2_HUMAN"  "CATA_HUMAN"
 [6] "CAH1_HUMAN"  "MYG_HUMAN"   "KCRM_HUMAN"  "ANT3_HUMAN"  "GSTA1_HUMAN"
[11] "PRDX1_HUMAN" "UBE2C_HUMAN" "NQO1_HUMAN"  "CATD_HUMAN"  "CYC_HUMAN"
[16] "ANXA5_HUMAN" "CATG_HUMAN"  "SYH_HUMAN"   "IL8_HUMAN"   "LYSC_HUMAN"
[21] "PDGFB_HUMAN" "TAU_HUMAN"   "SODC_HUMAN"  "YLR388W"     "LALBA_HUMAN"

> ###  Truth table
> ridx <- grep("HUMAN",pois.nms)
> tp <- length(ridx)
> fp <- length(pois.nms)-length(ridx)
> (tt.pois2 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))

  TP FP  TN FN
1 24  1 624 15

> ###  Relevant proteins according to the quasi-likelihood GLM
> ql.tbl <- test.results(ql.res,e,pData(e)$treat,"U600","U200",div,
                       alpha=0.05,minSpC=2,minLFC=1,method="BH")$tres
> (ql.nms <- rownames(ql.tbl)[ql.tbl$DEP])

 [1] "PRDX1_HUMAN" "TRFE_HUMAN"  "TRFL_HUMAN"  "ALBU_HUMAN"  "CYC_HUMAN"
 [6] "CATA_HUMAN"  "CAH2_HUMAN"  "GSTA1_HUMAN" "CAH1_HUMAN"  "UBE2C_HUMAN"
[11] "CATG_HUMAN"  "CATD_HUMAN"  "ANT3_HUMAN"  "SODC_HUMAN"  "MYG_HUMAN"
[16] "NQO1_HUMAN"  "KCRM_HUMAN"  "ANXA5_HUMAN" "TAU_HUMAN"   "SYH_HUMAN"
[21] "UB2E1_HUMAN" "IL8_HUMAN"   "LALBA_HUMAN" "UBC9_HUMAN"  "YOR098C"
[26] "LYSC_HUMAN"  "EGF_HUMAN"   "PDGFB_HUMAN" "YDR012W"     "RASH_HUMAN"

> ###  Truth table
> ridx <- grep("HUMAN",ql.nms)
> tp <- length(ridx)
> fp <- length(ql.nms)-length(ridx)
> (tt.ql2 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))
```

```
  TP FP  TN FN
1 28  2 623 11

> ###  Relevant proteins according to the negative-binomial GLM
> nb.tbl <- test.results(nb.res,e,pData(e)$treat,"U600","U200",div,
                     alpha=0.05,minSpC=2,minLFC=1,method="BH")$tres
> (nb.nms <- rownames(nb.tbl)[nb.tbl$DEP])

 [1] "TRFL_HUMAN"  "ALBU_HUMAN"  "TRFE_HUMAN"  "CAH2_HUMAN"  "CATA_HUMAN"
 [6] "CAH1_HUMAN"  "MYG_HUMAN"   "KCRM_HUMAN"  "ANT3_HUMAN"  "GSTA1_HUMAN"
[11] "PRDX1_HUMAN" "UBE2C_HUMAN" "NQO1_HUMAN"  "CATD_HUMAN"  "CYC_HUMAN"
[16] "ANXA5_HUMAN" "CATG_HUMAN"  "SYH_HUMAN"   "IL8_HUMAN"   "LYSC_HUMAN"
[21] "PDGFB_HUMAN" "TAU_HUMAN"   "SODC_HUMAN"  "YLR388W"

> ###  Truth table
> ridx <- grep("HUMAN",nb.nms)
> tp <- length(ridx)
> fp <- length(nb.nms)-length(ridx)
> (tt.nb2 <- data.frame(TP=tp,FP=fp,TN=ny-fp,FN=nh-tp))

