Statistical analysis of proteomics experiments with R and MSstats (v3.0.9)

Meena Choi, Tsung-Heng Tsai & Erik Verschueren

September 10, 2015

Prerequisites

Setting the working directory to where you saved files.

```
setwd('/Users/Meena/Dropbox/MSstats_GitHub_document')
getwd()
```

```
## [1] "/Users/Meena/Dropbox/MSstats_GitHub_document"
```

If you didn't have MSstats installed so far. Please install it now.

```
install.packages(pkgs = 'MSstats_3.0.9.tar.gz', repos = NULL, type = 'source')
```

Load MSstats and verify that you have the correct version (3.0.9) loaded.

```
library('MSstats', warn.conflicts = F, quietly = T, verbose = F)
?MSstats
```

DDA analysis with MSstats

Controlled mixture DDA data will be used for demonstration. This dataset is available in MSstats material github in the folder named 'example dataset/DDA_controlledMixture2009'. It is processed by Superhirn. original reference link

1. preparing the data for MSstats input

Let's start by reading in data.

```
DDA2009.superhirn<-read.csv("RawData.DDA.csv")

head(DDA2009.superhirn)
```

```
##   ProteinName PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      bovine     S.PVDIDTK_5              NA           5            NA
## 2      bovine     S.PVDIDTK_5              NA           5            NA
## 3      bovine     S.PVDIDTK_5              NA           5            NA
## 4      bovine     S.PVDIDTK_5              NA           5            NA
```

```
## 5      bovine     S.PVDIDTK_5              NA           5            NA
## 6      bovine     S.PVDIDTK_5              NA           5            NA
##   IsotopeLabelType Condition BioReplicate Run Intensity
## 1                L        C1            1   1   2636792
## 2                L        C1            1   2   1992418
## 3                L        C1            1   3   1982146
## 4                L        C2            1   4   5019594
## 5                L        C2            1   5   4560468
## 6                L        C2            1   6   3627849
```

2. Processing the data

2.1 Normalizing and summarizing data with dataProcess

```
?dataProcess
```

2.1.1 Default normalization and summarization options (! Always pay attention to the default options)

The default option for normalization is `equalizeMedians`, constant normalization shifts all the intensities in a run by a constant, to equalize the median of intensities across runs for label free experiment. This normalization works when we can assume the majority of proteins do not change across proteins. Therefore if your label free DDA dataset has small number of proteins, please be careful to use `equalizeMedians` Instead, if you have a spiked in standard then you want to set this to `globalStandards` and define the standard with `nameStandards` For label based experiment, `equalizeMedians` equalizes the median of reference intensities across runs for label based experiment. therefore, it works for dataset with small number of proteins.

Usually label-free DDA datasets have many missing values and noisy features with outliers. MSstats supports several ways to deal with this. The default option for summarization is `TMP` (Tukey's median polish, robust parameter estimation method with median across rows and columns) after imputation by AFT (accelerated failure time model, `MBimpute=TRUE`) based on censored intensity for NA (`censoredInt="NA"`, which assumes that NA intensities is censored, not detected due to below the cutoff) with cutoff value for censored = minimum of minimum value for a feature and minimum value for a run (`cutoffCensored="minFeatureNRun"`). It can handle missing values by imputing and outliers from TMP estimation. However, the runs, which have no measurements at all, will be removed and not be used for any calculation.

```
# default option
DDA2009.TMP <- dataProcess(raw = DDA2009.superhirn,  fillIncompleteRows = TRUE,
                           normalization = 'equalizeMedians',
                           summaryMethod = 'TMP',
                           censoredInt="NA", cutoffCensored="minFeatureNRun",
                           MBimpute=TRUE)
```

```
## * Use all features that the dataset origianally has.
```

```
##
##   Summary of Features :
##                            count
## # of Protein                   6
## # of Peptides/Protein    11-32
## # of Transitions/Peptide   1-1
##
##   Summary of Samples :
```

```
##                            C1 C2 C3 C4 C5 C6
## # of MS runs                3  3  3  3  3  3
## # of Biological Replicates  1  1  1  1  1  1
## # of Technical Replicates   3  3  3  3  3  3
```

```
##
##  Summary of Missingness :
##
##   # transitions are completely missing in one condition: 90
##
##     -> D.GPLTGTYR_23_NA_23_NA, F.HFHWGSSDDQGSEHTVDR_402_NA_402_NA, G.PLTGTYR_8_NA_8_NA, H.SFNVEYDDSQ
##
##   # run with 75% missing observations: 0
##
##
##  == Start the summarization per subplot...
## Getting the summarization by Tukey's median polish per subplot for protein  bovine ( 1  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  chicken ( 2  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  cyc_horse ( 3  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  myg_horse ( 4  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  rabbit ( 5  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  yeast ( 6  of  6 )
##
##  == the summarization per subplot is done.
```

If 'MBimpute=FALSE', imputation based on AFT model will be not performed and censored intensities (here NAs) will be just replace with cutoff value assigned in `cutoffCensored="minFeatureNRun"`. Then the runs with more than 50% measurement missing will use cutoff value, which is biased by cutoff value. In this case, you can remove the runs ths have more than 50 % missing values in analysis with `remove50missing=TRUE`.

```
names(DDA2009.TMP)
```

```
## [1] "ProcessedData"     "RunlevelData"      "SummaryMethod"
## [4] "ModelQC"           "PredictBySurvival"
```

```
# the data after reformatting and normalization
head(DDA2009.TMP$ProcessedData)
```

```
##      PROTEIN                      PEPTIDE TRANSITION
## 55    bovine            D.GPLTGTYR_23_NA      23_NA
## 937   bovine F.HFHWGSSDDQGSEHTVDR_402_NA     402_NA
## 1628  bovine   F.HWGSSDDQGSEHTVDR_229_NA     229_NA
## 19    bovine              G.PLTGTYR_8_NA       8_NA
## 1081  bovine       H.SFNVEYDDSQDK_465_NA     465_NA
## 469   bovine        K.AVVQDPALKPL_156_NA     156_NA
##                                FEATURE LABEL GROUP_ORIGINAL
## 55                D.GPLTGTYR_23_NA_23_NA     L             C1
## 937  F.HFHWGSSDDQGSEHTVDR_402_NA_402_NA     L             C1
## 1628   F.HWGSSDDQGSEHTVDR_229_NA_229_NA     L             C1
## 19                   G.PLTGTYR_8_NA_8_NA     L             C1
## 1081       H.SFNVEYDDSQDK_465_NA_465_NA     L             C1
## 469         K.AVVQDPALKPL_156_NA_156_NA     L             C1
```

```
##      SUBJECT_ORIGINAL RUN GROUP SUBJECT SUBJECT_NESTED INTENSITY ABUNDANCE
## 55                  1   1     1       1            1.1  757400.1  19.79517
## 937                 1   1     1       1            1.1 2087125.8  21.25756
## 1628                1   1     1       1            1.1 1485145.8  20.76665
## 19                  1   1     1       1            1.1 4986404.0  22.51404
## 1081                1   1     1       1            1.1 2488141.2  21.51111
## 469                 1   1     1       1            1.1 7519322.0  23.10664
##      METHOD
## 55        1
## 937       1
## 1628      1
## 19        1
## 1081      1
## 469       1
```

```
 # run-level summarized data
head(DDA2009.TMP$RunlevelData)
```

```
##   RUN   Protein LogIntensities more50missing GROUP GROUP_ORIGINAL
## 1   1    bovine       21.25800         FALSE     1             C1
## 2   1 myg_horse       22.77386         FALSE     1             C1
## 3   1    rabbit       12.54786          TRUE     1             C1
## 4   1     yeast       16.44018          TRUE     1             C1
## 5   1   chicken       18.29509         FALSE     1             C1
## 6   1 cyc_horse       19.49460         FALSE     1             C1
##   SUBJECT_ORIGINAL SUBJECT_NESTED SUBJECT
## 1                1            1.1       1
## 2                1            1.1       1
## 3                1            1.1       1
## 4                1            1.1       1
## 5                1            1.1       1
## 6                1            1.1       1
```

```
# Since this is not model-based, no model summary (here DDAskyline.quant$ModelQC=NULL).
# Only with 'summaryMethod="linear"'
head(DDA2009.TMP$ModelQC)
```

```
## NULL
```

```
# here 'TMP'
head(DDA2009.TMP$SummaryMethod)
```

```
## [1] "TMP"
```

```
# predict values by AFT with 'MBimpute=TRUE'.
# These values are matching with rownames of DDA2009.TMP$ProcessedData
head(DDA2009.TMP$PredictBySurvival)
```

```
##       55      937     1628       19     1081      469
## 19.63438 22.40369 21.67132 23.52286 21.55629 23.11901
```

