

US HUPO 2018 Short course - Section 7 : Protein significance analysis with MSstats

Meena Choi

March 11, 2018

This section describes steps and considerations 1) to properly format data processed by Skyline, prior to the **MSstats** analysis. 2) to properly use **MSstats** for significance analysis.

In the following example, the quantified peak intensities data from skyline file for **iPRG_10ppm_2rt_15cut_nosingle** of ABRF 2015 is used.

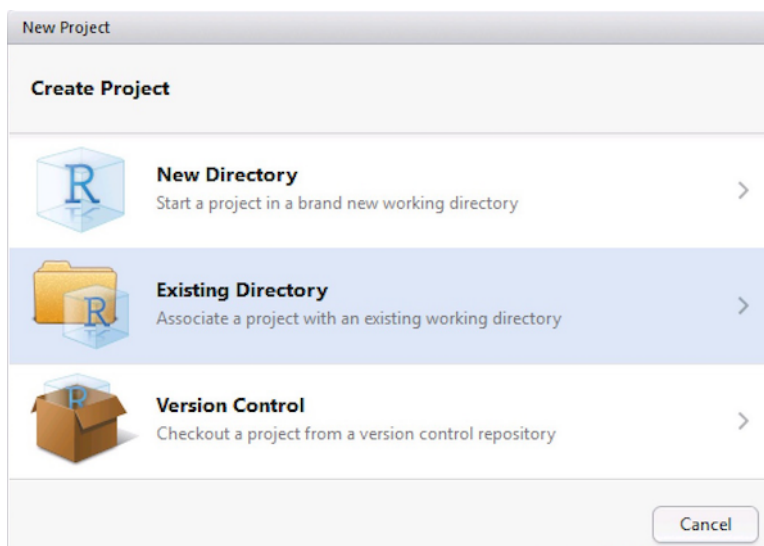
0. Prerequisites

Export the report from Skyline.

File > Export > Report > select MSstat Input > save .csv file.

1. Create a new Rstudio project

From the menu, select **File > New Project...**, then select **Existing Directory** and choose the directory where you downloaded 1) this script, 2) MSstats input report from Skyline, and 3) the example datasets for this tutorial. All the output files we'll be creating in this tutorial will be saved in the 'working directory' that now has been set by Rstudio.



Let's verify the working directory path with the `get working directory` command.

```
getwd()
```

```
## [1] "/Users/meenachoi/Dropbox/visits/2018/03_USHUP0/prep-Day2"
```

2. Read Skyline output

The required input data is generated automatically when using **MSstats** report format in Skyline. We first load and access the dataset processed by Skyline. The name of saved file from Skyline using **MSstats** report format is 'iPRG_10ppm_2rt_15cut_nosingle.csv'. or you can use the published data from **todo : add link**, This file is available in **panorama**.

```
# Read output from skyline
```

```
raw <- read.csv(file="iPRG_10ppm_2rt_15cut_nosingle.csv")
```

```
head(raw)
```

```
##               ProteinName PeptideSequence PeptideModifiedSequence
## 1 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
## 2 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
## 3 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
## 4 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
## 5 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
## 6 DECOY_sp|POCF18|YM085_YEAST      KDMYGPNPFQK      KDM[+16]YGNPFQK
##      PrecursorCharge PrecursorMz FragmentIon ProductCharge ProductMz
## 1                3      415.1974 precursor                3      415.1974
## 2                3      415.1974 precursor                3      415.1974
## 3                3      415.1974 precursor                3      415.1974
## 4                3      415.1974 precursor                3      415.1974
## 5                3      415.1974 precursor                3      415.1974
## 6                3      415.1974 precursor                3      415.1974
##      IsotopeLabelType Condition BioReplicate      FileName
## 1             light        NA            NA JD_06232014_sample1_B.raw
## 2             light        NA            NA JD_06232014_sample1_C.raw
## 3             light        NA            NA JD_06232014_sample1-A.raw
## 4             light        NA            NA JD_06232014_sample2_A.raw
## 5             light        NA            NA JD_06232014_sample2_B.raw
## 6             light        NA            NA JD_06232014_sample2_C.raw
##      Area StandardType Truncated annotation_QValue
## 1  147327            NA      False              NA
## 2 1373397            NA      False              NA
## 3   71765            NA      False              NA
## 4   66387            NA      False              NA
## 5  107736            NA      False              NA
## 6  380812            NA      False              NA
```

3. Load MSstats

Load MSstats first. Then you are ready to start MSstats.

```
library(MSstats)
?MSstats
```

4. Preprocessing for skyline output

4.1. Set annotation file

Annotation information is required to fill in `Condition` and `BioReplicate` for corresponding `Run` information. Users have to prepare as csv or txt file like 'iPRG_skyline_annotation.csv', which includes `Run`, `Condition`, and `BioReplicate` information, and load it in R.

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

```
##              Run Condition BioReplicate
## 1 JD_06232014_sample1-A.raw Condition1      1
## 2 JD_06232014_sample1-B.raw Condition1      2
## 3 JD_06232014_sample1-C.raw Condition1      3
## 4 JD_06232014_sample2-A.raw Condition2      4
## 5 JD_06232014_sample2-B.raw Condition2      5
## 6 JD_06232014_sample2-C.raw Condition2      6
## 7 JD_06232014_sample3-A.raw Condition3      7
## 8 JD_06232014_sample3-B.raw Condition3      8
## 9 JD_06232014_sample3-C.raw Condition3      9
## 10 JD_06232014_sample4-B.raw Condition4     10
## 11 JD_06232014_sample4-A.raw Condition4     11
## 12 JD_06232014_sample4-C.raw Condition4     12
```

4.2. Preprocessing with SkylinetoMSstatsFormat

The input data for `MSstats` is required to contain variables of `ProteinName`, `PeptideSequence`, `PrecursorCharge`, `FragmentIon`, `ProductCharge`, `IsotopeLabelType`, `Condition`, `BioReplicate`, `Run`, `Intensity`. These variable names should be fixed. `MSstats` input from Skyline adapts the column scheme of the dataset so that it fits `MSstats` input format. However there are several extra column names and also some of them need to be changed. `SkylinetoMSstatsFormat` function helps pre-processing for making right format of `MSstats` input from Skyline output. For example, it renames some column name, and replace truncated peak intensities with NA. Another important step is to handle isotopic peaks before using `dataProcess`. The output from Skyline for DDA experiment has several measurements of peak area from the monoisotopic, M+1 and M+2 peaks. To get a robust measure of peptide intensity, we can sum over isotopic peaks per peptide or use the highest peak. Here we take a summation per peptide ion.

Here is the summary of pre-processing steps in `SkylinetoMSstatsFormat` function (in orange box below).

In Skyline

Remove duplicated rows : exactly same values in some rows
Remove decoy proteins
Remove protein which has only one peptide per protein

In MSstats

SkylinetoMSstatsFormat

- Rename column names
 - Replace NA for truncated rows
 - Sum of isotopic peaks per peptide and charge
-
- Replace intensity = 1 with zero if intensity < 1
 - Log 2 transform for intensity
 - Normalization
 - **Extra step for imputation** : Distinguish missing at random and censored missing
 - Decide cutoff for censored missing values among all $\log_2(\text{intensity}) > 0$.
 - If $\log_2(\text{intensity}) < \text{cutoff}$, $\log_2(\text{intensity})$ replaces with zero and is considered as censored missing values (**censoredInt='0'**), then, will be imputed.
 - NA will be remained as NA, which are missing at random.

```
# reformatting and pre-processing for Skyline output.  
quant <- SkylinetoMSstatsFormat(raw, annotation=annot)
```

```
## ** Proteins, which names include DECOY, are removed.  
## ** Peptides, that are used in more than one proteins, are removed.  
## Warning in SkylinetoMSstatsFormat(raw, annotation = annot): NAs introduced  
## by coercion  
## ** Truncated peaks are replaced with NA.  
## ** For DDA datasets, three isotopic peaks per feature and run are summed.  
## ** 4 features have all NAs or zero intensity values and are removed.  
## ** 13 features have 1 or 2 intensities across runs and are removed.
```

```
head(quant)
```

```
##           ProteinName      PeptideSequence PrecursorCharge  
## 1 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
## 2 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
## 3 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
## 4 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
## 5 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
## 6 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16] YTPDTAADEEARK      3  
##   FragmentIon ProductCharge IsotopeLabelType  Condition BioReplicate  
## 1          sum           NA                L Condition1         3  
## 2          sum           NA                L Condition2         4  
## 3          sum           NA                L Condition4        10  
## 4          sum           NA                L Condition2         6  
## 5          sum           NA                L Condition4        12  
## 6          sum           NA                L Condition3         8
```

```
##           Run Intensity StandardType
## 1 JD_06232014_sample1_C.raw 7863713      NA
## 2 JD_06232014_sample2_A.raw 977615      NA
## 3 JD_06232014_sample4_B.raw 4102785      NA
## 4 JD_06232014_sample2_C.raw 6547299      NA
## 5 JD_06232014_sample4_C.raw 3972464      NA
## 6 JD_06232014_sample3_B.raw 8896052      NA
```

For further details, visit the help file using the following code.

```
?SkylinetoMSstatsFormat
```

5. Processing the data for analysis : Normalizing and summarizing data with dataProcess

! Always pay attention to the default options

After reading the datasets, `MSstats` performs 1) logarithm transformation of `Intensity` column, 2) normalization, 3) feature selection, (all features vs subset of features), 4) imputation for censored missing value, which are below the cutoff and undetectable, 5) run-level summarization.