  TP FP  TN FN
1 23  1 624 16
```

As you may see, without the post-test filter a relatively low adjusted p-value cut-off is required to keep an acceptable number of false positives. The post-test filter allows to relax the p-value cut-off improving at the same time both the number of true positives and false positives.

| Test | Significance | Filtered | TP | FP | TN | FN |
|---|---|---|---|---|---|---|
| Poisson | 0.05 | No | 26 | 12 | 613 | 13 |
| Quasi-likelihood | 0.05 | No | 30 | 24 | 601 | 9 |
| Negative-binomial | 0.05 | No | 24 | 6 | 619 | 15 |
| Poisson | 0.01 | No | 19 | 7 | 618 | 20 |
| Quasi-likelihood | 0.01 | No | 19 | 3 | 622 | 20 |
| Negative-binomial | 0.01 | No | 19 | 3 | 622 | 20 |
| Poisson | 0.05 | Yes | 24 | 1 | 624 | 15 |
| Quasi-likelihood | 0.05 | Yes | 28 | 2 | 623 | 11 |
| Negative-binomial | 0.05 | Yes | 23 | 1 | 624 | 16 |

Table 1: Truth tables

# 8 Other functions in the package

A useful tool to visualize the global results of differential expression tests is a table of accumulated frequencies of features by p-values in bins of log fold changes. It may help in finding the most appropriate post-test filter cut-off values in a given experiment.

```
> ### All features
> pval.by.fc(ql.tbl$adjp,ql.tbl$LogFC)

                 p.vals
LogFC             <=0.001 <=0.005 <=0.01 <=0.05 <=0.1 <=0.2 <=1
  (-Inf,-10]            0       0      2      6    13    21  40
  (-10,-2]              0       0      0      0     2     5  14
  (-2,-1]               0       0      0      2     4     8  41
  (-1,-0.848]           0       0      0      1     1     2   7
  (-0.848,-0.585]       0       0      0      4     6    15  52
  (-0.585,0]            0       0      1     10    16    36 282
  (0,0.585]             0       0      0      0     0     0 123
  (0.585,0.848]         0       0      0      0     2     3  16
  (0.848,1]             0       0      0      0     0     1  15
  (1,2]                 7      10     11     16    17    22  30
  (2,10]                4       6      7     11    12    13  17
  (10, Inf]             1       1      1      4     8    17  27
  Tot                  12      17     22     54    81   143 664

> ### Filtering by minimal signal
> fl <- ql.tbl$U600 > 2
> pval.by.fc(ql.tbl$adjp[fl],ql.tbl$LogFC[fl])

                 p.vals
LogFC             <=0.001 <=0.005 <=0.01 <=0.05 <=0.1 <=0.2 <=1
  (-Inf,-10]            0       0      0      0     0     0   0
  (-10,-2]              0       0      0      0     0     0   0
  (-2,-1]               0       0      0      0     0     0   0
  (-1,-0.848]           0       0      0      1     1     2   3
  (-0.848,-0.585]       0       0      0      3     4    10  13
  (-0.585,0]            0       0      1     10    16    36 195
  (0,0.585]             0       0      0      0     0     0  85
  (0.585,0.848]         0       0      0      0     2     3   7
  (0.848,1]             0       0      0      0     0     1   2
  (1,2]                 7      10     11     16    17    20  21
  (2,10]                4       6      7     11    12    12  12
  (10, Inf]             1       1      1      1     1     1   1
  Tot                  12      17     20     42    53    85 339
```

Another usual tool is a volcanoplot with the ability to visualize the effect of different post-test filter cut-off values.