If 'censoredInt=NULL', we assume that all intensities are randomly missing and there is no action for missing values.

```
# default option
DDA2009.TMP.random <- dataProcess(raw = DDA2009.superhirn,  fillIncompleteRows = TRUE,
                            normalization = 'equalizeMedians',
                            summaryMethod = 'TMP',
                            censoredInt=NULL)
```

```
## * Use all features that the dataset origianally has.

##
##   Summary of Features :
##                            count
## # of Protein                   6
## # of Peptides/Protein    11-32
## # of Transitions/Peptide   1-1
##
##   Summary of Samples :
##                              C1 C2 C3 C4 C5 C6
## # of MS runs                  3  3  3  3  3  3
## # of Biological Replicates   1  1  1  1  1  1
## # of Technical Replicates    3  3  3  3  3  3

##
##  Summary of Missingness :
##
##   # transitions are completely missing in one condition: 90
##
##     -> D.GPLTGTYR_23_NA_23_NA, F.HFHWGSSDDQGSEHTVDR_402_NA_402_NA, G.PLTGTYR_8_NA_8_NA, H.SFNVEYDDSQ
##
##   # run with 75% missing observations: 0
##
##
##  == Start the summarization per subplot...
## Getting the summarization by Tukey's median polish per subplot for protein  bovine ( 1  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  chicken ( 2  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  cyc_horse ( 3  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  myg_horse ( 4  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  rabbit ( 5  of  6 )
## Getting the summarization by Tukey's median polish per subplot for protein  yeast ( 6  of  6 )
##
##  == the summarization per subplot is done.
```

2.1.2 Different summarization options Besides summarizing observations with linear models MSstats also offers a summarization option using linear model, and as sum of log-intensities, which is the default Skyline behaviour. `summaryMethod="linear"` with `censoredInt=NULL` assumes that all NAs are randomly missing and use `lm` or `lmer` for parameter estimation. `summaryMethod="linear"` with `censoredInt="NA"` assumes that NA intensities are censored, not detected because intensities are below the cutoff. Then use AFT model with left-censored for parameter estimation. 'cutoffCensored' is the same as `summaryMethod="TMP"`. For `summaryMethod="linear"`, AFT model is only used for parameter estimation, not for imputation. Therefore `MBimpute=TRUE` with `summaryMethod="linear"` doesn't work (no imputation before parameter estimation, the same with `MBimpute=FALSE`).

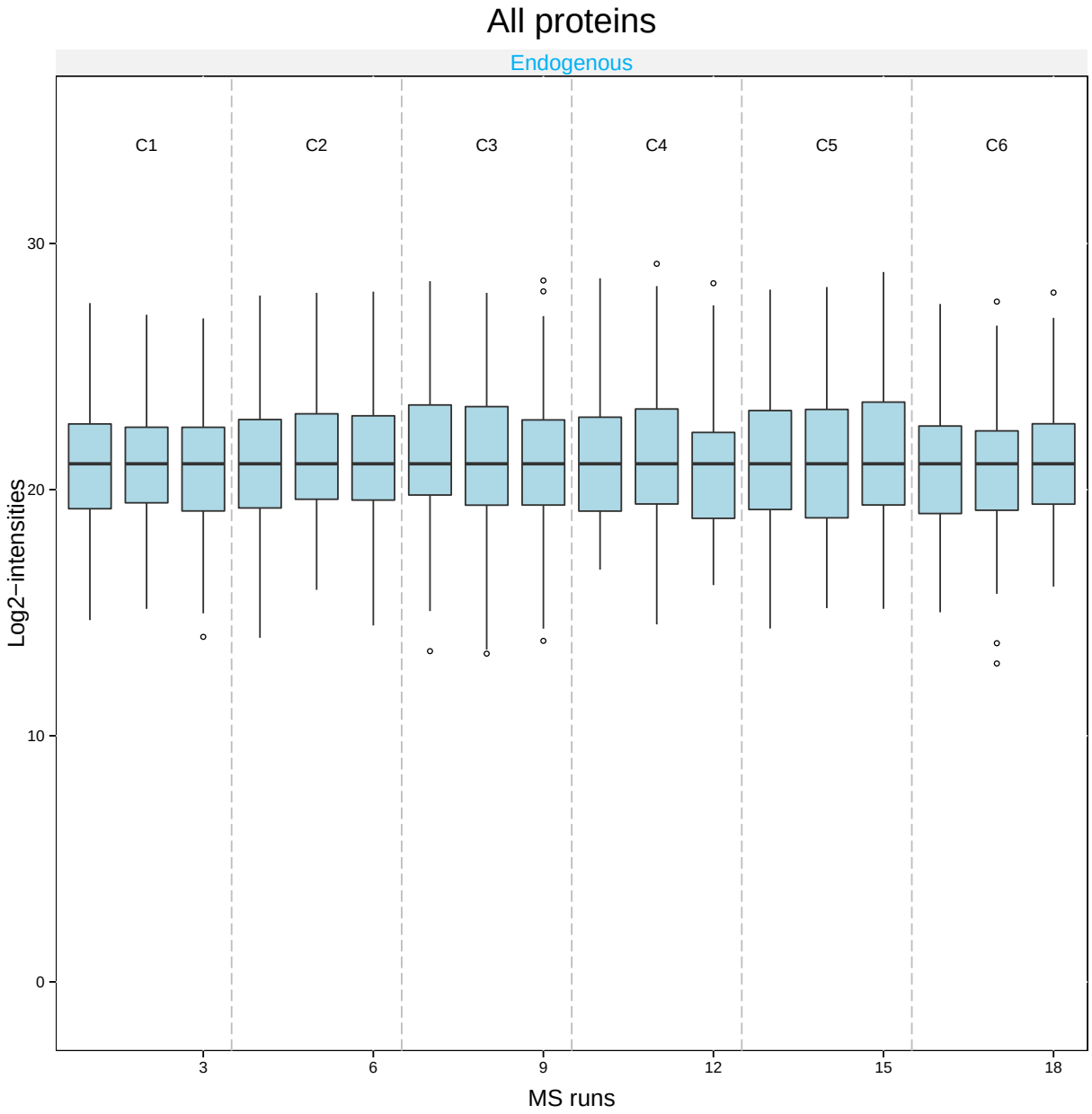
```
# linear mixed model (lm or lmer) with run and feature
DDA2009.linear <- dataProcess(raw = DDA2009.superhirn,
                              summaryMethod="linear", censoredInt=NULL)

# accerated failure model with left-censored. NA intensities are assumed as censored
DDA2009.linear.censored <- dataProcess(raw = DDA2009.superhirn,
                                       summaryMethod="linear", censoredInt="NA")
```