To get started with this function, visit the help section of `dataProcess` first:

Default normalization and summarization options

Then (1) normalization will be performed first. The default option for normalization is `equalizeMedians`. 'equalizeMedians' fits for label-based SRM experiments, which we can use reference signals. There are three more options for normalization. Depending on the suitable assumption for your experiment, you can choose one of them.

Then, (2) run level summarization will be performed including missing value imputation by accelerated failure model and robust parameter estimation by TMP (Tukey's median polish).

Below show the default for all options in `dataProcess` except `skylineReport` and `censoredInt`. For input from Skyline, `skylineReport=TRUE` and `censoredInt='0'`

```
# `TMP` is the default model-based summarization option
quant.processed <- dataProcess(raw = quant,
                              logTrans=2,
                              normalization = 'equalizeMedians',
                              summaryMethod = 'TMP',
                              MBimpute=TRUE,
                              censoredInt='0',
                              cutoffCensored='minFeature',
                              maxQuantileforCensored = 0.999)
```

Let's check output from `dataProcess`

```
# show the name of outputs
names(quant.processed)
```

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

```
# show reformatted and normalized data.
# 'ABUNDANCE' column has normalized log2 transformed intensities.
head(quant.processed$ProcessedData)
```

```
##          PROTEIN          PEPTIDE TRANSITION
## 27092 sp|D6VTK4|STE2_YEAST EGEVEPVDM[+16]YTPDTAADEEARK_3 sum_NA
## 10700 sp|D6VTK4|STE2_YEAST EGEVEPVDMYTPDTAADEEARK_3 sum_NA
## 20407 sp|D6VTK4|STE2_YEAST FYPGTLSSFQTD SINNDK_2 sum_NA
## 2213 sp|D6VTK4|STE2_YEAST IGPFDASYK_2 sum_NA
## 938 sp|D6VTK4|STE2_YEAST KETTSDK_2 sum_NA
## 26116 sp|D6VTK4|STE2_YEAST NQFYQLPTPTSSK_2 sum_NA
##          FEATURE LABEL GROUP_ORIGINAL
## 27092 EGEVEPVDM[+16]YTPDTAADEEARK_3_sum_NA L Condition1
## 10700 EGEVEPVDMYTPDTAADEEARK_3_sum_NA L Condition1
## 20407 FYPGTLSSFQTD SINNDK_2_sum_NA L Condition1
## 2213 IGPFDASYK_2_sum_NA L Condition1
## 938 KETTSDK_2_sum_NA L Condition1
## 26116 NQFYQLPTPTSSK_2_sum_NA L Condition1
## SUBJECT_ORIGINAL RUN GROUP SUBJECT SUBJECT_NESTED INTENSITY
## 27092 1 1 1 1 1.1 5222795
## 10700 1 1 1 1 1.1 182195648
## 20407 1 1 1 1 1.1 86229170
## 2213 1 1 1 1 1.1 157996653
## 938 1 1 1 1 1.1 177684007
## 26116 1 1 1 1 1.1 140368798
## ABUNDANCE METHOD originalRUN censored
## 27092 22.07353 1 JD_06232014_sample1_B.raw FALSE
## 10700 27.19805 1 JD_06232014_sample1_B.raw FALSE
## 20407 26.11881 1 JD_06232014_sample1_B.raw FALSE
## 2213 26.99246 1 JD_06232014_sample1_B.raw FALSE
## 938 27.16188 1 JD_06232014_sample1_B.raw FALSE
## 26116 26.82179 1 JD_06232014_sample1_B.raw FALSE
```

```
# This table includes run-level summarized log2 intensities. (column : LogIntensities)
# Now one summarized log2 intensities per Protein and Run.
# NumMeasuredFeature : show how many features are used for run-level summarization.
# If there is no missing value, it should be the number of features in certain protein.
# MissingPercentage : the number of missing features / the number of features in certain protein.
head(quant.processed$RunlevelData)
```

```
## RUN Protein LogIntensities NumMeasuredFeature
## 1 1 sp|D6VTK4|STE2_YEAST 26.81232 8
## 2 2 sp|D6VTK4|STE2_YEAST 26.60786 8
## 3 3 sp|D6VTK4|STE2_YEAST 26.58301 8
## 4 4 sp|D6VTK4|STE2_YEAST 26.83563 8
## 5 5 sp|D6VTK4|STE2_YEAST 26.79430 8
## 6 6 sp|D6VTK4|STE2_YEAST 26.60863 8
## MissingPercentage more50missing NumImputedFeature
## 1 0 FALSE 0
## 2 0 FALSE 0
## 3 0 FALSE 0
## 4 0 FALSE 0
## 5 0 FALSE 0
## 6 0 FALSE 0
```

```
##          originalRUN GROUP GROUP_ORIGINAL SUBJECT_ORIGINAL
## 1 JD_06232014_sample1_B.raw      1      Condition1          1
## 2 JD_06232014_sample1_C.raw      1      Condition1          1
## 3 JD_06232014_sample1-A.raw      1      Condition1          1
## 4 JD_06232014_sample2_A.raw      2      Condition2          2
## 5 JD_06232014_sample2_B.raw      2      Condition2          2
## 6 JD_06232014_sample2_C.raw      2      Condition2          2
## SUBJECT_NESTED SUBJECT
## 1          1.1      1
## 2          1.1      1
## 3          1.1      1
## 4          2.2      2
## 5          2.2      2
## 6          2.2      2

# show which summarization method is used.
head(quant.processed$SummaryMethod)

## [1] "TMP"
```

6. Visualization of processed data

6.1. Quality control plots

Now let's look at what the equalize medians procedure did to our data. QC plot is good to see the distribution of intensities per MS run and outliers. So, it is good visualization to check normalization. However, not good to see individual intensities.

```
# QC plot for normalized data with equalize median method
dataProcessPlots(data = quant.processed,
  type="QCplot",
  width=7, height=7,
  which.Protein = 1,
  address='iPRG_skyline_equalizeNorm_')
```

Then, iPRG_skyline_equalizeNorm_QCPlot.pdf are generated in the current directory.

Now the median log2 intensities per run across MS runs are the same.

6.2. Profile plots

Profile plot is good visualization to check individual measurements. Each dot means one intensity. The dots are linked with line per feature. If line is disconnected, that means there is no value (missing value). Color means different peptides and charge stages. different line type means different transition.

```
# if you have many MS runs, adjust width of plot (makd wider)
# Profile plot for the data with equalized median method
dataProcessPlots(data = quant.processed,
  type="Profileplot",
  width=7, height=7,
  address="iPRG_skyline_equalizeNorm_")
```

All proteins

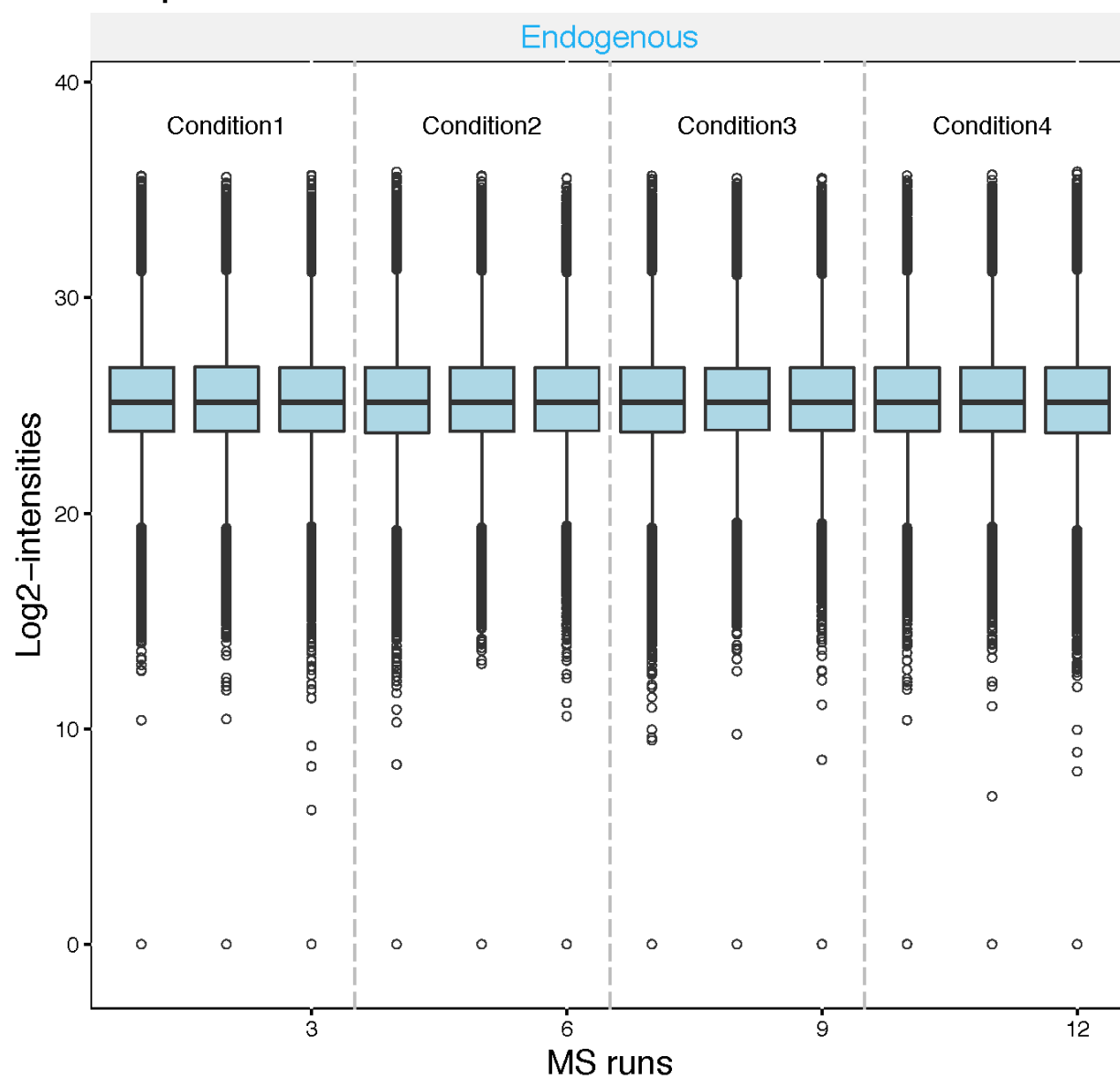


Figure 1:

iPRG_skyline_equalizeNorm_ProfilePlot.pdf and iPRG_skyline_equalizeNorm_ProfilePlot_wSummarization.pdf are generated in the current directory.