```
> par(mar=c(5,4,0.5,2)+0.1)
> res.volcanoplot(ql.tbl,max.pval=0.05,min.LFC=1,maxx=3,maxy=NULL,
                                    ylbls=3)
```

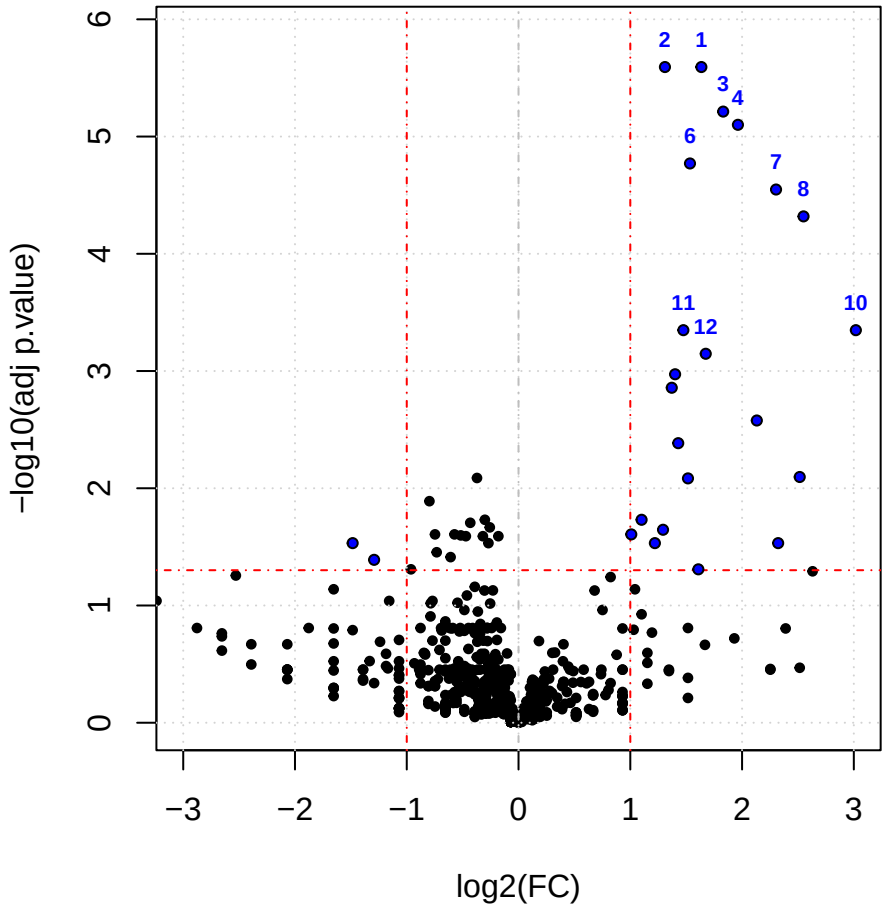


Figure 1: Volcanoplot

# References


[1] Shi, L. et al. *MAQC Consortium: The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models.* Nat Biotech 2010, 28, 827-838.

[2] Shi, L. et al. *The balance of reproducibility, sensitivity, and specificity of lists of differentially expressed genes in microarray studies.* BMC Bioinformatics 2008, 9 Suppl 9, S10.

[3] Luo, J. et al. *A comparison of batch effect removal methods for enhancement of prediction performance using MAQC-II microarray gene expression data.* The Pharmacogenomics Journal 2010, 10, 278-291.

[4] Mallick P., Kuster B. *Proteomics: a pragmatic perspective.* Nat Biotechnol 2010;28:695-709.

[5] Neilson K.A., Ali N.A., Muralidharan S., Mirzaei M., Mariani M., Assadourian G., et al. *Less label, more free: approaches in label-free quantitative mass spectrometry.* Proteomics 2011;11:535-53.

[6] Gregori J., Villareal L., Sanchez A., Baselga J., Villanueva J., *An Effect Size Filter Improves the Reproducibility in Spectral Counting-based Comparative Proteomics.* Journal of Proteomics 2013, http://dx.doi.org/10.1016/j.jprot.2013.05.030

[7] Laurent Gatto and Kathryn S. Lilley, MSnbase - an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation, Bioinformatics 28(2), 288-289 (2012).

[8] Chambers J.M. *Software for data analysis: programming with R*, 2008 Springer

[9] Genolini C. *A (Not So) Short Introduction to S4* (2008)

[10] Falcon S., Morgan M., Gentleman R. *An Introduction to Bioconductor's ExpressionSet Class* (2007)

[11] Agresti A., *Categorical Data Analysis*, Wiley-Interscience, Hoboken NJ, 2002

[12] Robinson MD, McCarthy DJ and Smyth GK (2010). *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.* Bioinformatics 26, 139-140

[13] Li, M.; Gray, W.; Zhang, H.; Chung, C. H.; Billheimer, D.; Yarbrough, W. G.; Liebler, D. C.; Shyr, Y.; Slebos, R. J. C. *Comparative shotgun proteomics using spectral count data and quasi-likelihood modeling.* J Proteome Res 2010, 9, 4295-4305

[14] Gregori J., Villareal L., Mendez O., Sanchez A., Baselga J., Villanueva J., *Batch effects correction improves the sensitivity of significance tests in spectral counting-based comparative discovery proteomics*, Journal of Proteomics, 2012, 75, 3938-3951

[15] Benjamini, Y., and Hochberg, Y. (1995). *Controlling the false discovery rate: a practical and powerful approach to multiple testing.* Journal of the Royal Statistical Society Series B, 57, 289-300.