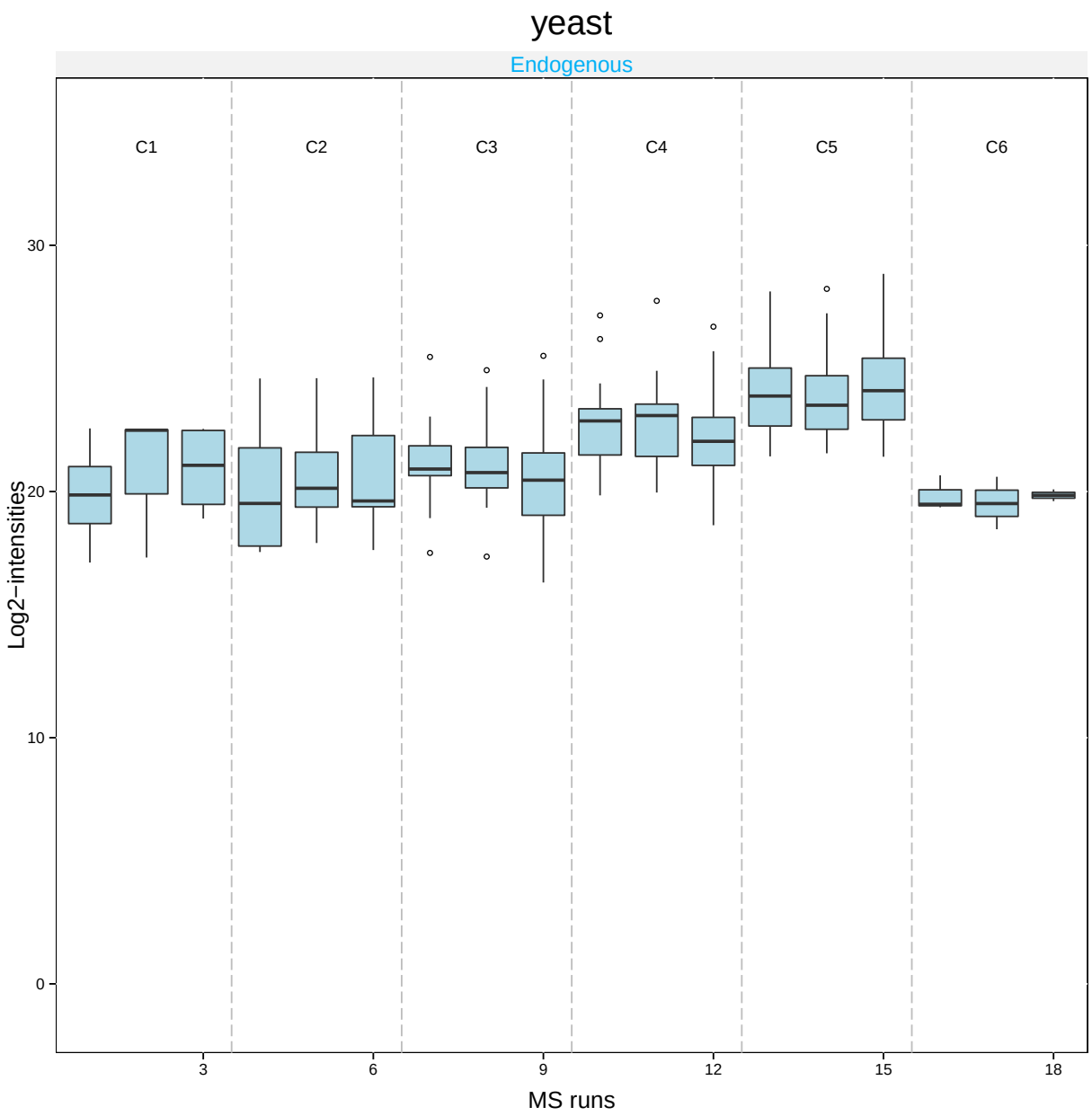
2.2 Visualization of processed data

2.2.1 Quality control and Normalization effects Now let's look at what the equalize medians procedure did to our data. We can generate these for all proteins but also for single Proteins at a time if we have a big dataset. Let's look at the 'Kininogen-1' (Kng1/NP_036828) protein in these example data.

```
# use type="QCplot" with all proteins and change the upper limit of y-axis=35
dataProcessPlots(data = DDA2009.TMP, type="QCplot", ylimUp=35)
```

```
# QCplot for only 'yeast' protein
dataProcessPlots(data = DDA2009.TMP, type="QCplot", ylimUp=35,
                 which.Protein="yeast", address="yeast_eqmedians_")
```


All proteins
Endogenous
C1
C2
C3
C4
C5
C6
30
20
10
0
Log2−intensities
3
6
9
12
15
18
MS runs



NOTE Don't worry about warning messages as below. It means they will not plot with NA values, which we want.