Then, Let's go through profile plots to see overall quality of data.

There are two pdfs for each protein, first is profile plot with normalized data and second plot is profile plot with normalized data and summarized data. This profile plot shows each peptide transition across runs, grouped per condition. Each peptide has a different colour/type layout.

This plot shows The panel on the right shows the same transitions in grey, with the values as summarized by the model overlayed in red.

Instead of making all profiles plots for all proteins, we can make plot for individual protein. Here is the example of spike-in protein, sp|P44015|VAC2_YEAST

```
dataProcessPlots(data = quant.processed,
                  type="Profileplot",
                  featureName="NA",
                  width=7, height=7,
                  which.Protein = 'sp|P44015|VAC2_YEAST',
                  address="iPRG_skyline_equalizeNorm_P44015_")
```

This is the study design. Let's check visualization for those proteins.

**** ! Extreme case : no measurement in certain run.****

MSstats needs at least one measurement per run. If not, can not get run-summarized value.

6.3. Condition plots

Condition plots illustrate the systematic difference between conditions. The dots indicate the mean of all measurements (Peptide ion and multiple MS runs) in each condition and default error bar is CI with 0.95 significant level. However, it is not related with model-based analysis.

```
dataProcessPlots(data = quant.processed,
                  type="conditionplot",
                  width=7, height=7,
                  address="iPRG_skyline_equalizeNorm_")
```

This is the condition plot for protein, sp|P44015|VAC2_YEAST .