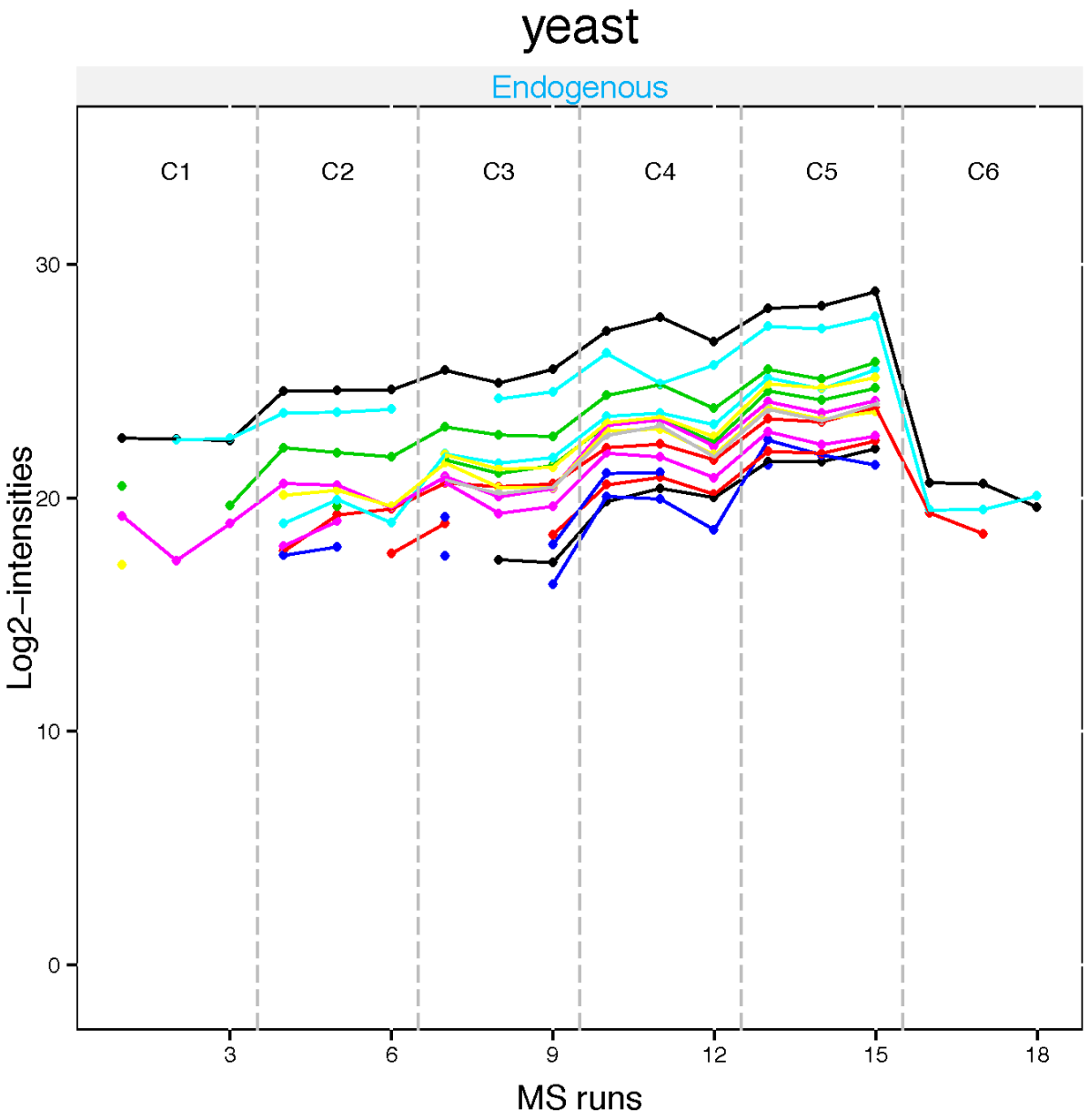
Warning messages: 1: In loop_apply(n, do.ply) : Removed 698 rows containing non-finite values (stat_boxplot).

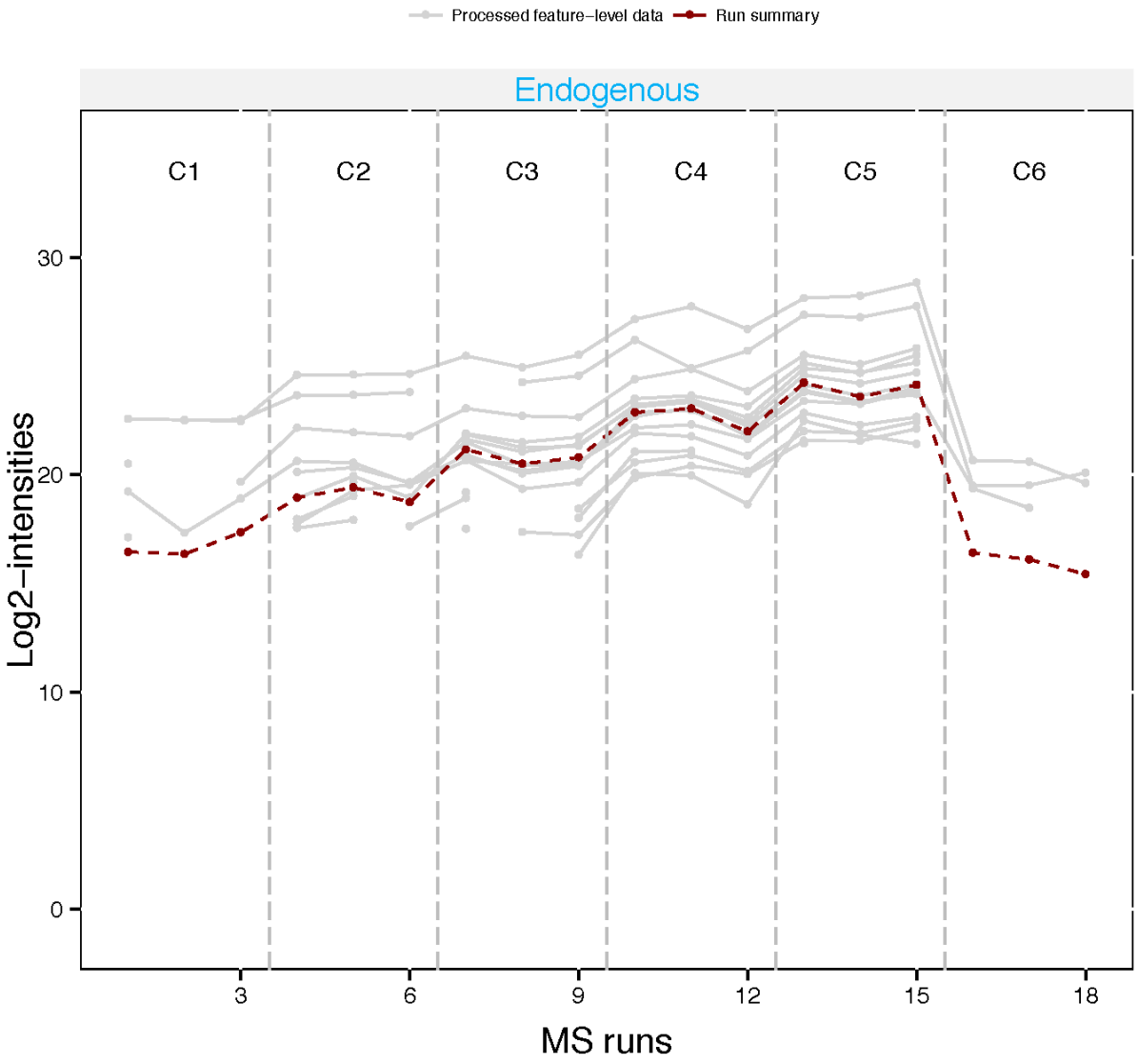
2.2.2 Summarization effects

Profile plots Let's look at how the original data look like. Profile plot show all individual measurements per protein. With type="profileplot", two pdfs will be generated. The first pdf includes plots (per protein) to show individual measurement for each peptide (peptide for DDA, transition for SRM or DIA) across runs, grouped per condition. Ech peptide has a different colour/type layout. If you dont want to generate these plots, please use 'originalPlot=FALSE'. The second pdf, which named with 'wSummarization' suffix, show

run-level summarized data per protein. The same peptides (or transition) in the first plot are presented in grey, with the summarized values by TMP overlayed in red.

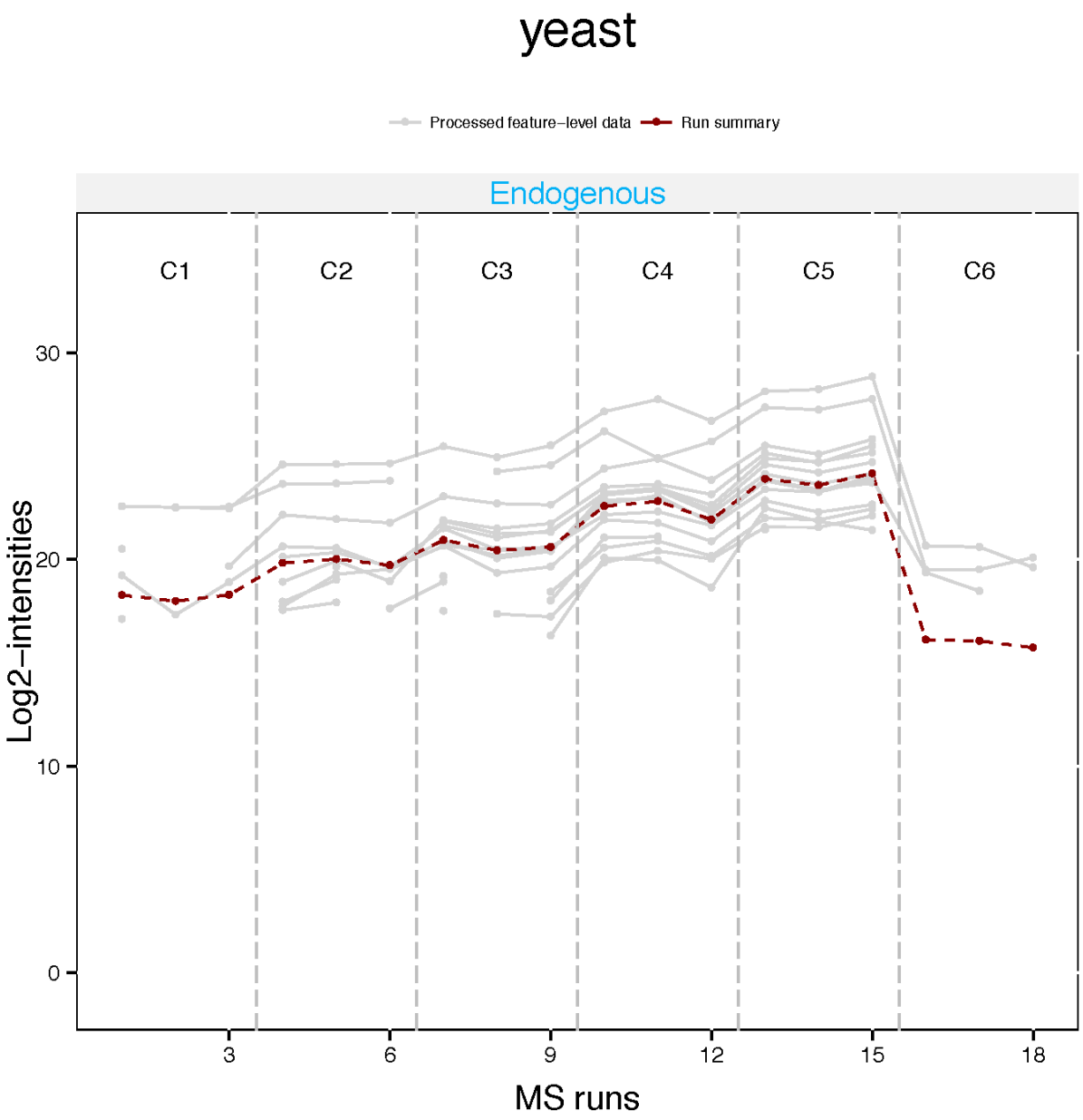
```
dataProcessPlots(data = DDA2009.TMP, type="Profileplot",  ylimUp=35,
                 featureName="NA", width=7, height=7, address="DDA2009_TMP_")
```





Have a look at the profile plots to compare the different options (TMP with considering missing values vs. TMP without considering missing values for summarizing in dataProcess steps.) Original profile plots are the same. But, summarization plots are different, especially for condition 'C1' and 'C2' in 'yeast' protein, which have many missing values.

```
dataProcessPlots(data = DDA2009.TMP.random, type="Profileplot",  ylimUp=35,
                 featureName="NA", width=7, height=7,
                 originalPlot=FALSE, summaryPlot=TRUE, address="DDA2009_TMP_random_")
```



3. Finding differentially abundant proteins across conditions

3.1 Comparing conditions with groupComparison

After we normalized the data and summarized each protein's behaviour across conditions with one of the dataProcess summarization methods, we are all set to compare protein changes between groups of conditions. Within MSstats we can do this with the `groupComparison` function, which takes as input the output of the `dataProcess` function.

```
?groupComparison
```

Of course we have to tell `groupComparison` which are the conditions we would like to compare...
We define our contrast matrix by adding a column for every condition, **in alphabetical order**.

```
levels(DDA2009.TMP$ProcessedData$GROUP_ORIGINAL)
```

```
## [1] "C1" "C2" "C3" "C4" "C5" "C6"
```

It shows the order of condition, which MSstats can recognize. Then, We add a row for every comparison we would like to make between groups of conditions.

0 is for conditions we would like to ignore. **1** is for conditions we would like to put in the numerator of the ratio or fold-change. **-1** is for conditions we would like to put in the denumerator of the ratio or fold-change.