7. Finding differentially abundant proteins across conditions

7.1 Assign contrast matrix

After we normalized the data and summarized each protein's behaviour across conditions in `dataProcess` step, 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
```

We have to tell `groupComparison` which are the conditions we would like to compare. You can make your `contrast.matrix` in R in a text editor. We define our contrast matrix by adding a column for every condition, **in alphabetical order**. We add a row for every comparison we would like to make between groups of conditions.

spID6VTK4ISTE2_YEAST

• Detected data

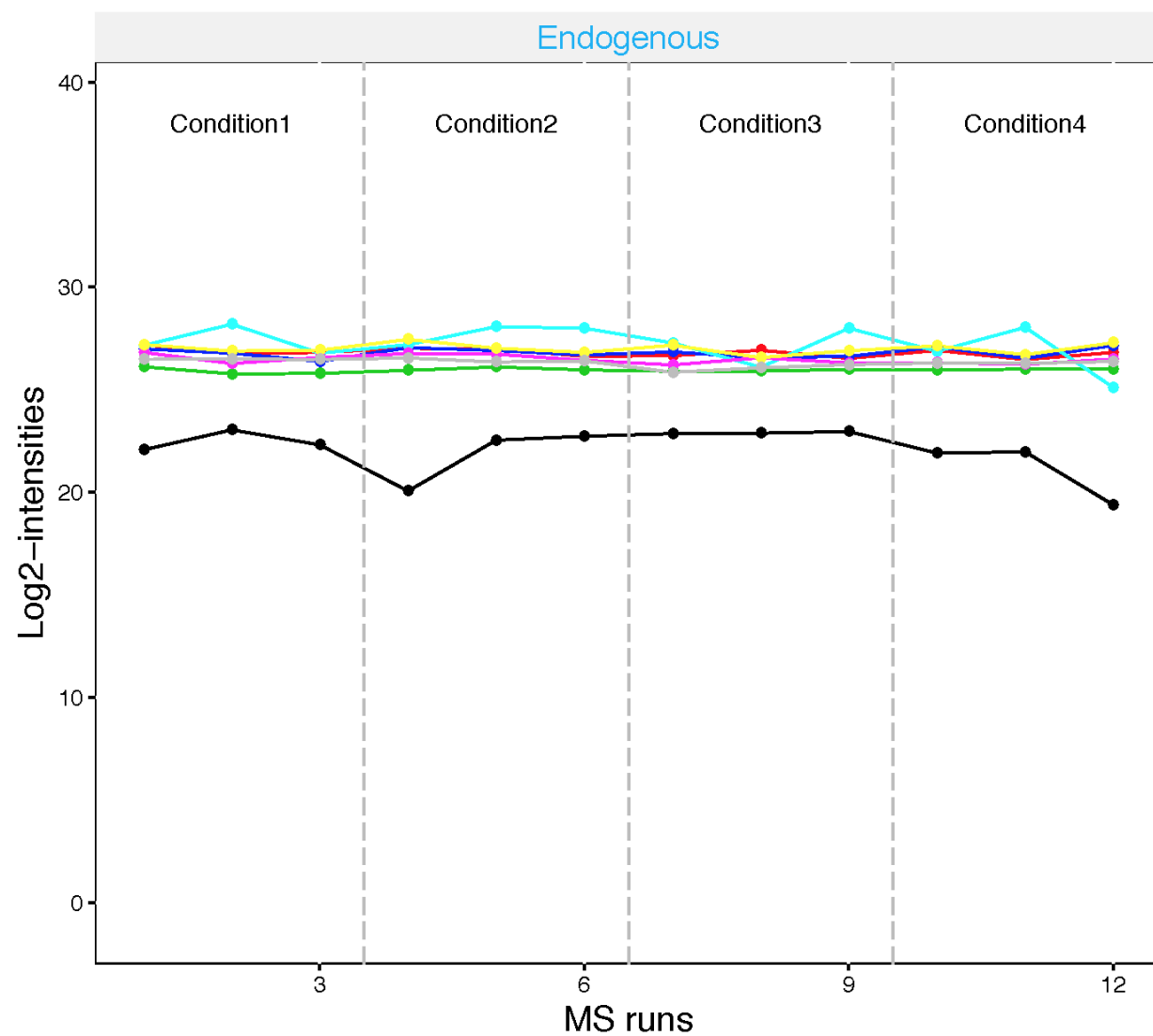


Figure 2:

spID6VTK4ISTE2_YEAST

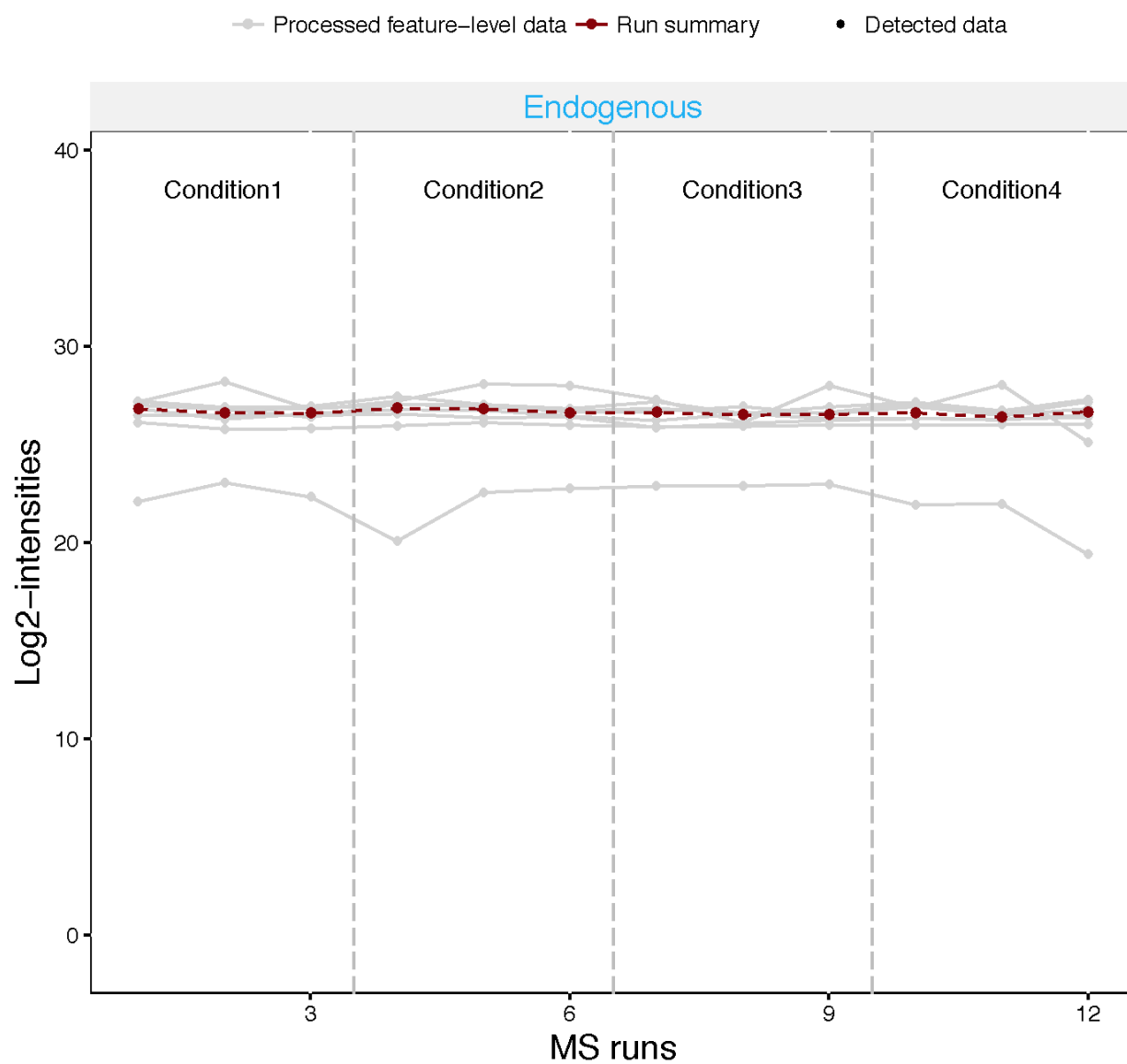


Figure 3:

spIP44015IVAC2_YEAST

• Detected data ○ Censored missing data

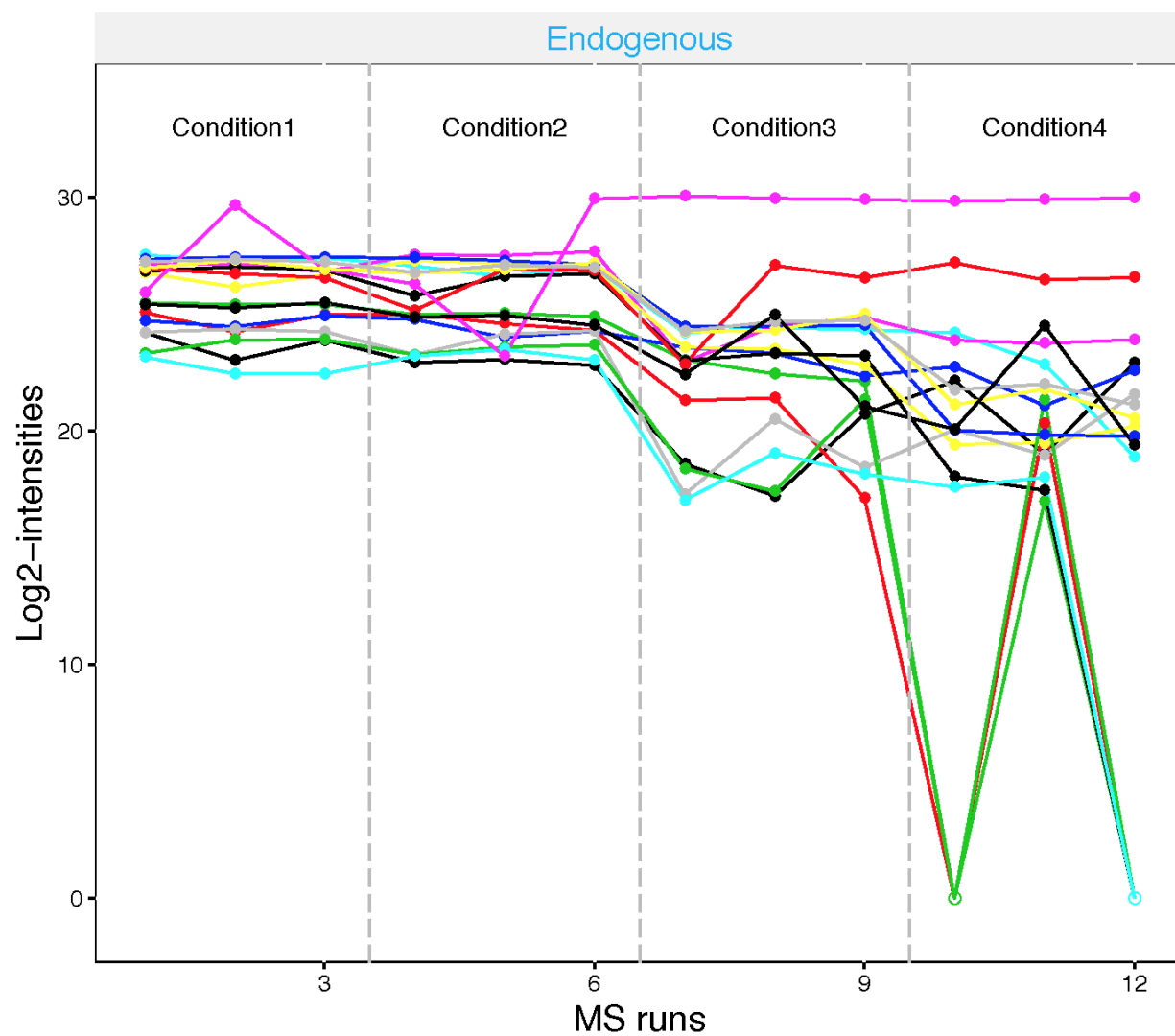


Figure 4:

spIP44015IVAC2_YEAST

— Processed feature-level data — Run summary • Detected data ○ Censored missing data

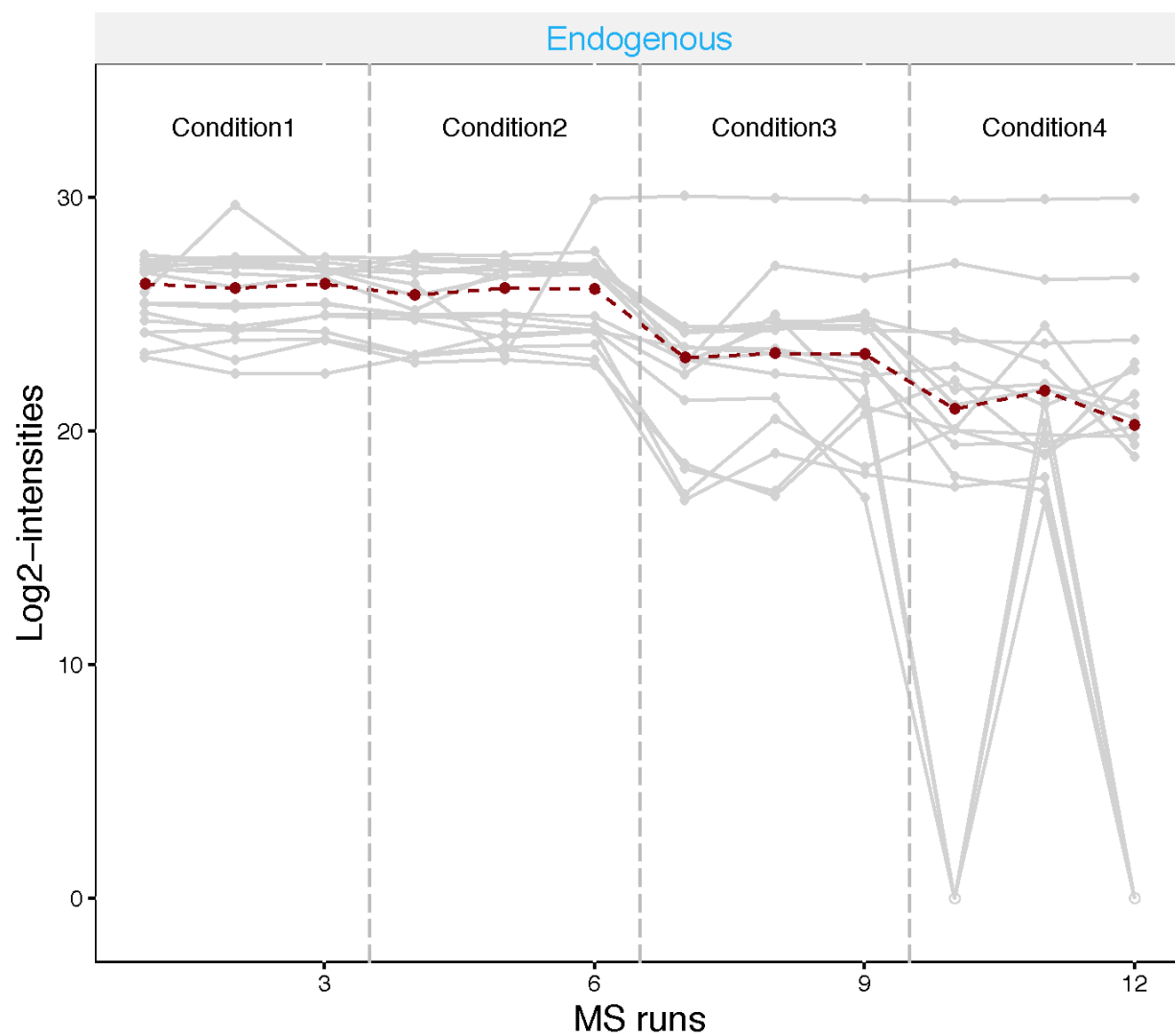


Figure 5:

	Samples			
Protein name	1	2	3	4
sp P44015 VAC2_YEAST	65	55	15	2
sp P55752 ISCB_YEAST	55	15	2	65
sp P44374 SFG2_YEAST	15	2	65	55
sp P44983 UTR6_YEAST	2	65	55	15
sp P44683 PGA4_YEAST	11	0.6	10	500
sp P55249 ZRT4_YEAST	10	500	11	0.6

Figure 6:

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 denominator of the ratio or fold-change.

If you have multiple groups, you can assign any group comparisons you are interested in. For example, if you have 4 different conditions, Condition1, Condition2, Condition3, Condition4, there are many possible comparisons.