For example, if you want to compare C2-C1, which means log(C2)-log(C1) and the same as log(C2/C1), set '1' for C2 and '-1' for C1 in the row. Also combination of multiple groups are possible. For example, if you want to compare between average of C2 and C3 and average of C1, (C3+C2)/2-C1 as formula, set '-1' for C1, '0.5' for C2 and '0.5' for C3, and '0' for rest of groups.

```
comparison1<-matrix(c(-1,1,0,0,0,0),nrow=1)
comparison2<-matrix(c(0,-1,1,0,0,0),nrow=1)
comparison3<-matrix(c(0,0,-1,1,0,0),nrow=1)
comparison4<-matrix(c(0,0,0,-1,1,0),nrow=1)
comparison5<-matrix(c(0,0,0,0,-1,1),nrow=1)
comparison6<-matrix(c(1,0,0,0,0,-1),nrow=1)

comparison<-rbind(comparison1,comparison2,comparison3,comparison4,comparison5,comparison6)
row.names(comparison)<-c("C2-C1","C3-C2","C4-C3","C5-C4","C6-C5","C1-C6")
```

We're ready to go! let's compare our two populations.

```
DDA2009.comparisons <- groupComparison(contrast.matrix = comparison, data=DDA2009.TMP)
```

Let's inspect the results to see what proteins are changing significantly between groups.

```
# output from groupComparison function has three data.frame
names(DDA2009.comparisons)
```

```
## [1] "ComparisonResult" "ModelQC"          "fittedmodel"
```

```
# name of columns in result data.frame
names(DDA2009.comparisons$ComparisonResult)
```

```
## [1] "Protein"    "Label"      "log2FC"     "SE"         "Tvalue"
## [6] "DF"         "pvalue"     "adj.pvalue"
```

```
# get only siginificant proteins and comparions among all comparisons
SignificantProteins =
  DDA2009.comparisons$ComparisonResult[DDA2009.comparisons$ComparisonResult$adj.pvalue < 0.05 ,]
nrow(SignificantProteins)
```

```
## [1] 33
```

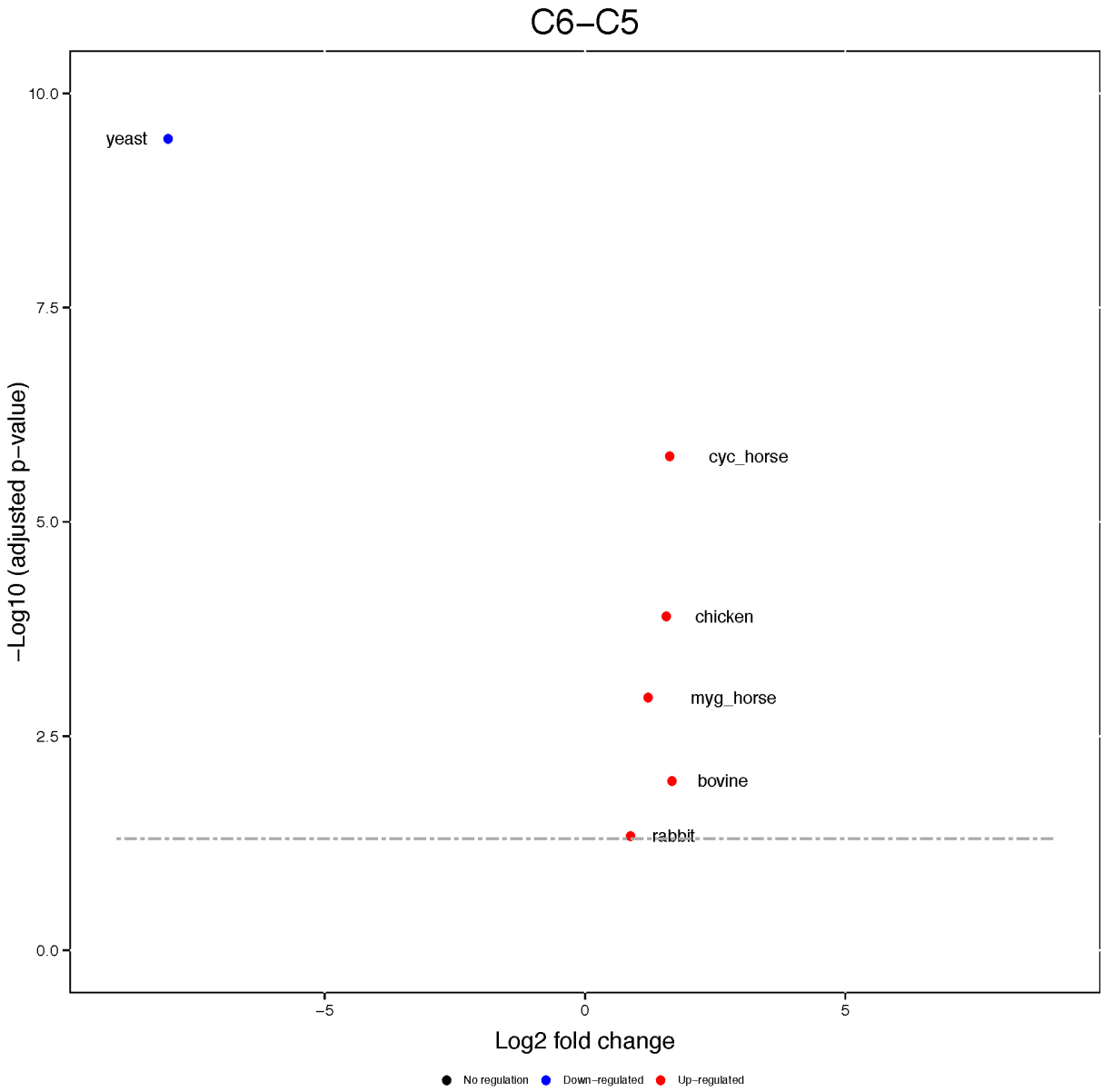
3.2 Visualization of differentially abundant proteins

```
?groupComparisonPlots
```

Volcano plots visualize the outcome of one comparison between conditions for all the pro- teins, and combines the information on statistical and practical significance. The y-axis displays the FDR-adjusted p-values on the negative log2 scale, and represents statistical significance. The horizontal dashed line represents the FDR cutoff. The points above the FDR cutoff line are statistically significant differentially abundant proteins. These points are colored in red for upregulated proteins, and in blue for downregulated proteins. The x-axis is the model-based estimate of log-fold change (the base of logarithm transform is the same as specified in the logTrans option of the dataProcess step), and represents practical significance. It is possible to specify a practical significance cutoff based on the estimate of fold change in addition to the statistical significance cutoff. If the fold change cutoff is specified, the points above the horizontal cutoff line but within the vertical cutoff line will be judged as not differentially abundant (and will be colored in black). The practical significance cutoff can only be applied in addition to the statistical significance cutoff (i.e. the fold change alone does not present enough evidence for differential abundance).

```
groupComparisonPlots(data = DDA2009.comparisons$ComparisonResult, type = 'VolcanoPlot')
```