```
# check unique conditions and check order of condition information
```

```
# In this case, Disease and Healthy
```

```
unique(quant.processed$ProcessedData$GROUP_ORIGINAL)
```

```
## [1] Condition1 Condition2 Condition3 Condition4
```

```
## Levels: Condition1 Condition2 Condition3 Condition4
```

```
comparison1<-matrix(c(-1,1,0,0),nrow=1)
```

```
comparison2<-matrix(c(-1,0,1,0),nrow=1)
```

```
comparison3<-matrix(c(-1,0,0,1),nrow=1)
```

```
comparison4<-matrix(c(0,-1,1,0),nrow=1)
```

```
comparison5<-matrix(c(0,-1,0,1),nrow=1)
```

```
comparison6<-matrix(c(0,0,-1,1),nrow=1)
```

```
comparison<-rbind(comparison1, comparison2, comparison3, comparison4, comparison5, comparison6)
```

```
row.names(comparison)<-c("C2-C1", "C3-C1", "C4-C1", "C3-C2", "C4-C2", "C4-C3")
```

```
comparison
```

```
##      [,1] [,2] [,3] [,4]
```

```
## C2-C1  -1    1    0    0
```

```
## C3-C1  -1    0    1    0
```

```
## C4-C1  -1    0    0    1
```

```
## C3-C2   0   -1    1    0
```

```
## C4-C2   0   -1    0    1
```

```
## C4-C3   0    0   -1    1
```

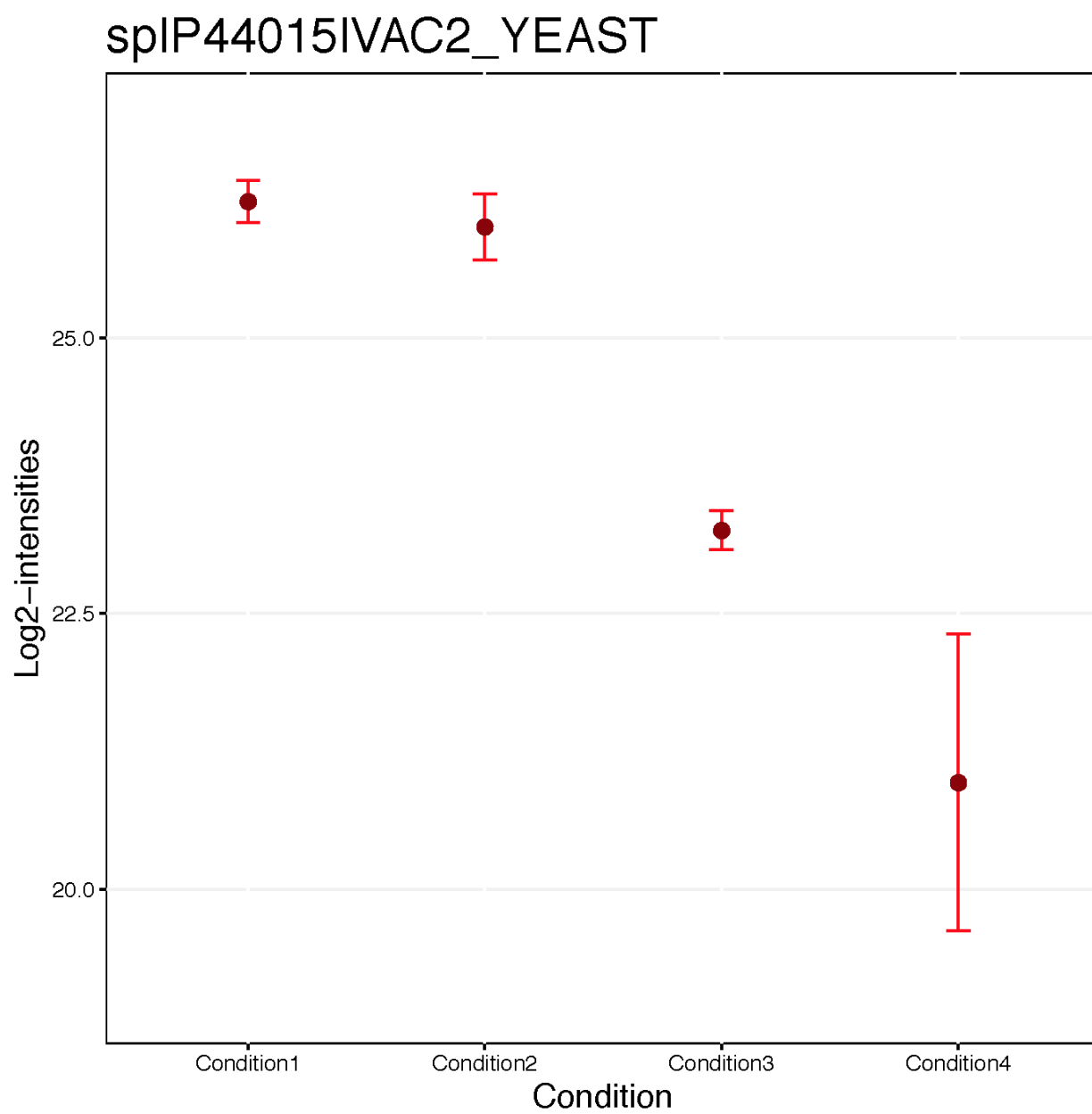


Figure 7:

7.2. Comparing conditions with groupComparison

groupComparison uses the run-level summarized data (\$RunlevelData from dataProcess function) for hypothesis testing.

```
test <- groupComparison(contrast.matrix=comparison, data=quant.processed)
```

Let's check the output.

```
names(test)
```

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

```
# Show test result
# Label : which comparison is used
# log2FC : estimated log2 fold change between Diseased and Healthy
# adj.pvalue : adjusted p value
# issue : detect whether this protein has any issue for comparison
# such as, there is measurement in certain group, or no measurement at all.
# MissingPercentage : the number of missing intensities/total number of intensities
# in conditions your are interested in for comparison
# ImputationPercentage : the number of imputed intensities/total number of intensities
# in conditions your are interested in for comparison
head(test$ComparisonResult)
```

```
##          Protein Label      log2FC      SE      Tvalue DF
## 1  sp|D6VTK4|STE2_YEAST C2-C1  0.07846162 0.09648008  0.8132416  8
## 7  sp|013297|CET1_YEAST C2-C1 -0.11119732 0.07749333 -1.4349278  8
## 13 sp|013329|FOB1_YEAST C2-C1 -0.16209422 0.29090086 -0.5572146  8
## 19 sp|013539|THP2_YEAST C2-C1 -0.43742107 0.82871351 -0.5278315  8
## 25 sp|013547|CCW14_YEAST C2-C1 -0.05774773 0.14672405 -0.3935805  8
## 31 sp|013563|RPN13_YEAST C2-C1 -0.16945187 0.09518591 -1.7802201  8
##          pvalue adj.pvalue issue MissingPercentage ImputationPercentage
## 1  0.4396100  0.9991155  NA              0              0
## 7  0.1892233  0.9991155  NA              0              0
## 13 0.5926243  0.9991155  NA              0              0
## 19 0.6119387  0.9991155  NA              0              0
## 25 0.7041724  0.9991155  NA              0              0
## 31 0.1129126  0.9991155  NA              0              0
```

```
# After fitting linear model, residuals and fitted values can be shown.
head(test$ModelQC)
```

```
##  RUN          PROTEIN ABUNDANCE NumMeasuredFeature MissingPercentage
## 1   1 sp|D6VTK4|STE2_YEAST  26.81232              8              0
## 2   2 sp|D6VTK4|STE2_YEAST  26.60786              8              0
## 3   3 sp|D6VTK4|STE2_YEAST  26.58301              8              0
## 4   4 sp|D6VTK4|STE2_YEAST  26.83563              8              0
## 5   5 sp|D6VTK4|STE2_YEAST  26.79430              8              0
## 6   6 sp|D6VTK4|STE2_YEAST  26.60863              8              0
##  more50missing NumImputedFeature      originalRUN GROUP
## 1          FALSE              0 JD_06232014_sample1_B.raw  1
## 2          FALSE              0 JD_06232014_sample1_C.raw  1
## 3          FALSE              0 JD_06232014_sample1-A.raw  1
## 4          FALSE              0 JD_06232014_sample2_A.raw  2
## 5          FALSE              0 JD_06232014_sample2_B.raw  2
## 6          FALSE              0 JD_06232014_sample2_C.raw  2
```



```
##   GROUP_ORIGINAL SUBJECT_ORIGINAL SUBJECT_NESTED SUBJECT  residuals
## 1   Condition1           1           1.1           1  0.14458899
## 2   Condition1           1           1.1           1 -0.05986794
## 3   Condition1           1           1.1           1 -0.08472105
## 4   Condition2           2           2.2           2  0.08944428
## 5   Condition2           2           2.2           2  0.04811446
## 6   Condition2           2           2.2           2 -0.13755874
##      fitted
## 1 26.66773
## 2 26.66773
## 3 26.66773
## 4 26.74619
## 5 26.74619
## 6 26.74619
```

```
# Fitted model per protein
head(test$fittedmodel)
```

```
## [[1]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
##    26.66773     0.07846    -0.11566    -0.11841
##
##
## [[2]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
##    24.70386    -0.11120    -0.14451    -0.09633
##
##
## [[3]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
##    23.6207     -0.1621     0.1209    -0.3511
##
##
## [[4]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
```

```
##      26.4006      -0.4374      -0.4318      -0.5024
##
##
## [[5]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
##      27.04775      -0.05775      -0.29114      -0.09482
##
##
## [[6]]
##
## Call:
## lm(formula = ABUNDANCE ~ GROUP, data = data2)
##
## Coefficients:
## (Intercept)      GROUP2      GROUP3      GROUP4
##      26.18595      -0.16945      -0.07609      -0.21625
```

Let's save the testing result as .csv file.