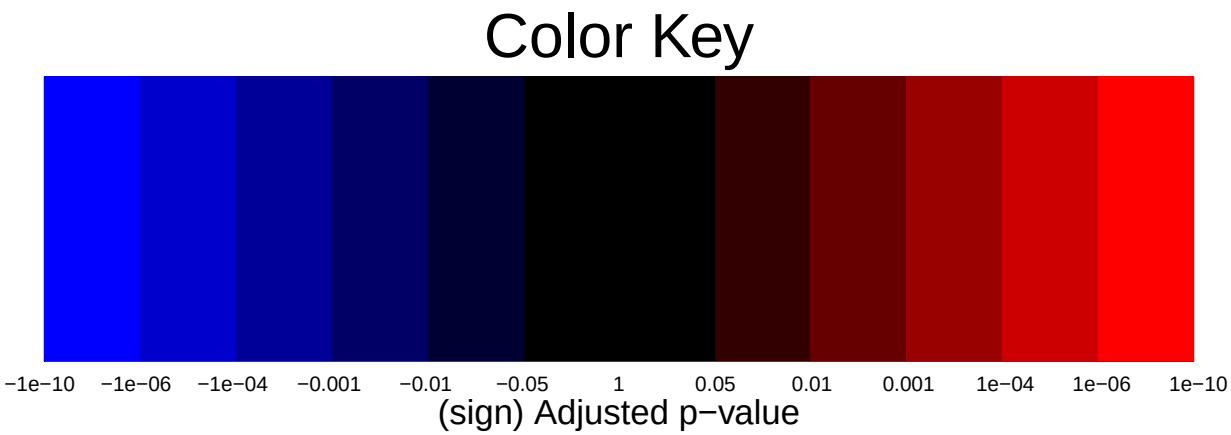
Then, 'VolcanoPlot.pdf' will be saved under the folder you assigned. It has the plots per comparison you set in contrast.matrix. Below is one of volcano plot, for comparison 'C6-C5'. Please check '?groupComparisonPlots' for detail, such as labelling protein names, size of dots, font sizes, etc.

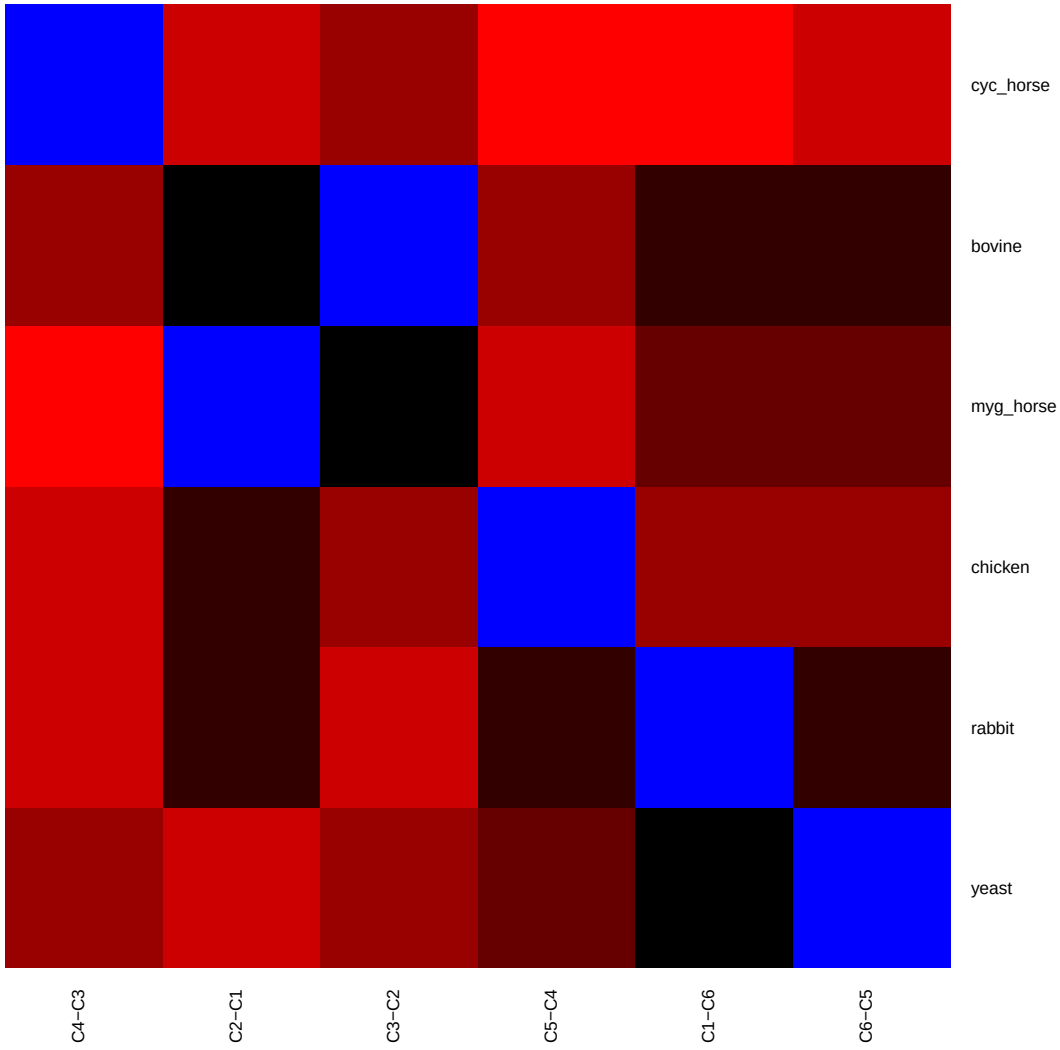


Heatmaps illustrate the patterns of up- and down-regulation of proteins in several comparisons. Columns in the heatmaps are comparison of conditions you assigned in contrast.matrix, and rows are proteins. The heatmaps display signed FDR-adjusted p-values of the tests, colored in red/blue for significantly up-/down-regulated proteins, while taking into account the specified FDR cutoff and the additional optional fold change cutoff. Brighter colors indicate stronger evidence in favor of differential abundance. Black color represents proteins are not significantly differentially abundant.

```
groupComparisonPlots(data = DDA2009.comparisons$ComparisonResult, type = 'Heatmap')
```

Then, 'VolcanoPlot.pdf' will be saved under the folder you assigned. It has the plots per comparison you set in contrast.matrix. Below is one of volcano plot, for comparison 'C6-C5'. Please check '?groupComparisonPlots' for detail, such as labelling protein names, size of dots, font sizes, etc.


Color Key
−1e−10
−1e−06
−1e−04
−0.001
−0.01
−0.05
1
0.05
0.01
0.001
1e−04
1e−06
1e−10
(sign) Adjusted p−value



Convert skyline output to MSstats required format

If you use Skyline output, please follow this section. The same original raw dataset from previous section is used, but it is processed by Skyline.

1. preparing the data for MSstats input

Let's read the data.

```
raw<-read.csv(file="ControlMixerMSstatsInputfromskyline.csv")
head(raw)
```

```
##   ProteinName  PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 2      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 3      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 4      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 5      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 6      bovine HWGSSDDQGSEHTVDR               2   precursor             2
##   IsotopeLabelType Condition BioReplicate         FileName    Area
## 1            light         1            1 B06-8004_c.mzXML 1015466
## 2            light         3            2 B06-8006_c.mzXML  907841
## 3            light         5            3 B06-8008_c.mzXML 1263905
## 4            light         2            4 B06-8010_c.mzXML 2457121
## 5            light         4            5 B06-8012_c.mzXML  958204
## 6            light         6            6 B06-8014_c.mzXML  788090
##   StandardType Truncated
## 1           NA     False
## 2           NA     False
## 3           NA     False
## 4           NA     False
## 5           NA     False
## 6           NA     False
```

The raw data (input data for MSstats) is required to contain variable of ProteinName, PeptideSequence, PrecursorCharge, FragmentIon, ProductCharge, IsotopeLabelType, Condition, BioReplicate, Run, Intensity. The variable names should be fixed. MSstats input from Skyline adapts the column scheme of the dataset so that it fits MSstats input format. However you should change `FileName` to `Run` and `Area` to `Intensity`.

```
colnames(raw)[9] <- 'Run'
colnames(raw)[10] <- 'Intensity'
head(raw)
```

```
##   ProteinName  PeptideSequence PrecursorCharge FragmentIon ProductCharge
## 1      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 2      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 3      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 4      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 5      bovine HWGSSDDQGSEHTVDR               2   precursor             2
## 6      bovine HWGSSDDQGSEHTVDR               2   precursor             2
##   IsotopeLabelType Condition BioReplicate             Run Intensity
## 1            light         1            1 B06-8004_c.mzXML   1015466
## 2            light         3            2 B06-8006_c.mzXML    907841
## 3            light         5            3 B06-8008_c.mzXML   1263905
## 4            light         2            4 B06-8010_c.mzXML   2457121
## 5            light         4            5 B06-8012_c.mzXML    958204
## 6            light         6            6 B06-8014_c.mzXML    788090
##   StandardType Truncated
## 1           NA     False
## 2           NA     False
## 3           NA     False
```

```
## 4           NA     False
## 5           NA     False
## 6           NA     False
```

The difference between output from Skyline and other spectral processing tool is that Skyline can distinguish random missing (NA) and censored missing (zero). The output from Skyline can have NA (expect small number of NAs or none of them) and 0(zero). Then, we can use zero values in intensity are censored.

```
sum(is.na(raw$Intensity))
```

```
## [1] 0
```

```
sum(raw$Intensity==0)
```

```
## [1] 4326
```

1.1 preprocessing with DDA experiment from Skyline output

The output from Skyline for DDA experiment needs one extra step before using MSstats. It has several peak area from the monoisotopic, M+1 and M+2 peaks. To get a robust measure of peptide intensity, we can sum per peptide or use highest peak per peptide. Here we will sum per peptide.

```
library(reshape2)

raw$pepprecursor<-paste(raw$PeptideSequence, raw$PrecursorCharge, sep="_")

data_w = dcast( Run ~ pepprecursor, data=raw, value.var='Intensity', fun.aggregate=sum, fill=NULL)

newdata = melt(data_w, id.vars=c('Run'))
colnames(newdata)[colnames(newdata) %in% c("variable","value")]<-c('pepprecursor','Intensity')

uniinfo<-unique(raw[,c("ProteinName","PeptideSequence","PrecursorCharge","pepprecursor")])
newraw<-merge(newdata,uniinfo, by="pepprecursor")

uniinfo<-unique(raw[,c("Run","BioReplicate","Condition")])
newraw<-merge(newraw,uniinfo, by="Run")

newraw$BioReplicate<-1 # it should be change based on your experiment.
newraw$FragmentIon<-"sum"
newraw$ProductCharge<-NA
newraw$IsotopeLabelType<-"L"

raw<-newraw
# now 'raw' is ready to use MSstats
```

2. different option for dataProcess

Only different options in `dataProcess` you should use for Skyline output are `skylineReport=TRUE` (remove `Truncated=TRUE` rows and handle intensity=0) and `censoredInt="0"`(use intensity=0 as censored to handle missing values).

Convert MaxQuant output to MSstats required input

If you have MaxQuant output, please follow this section. Here controlled mixture data with dynamic range bench mark (Cox, 2014 in MCP) is used to demonstrate.

1. First, get protein ID information

```
proteinGroups<-read.table("DDA2014_proteinGroups.txt", sep="\t", header=TRUE)
```

2. Read in annotation including condition and biological replicates: annotation.csv

```
annot <- read.csv("DDA2014_annotation.csv", header=TRUE)
```

3. Read in MaxQuant file: evidence.txt

```
infile <- read.table("evidence.txt", sep="\t", header=TRUE)
```

4. Reformat for MSstats required input

```
# check options for converting format
?MaxQtoMSstatsFormat
```

```
msstats.raw<-MaxQtoMSstatsFormat(evidence=infile, annotation=annot, proteinGroups=proteinGroups)

# now 'msstats.raw' is ready for MSstats
head(msstats.raw)
```

```
##   ProteinName        PeptideSequence PrecursorCharge FragmentIon
## 1      A5A614            QVAESTPDIPK               2          NA
## 2   O00762ups            DPAATSVAAAR               2          NA
## 3   O00762ups FLTPCYHPNVDTQGNICLDILK               2          NA
## 4   O00762ups FLTPCYHPNVDTQGNICLDILK               3          NA
## 5   O00762ups             GAEPSGGAAR               2          NA
## 6   O00762ups          GISAFPESDNLFK               2          NA
##   ProductCharge IsotopeLabelType Condition BioReplicate
## 1            NA                L      UPS1            1
## 2            NA                L      UPS1            1
## 3            NA                L      UPS1            1
## 4            NA                L      UPS1            1
## 5            NA                L      UPS1            1
## 6            NA                L      UPS1            1
##                             Run  Intensity
## 1 20130510_EXQ1_IgPa_QC_UPS1_01         NA
## 2 20130510_EXQ1_IgPa_QC_UPS1_01 1144800000
## 3 20130510_EXQ1_IgPa_QC_UPS1_01   32793000
## 4 20130510_EXQ1_IgPa_QC_UPS1_01  566960000
## 5 20130510_EXQ1_IgPa_QC_UPS1_01   58709000
## 6 20130510_EXQ1_IgPa_QC_UPS1_01  861090000
```