```
Skyline.intensity.comparison.result <- test$ComparisonResult

write.csv(Skyline.intensity.comparison.result, file='testResult_iprg_skyline.csv')
```

Let's inspect the results to see what proteins are changing significantly between Diseased and Healthy.

```
head(Skyline.intensity.comparison.result)
```

```
##              Protein Label      log2FC      SE      Tvalue DF
## 1  sp|D6VTK4|STE2_YEAST C2-C1  0.07846162 0.09648008  0.8132416  8
## 7  sp|O13297|CET1_YEAST C2-C1 -0.11119732 0.07749333 -1.4349278  8
## 13 sp|O13329|FOB1_YEAST C2-C1 -0.16209422 0.29090086 -0.5572146  8
## 19 sp|O13539|THP2_YEAST C2-C1 -0.43742107 0.82871351 -0.5278315  8
## 25 sp|O13547|CCW14_YEAST C2-C1 -0.05774773 0.14672405 -0.3935805  8
## 31 sp|O13563|RPN13_YEAST C2-C1 -0.16945187 0.09518591 -1.7802201  8
##      pvalue adj.pvalue issue MissingPercentage ImputationPercentage
## 1  0.4396100  0.9991155    NA                0                0
## 7  0.1892233  0.9991155    NA                0                0
## 13 0.5926243  0.9991155    NA                0                0
## 19 0.6119387  0.9991155    NA                0                0
## 25 0.7041724  0.9991155    NA                0                0
## 31 0.1129126  0.9991155    NA                0                0
```

```
SignificantProteins <-
  Skyline.intensity.comparison.result[Skyline.intensity.comparison.result$adj.pvalue < 0.05 ,]
nrow(SignificantProteins)
```

```
## [1] 29
```

```
SignificantProteinsUpInDiseased <- SignificantProteins[SignificantProteins$log2FC > 2 ,]
nrow(SignificantProteinsUpInDiseased)
```

```
## [1] 14
```

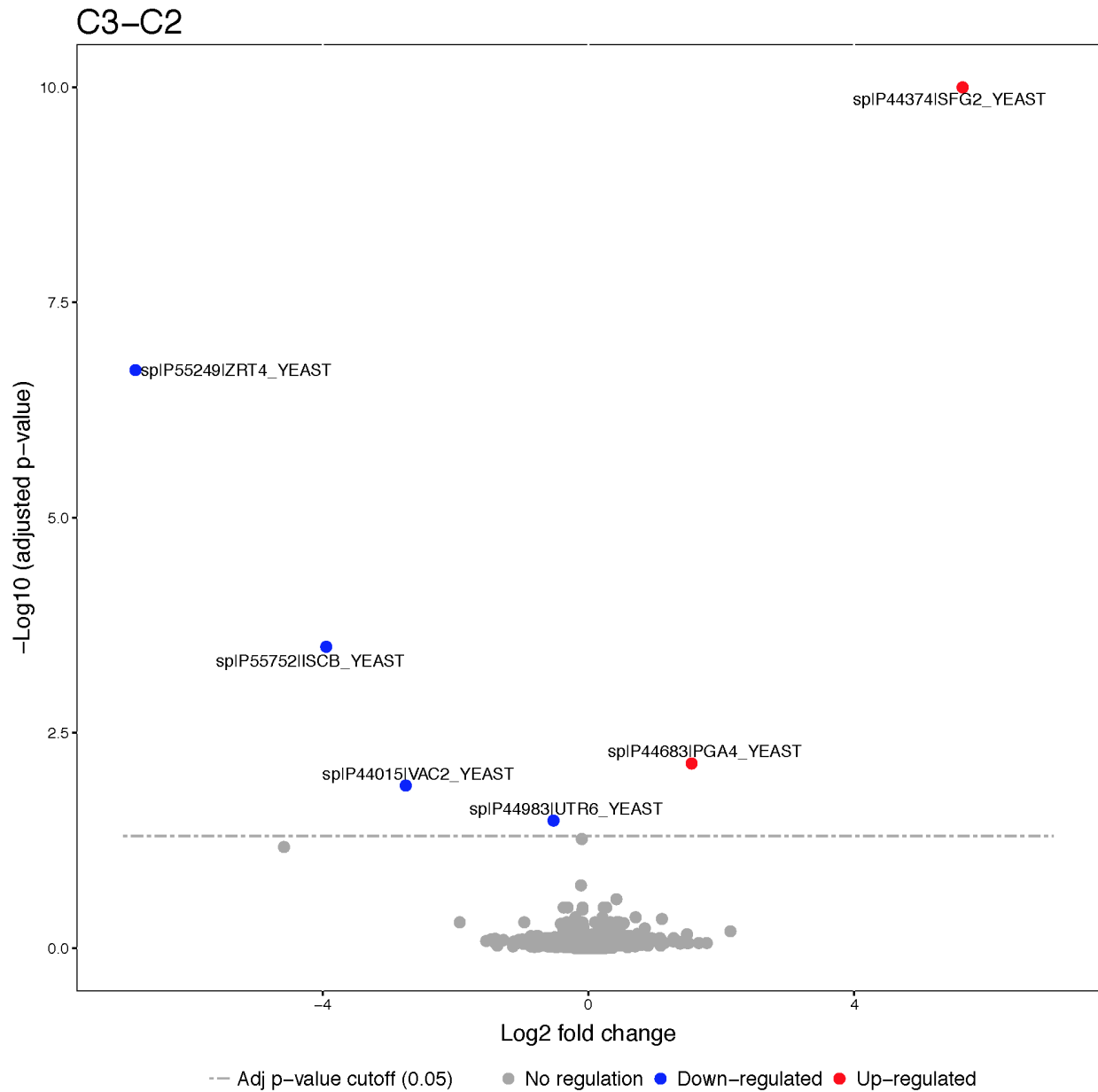
8. Visualization of differentially abundant proteins

?groupComparisonPlots

8.1. Volcano plot

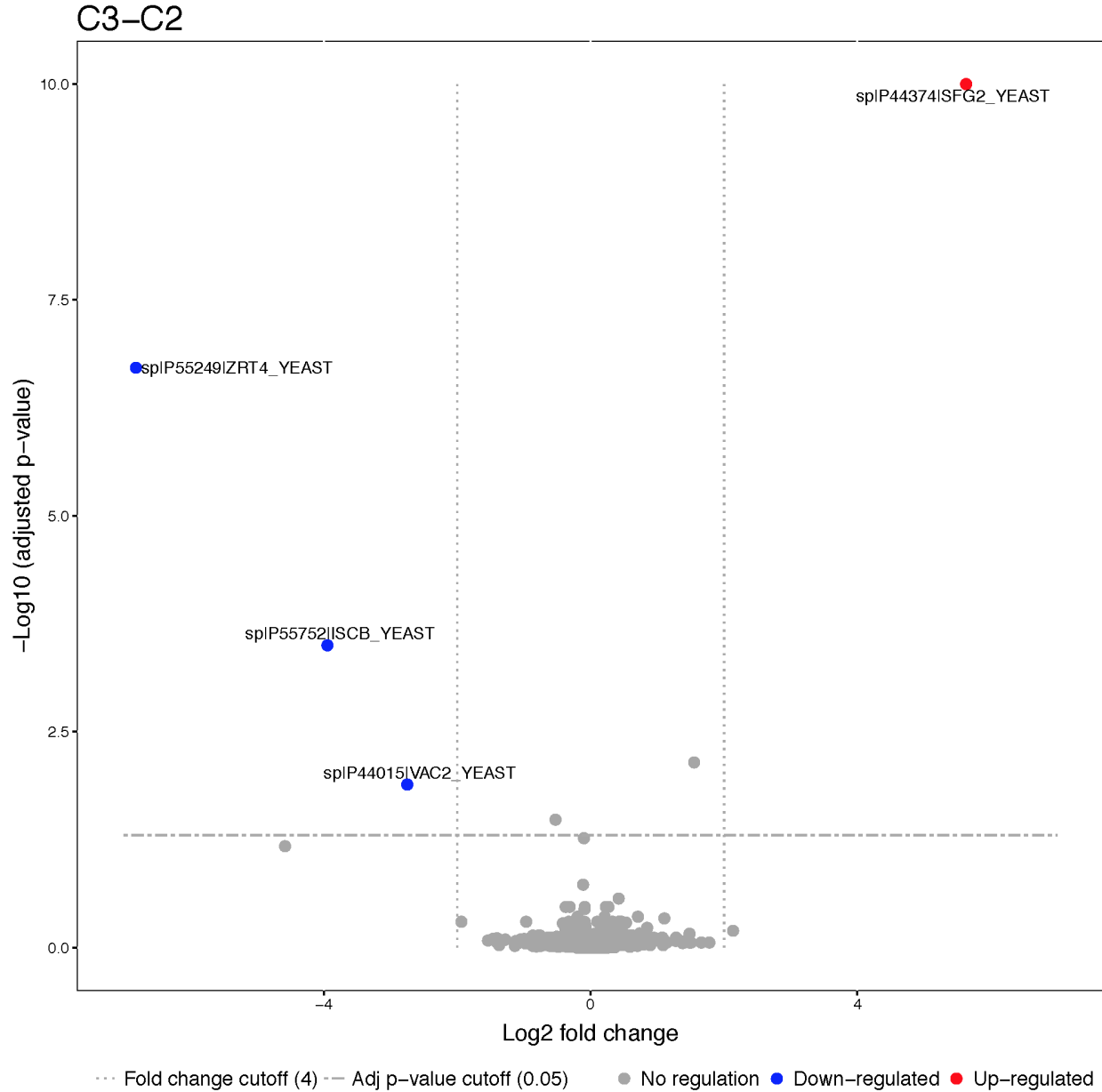
Volcano plots allow us to visually separate strong changes, which are not significant, from strong and significant changes. Look for these subjects in the upper right and upper left quadrants of the plot. Protein name will be shown only for significant proteins.

```
groupComparisonPlots(data = Skyline.intensity.comparison.result,  
                     type = 'VolcanoPlot',  
                     address = 'iprg_skyline_')
```



We can set up estimated fold change cutoff.

```
groupComparisonPlots(data = Skyline.intensity.comparison.result,
  type = 'VolcanoPlot',
  sig = 0.05,
  FCcutoff = 2^2,
  address = 'iprg_skyline_FCcutoff4_')
```



8.2. Comparison plot

Comparison plots illustrate model-based estimates of log-fold changes, and the associated uncertainty, in several comparisons of conditions for one protein. X-axis is the comparison of interest. Y-axis is the log fold change. The dots are the model-based estimates of log-fold change, and the error bars are the model-based 95% confidence intervals (the option `sig` can be used to change the significance level of significance). For simplicity, the confidence intervals are adjusted for multiple comparisons within protein only, using the Bonferroni approach. For proteins with N comparisons, the individual confidence intervals are at the level of $1\text{-sig}/N$.

```
groupComparisonPlots(Skyline.intensity.comparison.result,
                      type="ComparisonPlot",
                      address="testResult_iprg_skyline_")
```

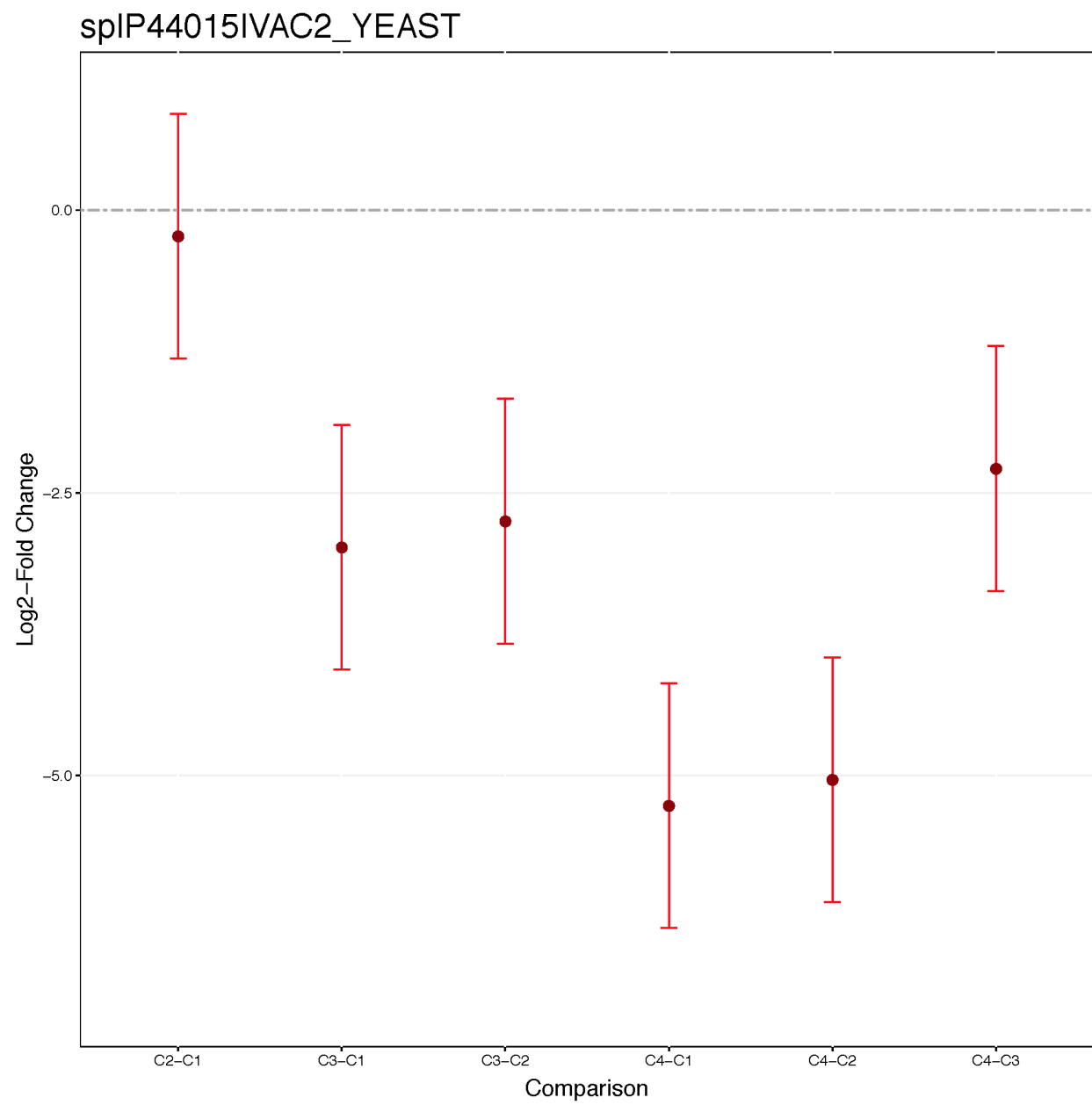


Figure 8:

9. Planning future experimental designs

This last analysis step views the dataset as a pilot study of a future experiment, utilizes its variance components, and calculates the minimal number of replicates required in a future experiment to achieve the desired statistical power. The calculation is performed by the function `designSampleSize`, which takes as input the fitted model in `groupComparison`. Sample size calculation assumes same experimental design (i.e. group comparison, time course or paired design) as in the current dataset, and uses the model fit to estimate the median variance components across all the proteins. Finally, sample size calculation assumes that a large proportion of proteins (specifically, 99%) will not change in abundance in the future experiment. This assumption also provides conservative results. Using the estimated variance components, the function relates the number of biological replicates per condition (`numSample`, rounded to 0 decimal), average statistical power across all the proteins (`power`), minimal fold change that we would like to detect (can be specified as a range, e.g. `desiredFC=c(1.1, 2)`), and the False Discovery Rate (FDR). The user should specify all these quantities but one, and the function will solve for the remainder. The quantity to solve for should be set to `= TRUE`.

```
?designSampleSize
```

9.1. Calculating statistical power

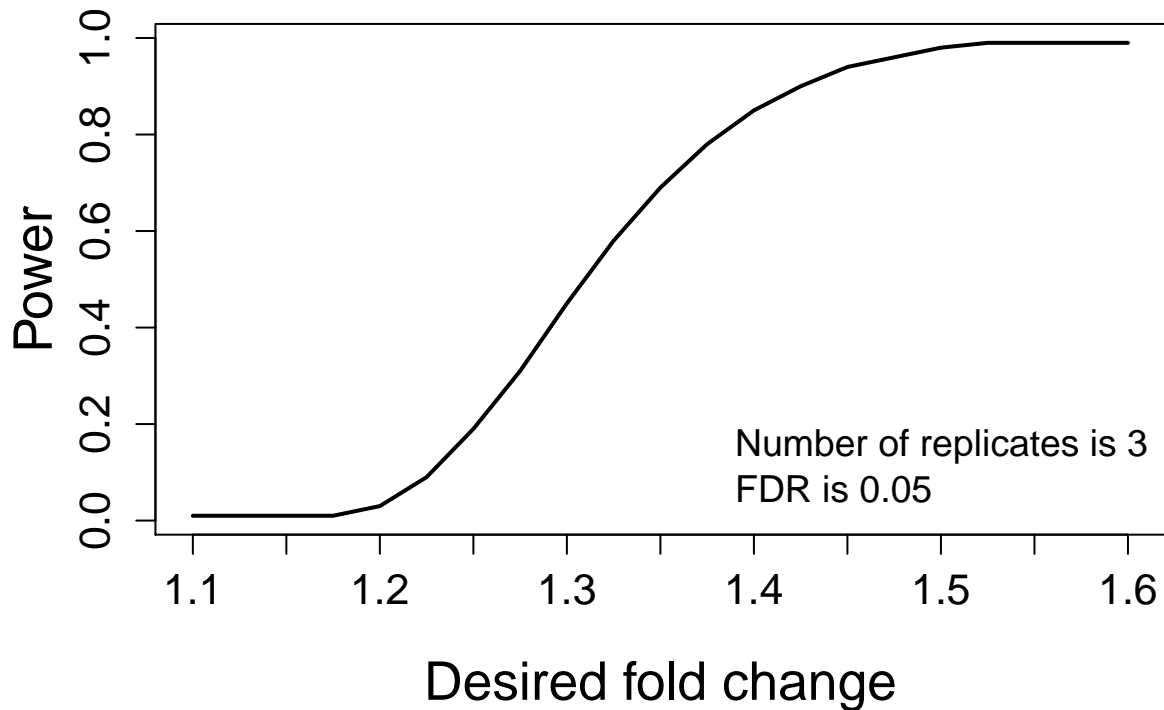
```
# calculate the power
test.power <- designSampleSize(data = test$fittedmodel,
                               desiredFC = c(1.1, 1.6),
                               FDR = 0.05,
                               power = TRUE,
                               numSample = 3)

test.power
```

##	desiredFC	numSample	FDR	power	CV
## 1	1.100	3	0.05	0.01	0.010
## 2	1.125	3	0.05	0.01	0.010
## 3	1.150	3	0.05	0.01	0.010
## 4	1.175	3	0.05	0.01	0.010
## 5	1.200	3	0.05	0.03	0.009
## 6	1.225	3	0.05	0.09	0.009
## 7	1.250	3	0.05	0.19	0.009
## 8	1.275	3	0.05	0.31	0.009
## 9	1.300	3	0.05	0.45	0.009
## 10	1.325	3	0.05	0.58	0.009
## 11	1.350	3	0.05	0.69	0.008
## 12	1.375	3	0.05	0.78	0.008
## 13	1.400	3	0.05	0.85	0.008
## 14	1.425	3	0.05	0.90	0.008
## 15	1.450	3	0.05	0.94	0.008
## 16	1.475	3	0.05	0.96	0.008
## 17	1.500	3	0.05	0.98	0.008
## 18	1.525	3	0.05	0.99	0.007
## 19	1.550	3	0.05	0.99	0.007
## 20	1.575	3	0.05	0.99	0.007
## 21	1.600	3	0.05	0.99	0.007

9.2. Visualizing the relationship between desired fold-change and power

```
designSampleSizePlots(data = test.power)
```



9.3. Designing sample size for desired fold-change

```
# Minimal number of biological replicates per condition
samplesize <- designSampleSize(data = test$fittedmodel,
                                desiredFC = c(1.1, 1.6),
                                FDR = 0.05,
                                power = 0.9,
                                numSample = TRUE)

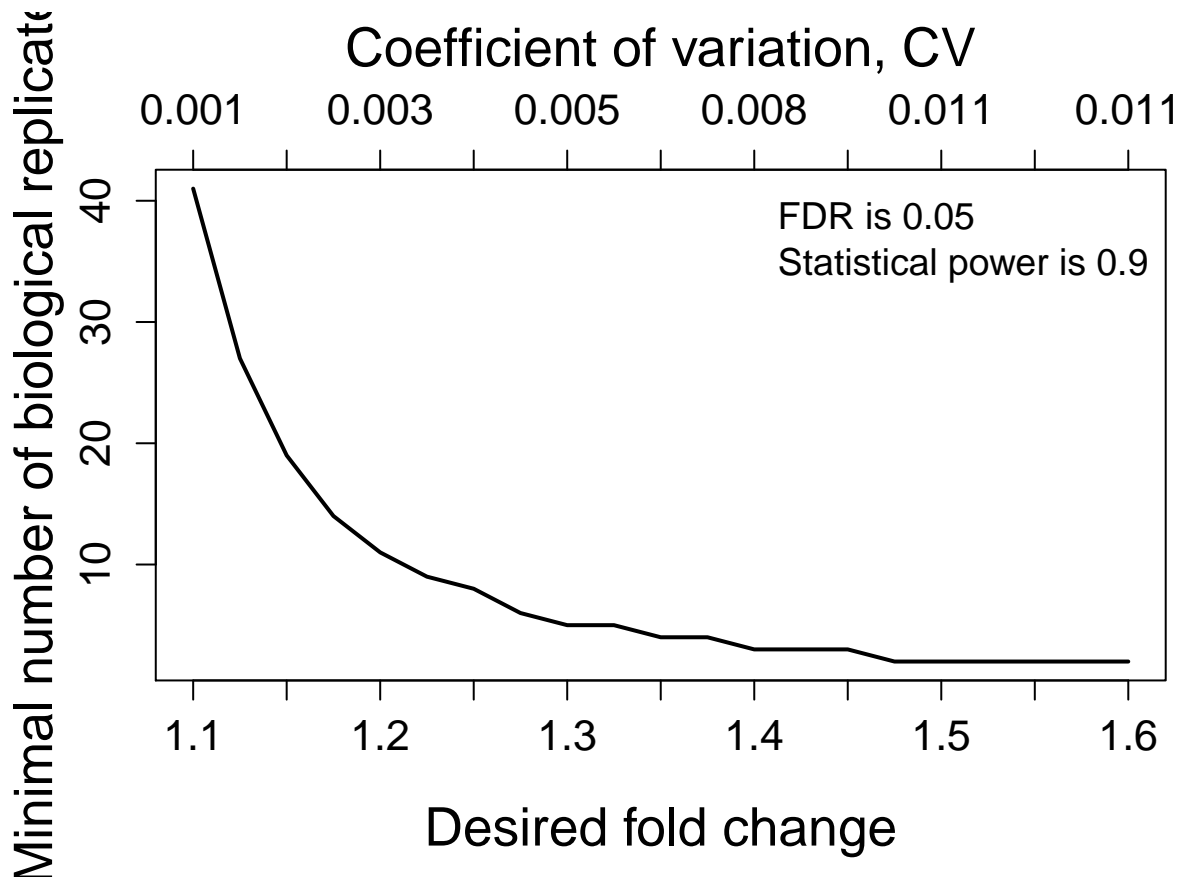
samplesize
```

##	desiredFC	numSample	FDR	power	CV
## 1	1.100	41	0.05	0.9	0.001
## 2	1.125	27	0.05	0.9	0.001
## 3	1.150	19	0.05	0.9	0.002
## 4	1.175	14	0.05	0.9	0.002
## 5	1.200	11	0.05	0.9	0.003
## 6	1.225	9	0.05	0.9	0.003
## 7	1.250	8	0.05	0.9	0.003
## 8	1.275	6	0.05	0.9	0.004
## 9	1.300	5	0.05	0.9	0.005
## 10	1.325	5	0.05	0.9	0.005
## 11	1.350	4	0.05	0.9	0.006
## 12	1.375	4	0.05	0.9	0.006
## 13	1.400	3	0.05	0.9	0.008
## 14	1.425	3	0.05	0.9	0.008

## 15	1.450	3	0.05	0.9	0.008
## 16	1.475	2	0.05	0.9	0.012
## 17	1.500	2	0.05	0.9	0.011
## 18	1.525	2	0.05	0.9	0.011
## 19	1.550	2	0.05	0.9	0.011
## 20	1.575	2	0.05	0.9	0.011
## 21	1.600	2	0.05	0.9	0.011

9.4. Visualizing the relationship between desired fold-change and minimum sample size number

```
designSampleSizePlots(data = samplesize)
```



10. Protein subject quantification

If there is no technical replicate, subject (or sample) quantification should be the same as run-level summarization. If there are technical replicates, subject-level summarization(quantification) with run-level summarization will be useful for downstream analysis, such as classification.

```
?quantification
```

```
sampleQuant <- quantification(quant.processed)
head(sampleQuant)
```

```
##           Protein Condition1_1 Condition2_2 Condition3_3
## 1  sp|D6VTK4|STE2_YEAST      26.60786      26.79430      26.53029
## 2  sp|013297|CET1_YEAST      24.71809      24.57865      24.62652
## 3  sp|013329|FOB1_YEAST      23.47075      23.43427      23.73741
## 4  sp|013539|THP2_YEAST      27.38510      25.90646      25.91799
## 5  sp|013547|CCW14_YEAST      27.11638      26.91302      26.75541
## 6  sp|013563|RPN13_YEAST      26.17056      26.01078      26.11412
## Condition4_4
## 1      26.60612
## 2      24.64886
## 3      23.16646
## 4      25.91781
## 5      26.98082
## 6      26.05415
```

11. msstats.log and sessionInfo.txt

These two files are important to keep the records of package versions and options in functions.

```

"R.version.3.3.2..2016.10.31."
"Platform: x86_64-apple-darwin13.4.0 (64-bit)"
"Running under: OS X El Capitan 10.11.6"
"locale:"
"[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8"
"attached base packages:"
"[1] stats      graphics  grDevices  utils      datasets  methods   base      "
"other attached packages:"
"[1] MSstats_3.7.2"
"loaded via a namespace (and not attached):"
" [1] Rcpp_0.12.7      magrittr_1.5      BiocGenerics_0.18.0"
" [4] splines_3.3.2    MASS_7.3-45       munsell_0.4.3     "
" [7] colorspace_1.2-6 lattice_0.20-34    minqa_1.2.4       "
"[10] stringr_1.1.0    plyr_1.8.4        caTools_1.17.1    "
"[13] tools_3.3.2      parallel_3.3.2    grid_3.3.2        "
"[16] Biobase_2.32.0   gtable_0.2.0      nlme_3.1-128      "
"[19] KernSmooth_2.23-15 marray_1.50.0     mzR_2.6.3         "
"[22] gtools_3.5.0     ProtGenerics_1.4.0 survival_2.39-5    "
"[25] lme4_1.1-12      lazyeval_0.2.0    assertthat_0.1    "
"[28] tibble_1.2       Matrix_1.2-7.1    reshape2_1.4.1    "
"[31] nloptr_1.0.4     ggplot2_2.2.0     bitops_1.0-6      "
"[34] codetools_0.2-15 ggrepel_0.5        stringi_1.1.1     "
"[37] limma_3.28.21    gdata_2.17.0      gplots_3.0.1      "
"[40] scales_0.4.1     minpack.lm_1.2-0  "
" "
" "
"MSstats - dataProcess function"
" "
"The required input : provided - okay"
summaryMethod : TMP
cutoffCensored : minFeature
censoredInt : 0
New input format : made new columns for analysis - okay
** There are 2585 intensities which are zero. These intensities are replaced with 1.
Logarithm transformation: log2 transformation is done - okay
fillIncompleteRows = TRUE
Balanced data format with NA for missing feature intensities - okay
Factorize in columns(GROUP, SUBJECT, GROUP_ORIGINAL, SUBJECT_ORIGINAL, SUBJECT_ORIGINAL_NESTED,
FEATURE, RUN) - okay
Normalization : Constant normalization (equalize medians) - okay
Between Run Interference Score is not calculated.
* Use all features that the dataset originally has.
1 level of Isotope type labeling in this experiment
Summary of Features :
# of Protein : 3027
# of Peptides/Protein : 2-196
# of Transitions/Peptide : 1-1

```

Figure 9: