qPLEXanalyzer

Matthew Eldridge, Kamal Kishore, Ashley Sawle

April 9, 2019

Contents

1 Overview

This document provides brief tutorial of the *qPLEXanalyzer* package, a toolkit with multiple functionalities, for statistical analysis of qPLEX-RIME proteomics data (see ?qPLEXanalyzer at the R prompt for a brief overview). The qPLEX-RIME approach combines the RIME method with multiplex TMT chemical isobaric labelling to study the dynamics of chromatin-associated protein complexes. The package can also be used for isobaric labelling (TMT or iTRAQ) based total proteome analysis.

- Import quantitative dataset: A pre-processed quantitative dataset generated from MaxQuant, Proteome Discoverer or any other proteomic software consisting of peptide intensities with associated features along with sample meta-data information can be imported by *qPLEXanalyzer*.
- Quality control: Computes and displays quality control statistics plots of the quantitative dataset.
- Data normalization: Quantile normalization, central tendencies scaling and linear regression based normalization.
- Aggregation of peptide intensities into protein intensities
- Differential statistical analysis: *limma* based analysis to identify differentially abundant proteins.

```
library(qPLEXanalyzer)
library(gridExtra)
data(human_anno)
data(exp2_Xlink)
```

2 Import quantitative dataset

MSnbase package by Laurent Gatto provides methods to facilitate reproducible analysis of MS-based proteomics data. *MSnSet* class of *MSnbase* provides architecture for storing quantitative MS proteomics data and the experimental meta-data. In *qPLEXanalyzer*, we store pre-processed quantitative proteomics data within this standardized object. The `convertToMSnset` function creates an *MSnSet* object from the quantitative dataset of peptides/protein intensities. This dataset must consist of peptides identified with high confidence in all the samples.

The default input dataset is the pre-processed peptide intensities from MaxQuant, Proteome Discoverer or any other proteomic software (see ?convertToMSnset at the R prompt for more details). Only peptides uniquely matching to a protein should be used as an input. Alternatively, the protein level quantification by the aggregation of the peptide TMT intensities can also be used as input. Peptides/Protein intensities with missing values in one or more samples can either be excluded or included in the *MSnSet* object. If the missing values are kept in the *MSnSet* object, these must be imputed either by user defined methods or by those provided in *MSnbase* package. The downstream functions of *qPLEXanalyzer* expects no missing values in the *MSnSet* object.

The example dataset shown below is from an ER qPLEX-RIME experiment in MCF7 cells that was performed to compare two different ways of cell crosslinking: DSG/formaldehyde (double) or formaldehyde alone (single). It consists of four biological replicates for each condition along with two IgG samples pooled from replicates of each group.

```
MSnset_data <- convertToMSnset(exp2_Xlink$intensities,
  metadata = exp2_Xlink$metadata,
  indExpData = c(7:16), Sequences = 2, Accessions = 6
)
```

3 Quality control

Once an *MSnSet* object has been created, various descriptive statistics methods can be used to check the quality of the dataset. The `intensityPlot` function generates a peptide intensity distribution plot that helps in identifying samples with outlier distributions. Figure 1 shows the distribution of the log-intensity of peptides/proteins for each sample. An outlier sample DSG.FA.rep01 can be identified from this plot. IgG control samples representing low background intensities will have shifted/distinct intensity distribution curve as compared to other samples and should not be considered as outliers.

```
intensityPlot(MSnset_data, title = "Peptide intensity distribution")
```

The intensities can also be viewed in the form of boxplots by `intensityPlot`. Figure 2 shows the distribution of peptides intensities for each sample. `rliPlot` can be used to visualise unwanted variation in a data set. It is similar to the relative log expression plot developed for microarray analysis - see Gandolfo and Speed (2018). Rather than examining gene expression, the RLI plot (Figure 3) uses the MS intensities for each peptide or the summarised protein intensities.

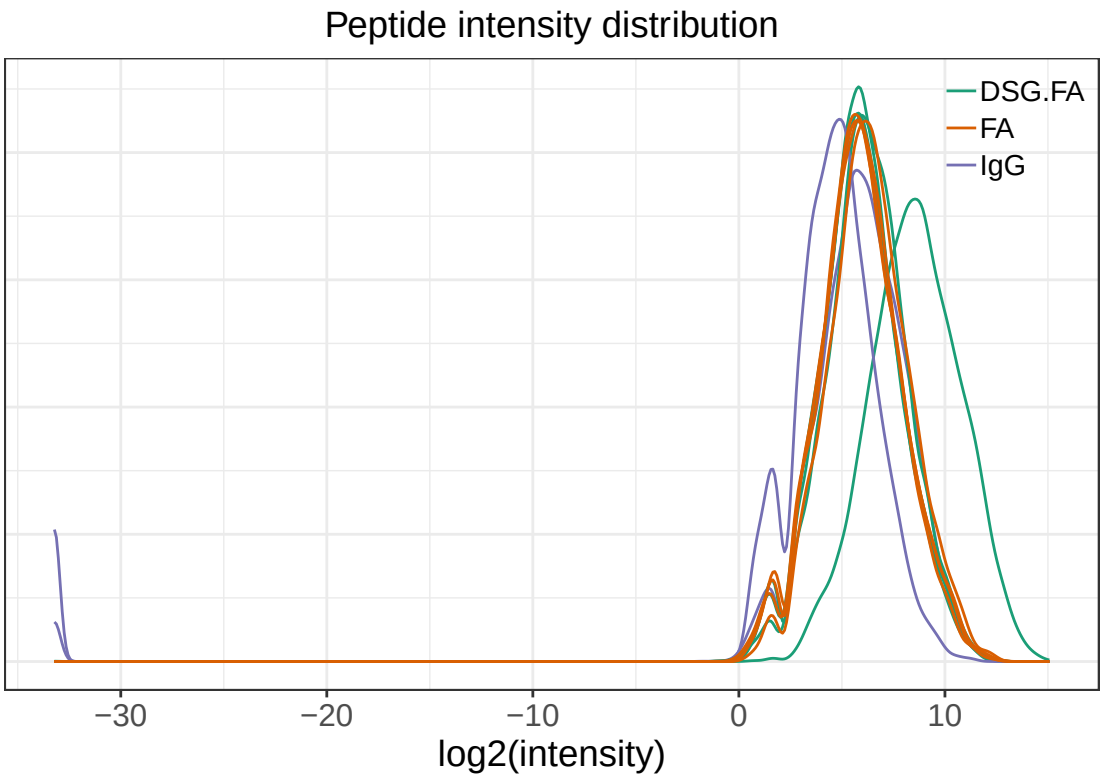

Figure 1: Density plots of raw intensities for TMT-10plex experiment.

```
intensityBoxplot(MSnset_data, title = "Peptide intensity distribution")
```

```
rliPlot(MSnset_data, title = "Relative Peptide intensity")
```

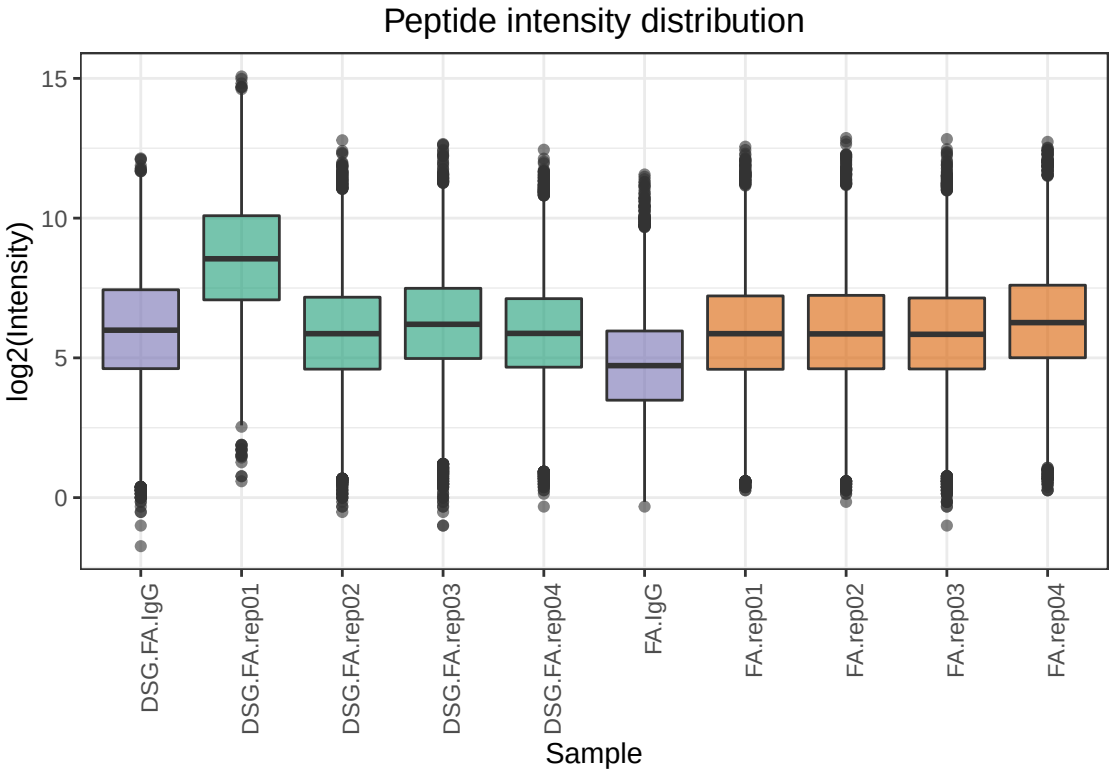

Figure 2: Boxplot of raw intensities for TMT-10plex experiment.

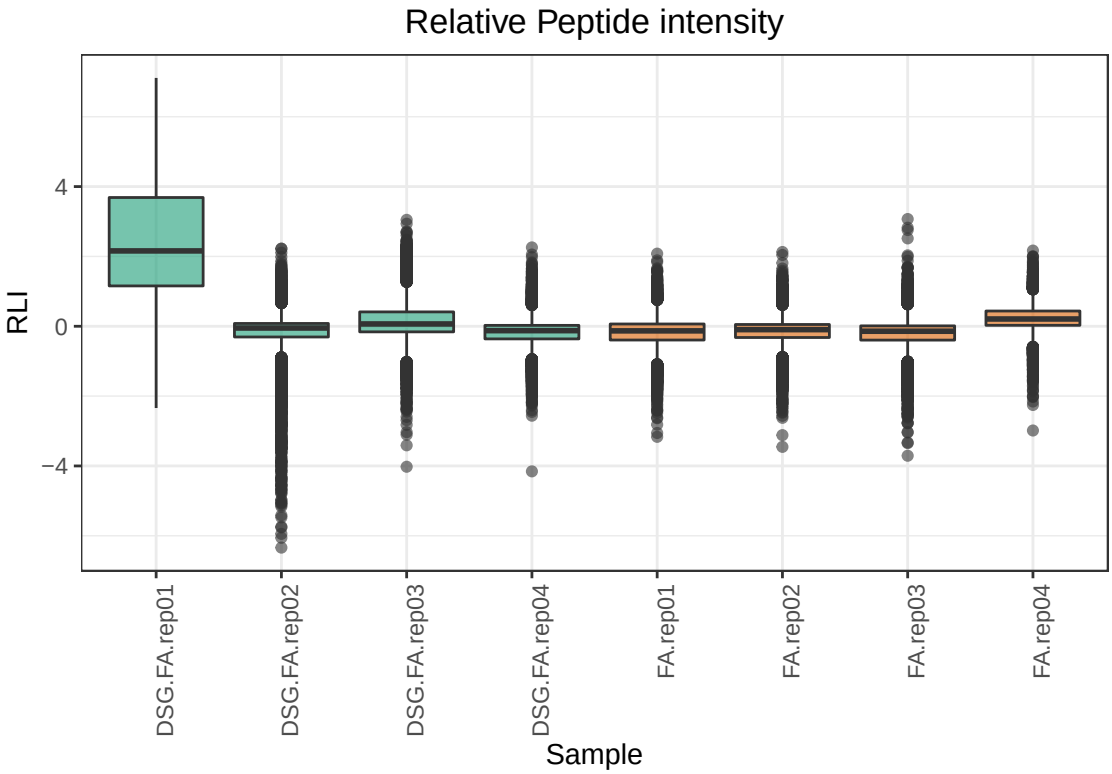

Figure 3: RLI of raw intensities for TMT-10plex experiment.

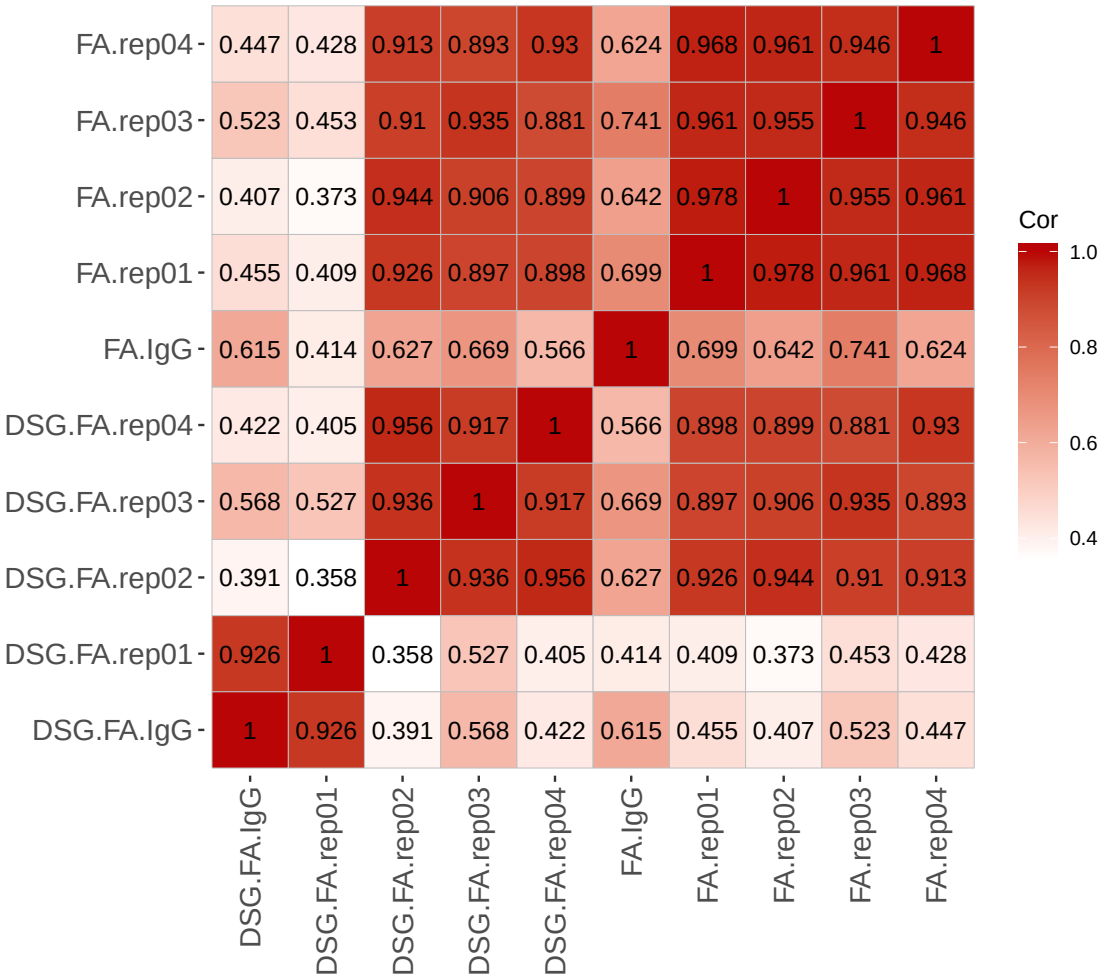

Figure 4: Correlation plot of peptide intensitites

A Correlation plot can be generated by `corrPlot` to visualize the level of linear association of samples within and between groups. The plot in Figure 4 displays high correlation among samples within each group, however an outlier sample is also identified in one of the groups (DSG.FA).

```
corrPlot(MSnset_data)
```

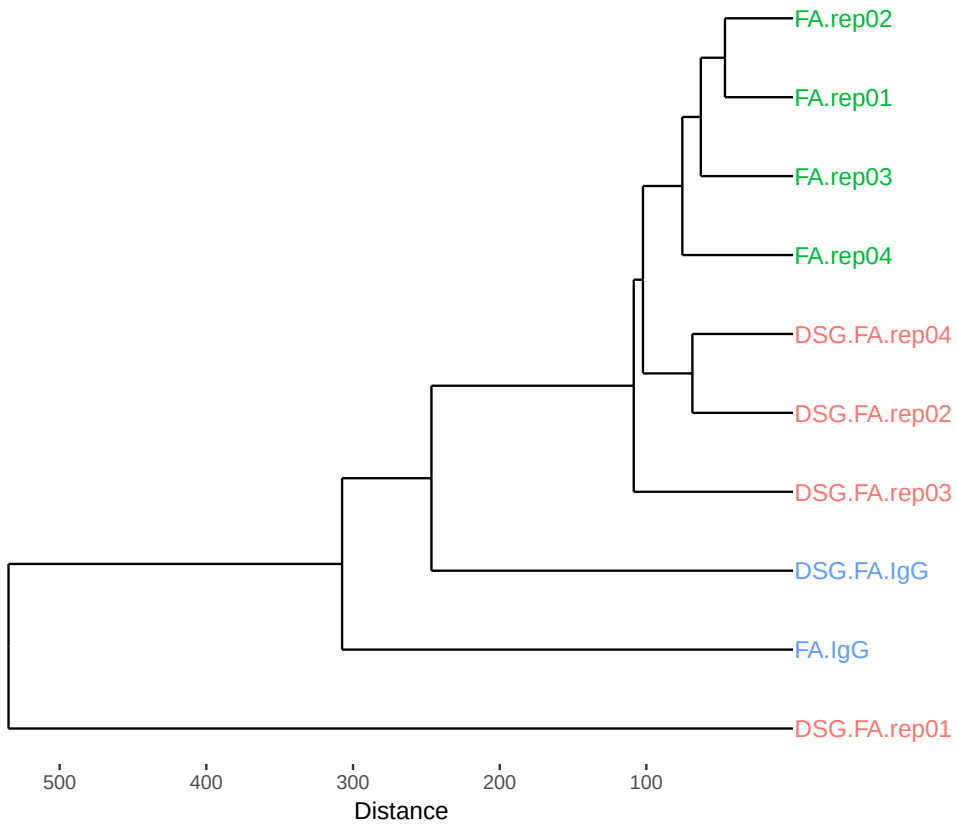

Figure 5: Clustering plot of peptide intensitites

Hierarchical clustering can be performed by `hierarchicalPlot` to produce a dendrogram displaying the hierarchical relationship among samples (Figure 5). The horizontal axis shows the dissimilarity (measured by means of the Euclidean distance) between samples: similar samples appear on the same branches. Colors correspond to groups. If the data set contains zeros, it will be necessary to add a small value (e.g. 0.01) to the intentsities in order to avoid errors while generating dendrogram.

```
exprs(MSnset_data) <- exprs(MSnset_data) + 0.01
hierarchicalPlot(MSnset_data)
```

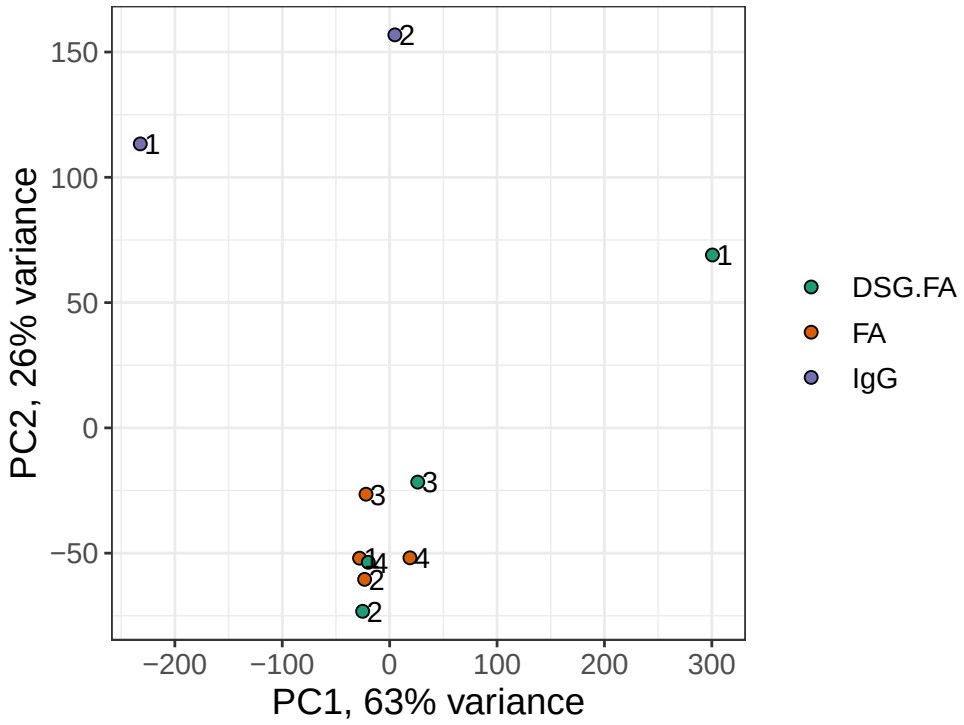

Figure 6: PCA plot of peptide intensitites

A visual representation of the scaled loading of the first two dimensions of a PCA analysis can be obtained by `pcaPlot` (Figure 6). Co-variances between samples are approximated by the inner product between samples. Highly correlated samples will appear close to each other. The samples could be labeled by name, replicate, group or experiment run allowing for identification of potential batch effects.

```
pcaPlot(MSnset_data, labelColumn = "BioRep", pointsize = 2)
```

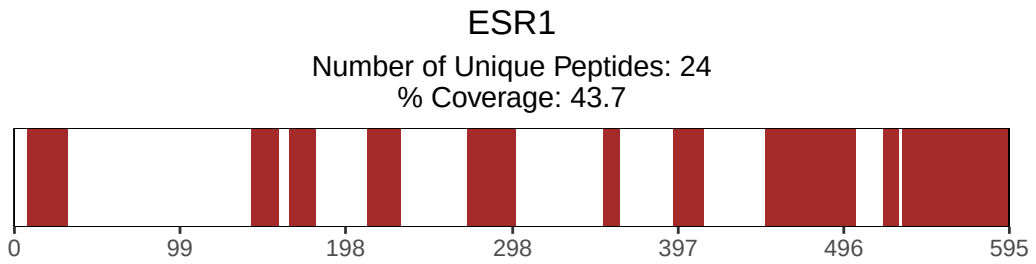

Figure 7: Peptide sequence coverage plot

A plot showing regions of the bait protein covered by captured peptides can be produced using `coveragePlot` (Figure 7). The plot shows the location of peptides that have been identified with high confidence across the protein sequence and the corresponding percentage of coverage. This provides a means of assessing the efficiency of the immunoprecipitation approach in the qPLEX-RIME method. For a better evaluation of the pull down assay we could compare the observed bait protein coverage with the theoretical coverage from peptides predicted by known cleavage sites.

```
mySequenceFile <- system.file("extdata", "P03372.fasta", package = "qPLEXanalyzer")
coveragePlot(MSnset_data,
  ProteinID = "P03372", ProteinName = "ESR1",
  fastaFile = mySequenceFile
)
```

4 Data normalization

The data can be normalized to remove experimental artifacts (e.g. differences in sample loading variability, systemic variation) in order to separate biological variations from those introduced during the experimental process. This would improve downstream statistical analysis to obtain more accurate comparisons. Different normalization methods can be used depending on the data:

- Quantiles: The peptide intensities are roughly replaced by the order statistics on their abundance. The key assumption underneath is that there are only few changes between different groups. This normalization technique has the effect of making the distributions of intensities from the different samples identical in terms of their statistical properties. It is the strongest normalization method and should be used carefully as it erases most of the difference between the samples. We would recommend using it only for total proteome but not for qPLEX-RIME data.
- Mean/median scaling: In this normalization method the central tendencies (mean or median) of the samples are aligned. The central tendency for each sample is computed and log transformed. A scaling factor is determined by subtracting from each central tendency the mean of all the central tendencies. The raw intensities are then divided by the scaling factor to get normalized ones.
- Row scaling: In this normalization method each peptide/protein intensity is divided by the mean/median of its intensity across all samples and log2 transformed.

It is imperative to check the intensity distribution plot and PCA plot before and after normalization to verify its effect on the dataset. In qPLEX-RIME data, the IgG (or control samples) should be normalized

separately from the bait protein pull-down samples. As IgG samples represent the low background intensity, their intensity distribution profile is different from bait pull-downs. Hence, normalizing the two together would result in over-correction of the IgG intensity resulting in inaccurate computation of differences among groups.

If no normalization is necessary, skip this step and move to aggregation of peptides.

For this dataset, an outlier sample was identified by quality control plots and removed from further analysis. Figure 8 displays the effect of various normalization methods on the peptide intensities distribution.

```
MSnset_data <- MSnset_data[, -5]
p1 <- intensityPlot(MSnset_data, title = "No normalization")

MSnset_norm_q <- normalizeQuantiles(MSnset_data)
p2 <- intensityPlot(MSnset_norm_q, title = "Quantile")

MSnset_norm_ns <- normalizeScaling(MSnset_data, scalingFunction = median)
p3 <- intensityPlot(MSnset_norm_ns, title = "Scaling")

MSnset_norm_gs <- groupScaling(MSnset_data,
                               scalingFunction = median,
                               groupingColumn = "SampleGroup")
p4 <- intensityPlot(MSnset_norm_gs, title = "WithinGrp Scaling")

grid.arrange(p1, p2, p3, p4, ncol = 2, nrow = 2)
```

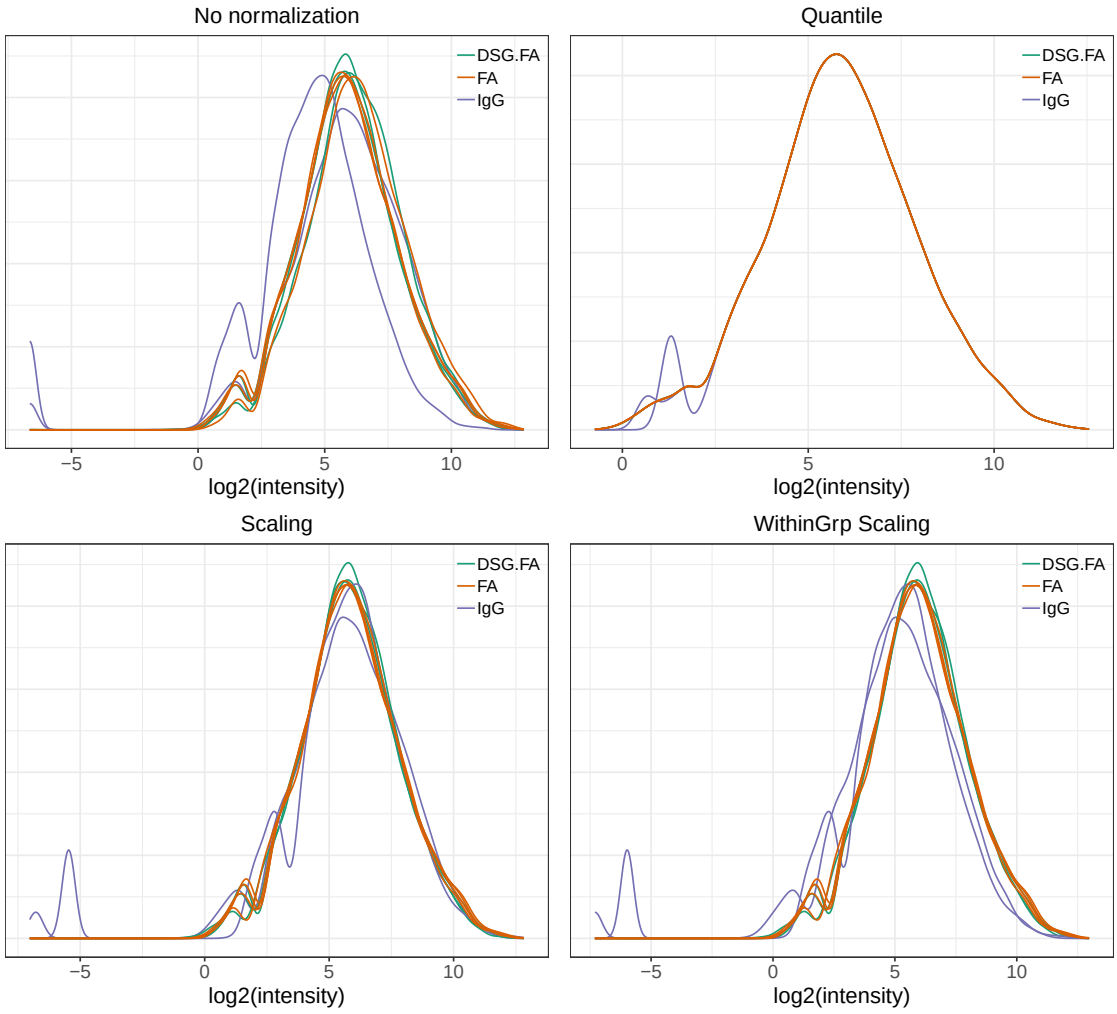

Figure 8: Peptide intensity distribution with various normalization methods

5 Aggregation of peptide intensities into protein intensities

The quantitative dataset could consist of peptide or protein intensities. If the dataset consists of peptide information, they can be aggregated to protein intensities for further analysis. For this, an annotation file consisting of proteins with unique ID must be provided. An example file can be found with the package corresponding to uniprot annotation of human proteins. It consists of four columns: 'Accessions', 'Gene', 'Description' and 'GeneSymbol'. The columns 'Accessions'and 'GeneSymbol' are mandatory for successful downstream analysis while the other two columns are optional. The *UniProt.ws* package provides a convenient means of obtaining these annotations using Uniprot protein accessions, as shown in the section below. The `summarizeIntensities` function expects an annotation file in this format.

```
library(UniProt.ws)
library(dplyr)
proteins <- unique(fData(MSnset_data)$Accessions)[1:10]
columns <- c("ENTRY-NAME", "PROTEIN-NAMES", "GENES")
hs <- UniProt.ws::UniProt.ws(taxId = 9606)
first_ten_anno <- UniProt.ws::select(hs, proteins, columns, "UNIPROTKB") %>%
  as_tibble() %>%
  mutate(GeneSymbol = gsub(" .*", "", GENES)) %>%
  select(
    Accessions = "UNIPROTKB", Gene = "ENTRY-NAME",
    Description = "PROTEIN-NAMES", GeneSymbol
  )
head(arrange(first_ten_anno, Accessions))
```

```
## # A tibble: 6 x 4
##   Accessions Gene     Description                                GeneSymbol
##   <chr>      <chr>    <chr>                                      <chr>
## 1 P04264     K2C1_HU~ Keratin, type II cytoskeletal 1 OS=Hom~ KRT1
## 2 P05783     K1C18_H~ Keratin, type I cytoskeletal 18 OS=Hom~ KRT18
## 3 P14866     HNRPL_H~ Heterogeneous nuclear ribonucleoprotei~ HNRNPL
## 4 P35527     K1C9_HU~ Keratin, type I cytoskeletal 9 OS=Homo~ KRT9
## 5 P35908     K22E_HU~ Keratin, type II cytoskeletal 2 epider~ KRT2
## 6 P39748     FEN1_HU~ Flap endonuclease 1 OS=Homo sapiens OX~ FEN1
```

The aggregation can be performed by calculating the sum, mean or median of the raw or normalized peptide intensities. The summarized intensity for a selected protein could be visualized using `peptideIntensityPlot`. It plots all peptides intensities for a selected protein along with summarized intensity across all the samples (Figure 9).

```
MSnset_Pnorm <- summarizeIntensities(MSnset_norm_gs, sum, human_anno)
```

```
peptideIntensityPlot(MSnset_data,
  combinedIntensities = MSnset_Pnorm,
  ProteinID = "P03372",
  ProteinName = "ESR1"
)
```

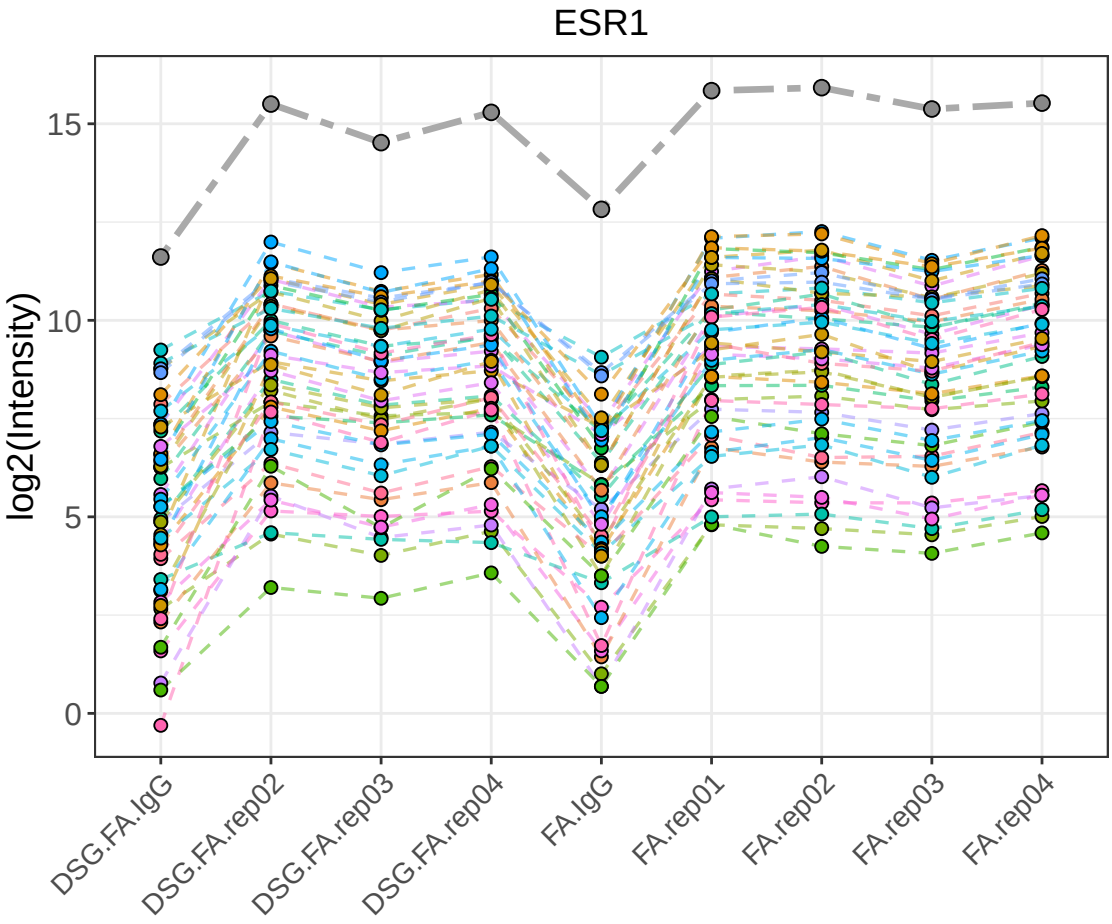

Figure 9: Summarized protein intensity

6 Regression Analysis

To correct for the potential dependency of immunoprecipitated proteins (in qPLEX-RIME) on the bait protein, a linear regression method is available in *qPLEXanalyzer*. The `regressIntensity` function performs a regression analysis in which bait protein levels is the independent variable (x) and the profile of each of the other protein is the dependent variable (y). The residuals of the y=ax+b linear model represent the protein quantification profiles that are not driven by the amount of the bait protein.

The advantage of this approach is that proteins with strong dependency on the target protein are subjected to significant correction, whereas proteins with small dependency on the target protein are slightly corrected. In contrast, if a standard correction factor were used, it would have the same magnitude of effect on all proteins. The control samples (such as IgG) should be excluded from the regression analysis. The `regressIntensity` function also generates the plot displaying the correlation between bait and other protein before and after applying this method (Figure 10).

The example dataset shown below is from an ER qPLEX-RIME experiment carried out in MCF7 cells to investigate the dynamics of the ER complex assembly upon 4-hydroxytamoxifen (OHT) treatment at 2h, 6h and 24h or at 24h post-treatment with the vehicle alone (ethanol). It consists of six biological replicates for each condition spanned across three TMT experiments along with two IgG mock pull down samples in each experiment.

```
data(exp3_OHT_ESR1)
MSnset_reg <- convertToMSnset(exp3_OHT_ESR1$intensities_qPLEX2,
  metadata = exp3_OHT_ESR1$metadata_qPLEX2,
  indExpData = c(7:16), Sequences = 2, Accessions = 6
)
MSnset_P <- summarizeIntensities(MSnset_reg, sum, human_anno)
MSnset_P <- rowScaling(MSnset_P, mean)
IgG_ind <- which(pData(MSnset_P)$SampleGroup == "IgG")
Reg_data <- regressIntensity(MSnset_P,
                             controlInd = IgG_ind,
                             ProteinId = "P03372")
```

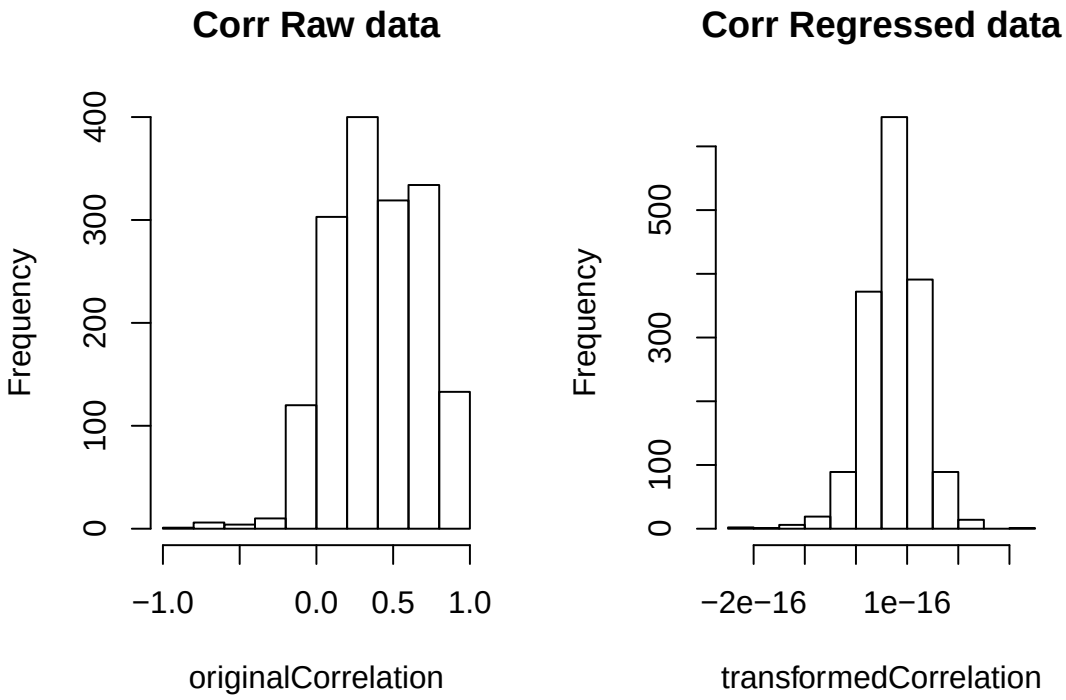

Figure 10: Correlation between bait protein and enriched proteins before and after regression

7 Differential statistical analysis

A statistical analysis for the identification of differentially regulated or bound proteins is carried out using *limma* based analysis. It uses linear models to assess differential expression in the context of multifactor designed experiments. Firstly, a linear model is fitted for each protein where the model includes variables for each group and MS run. Then, log2 fold changes between comparisons are estimated using `computeDiffStats`. Multiple testing correction of p-values are applied using the Benjamini-Hochberg method to control the false discovery rate (FDR). Finally, `getContrastResults` is used to get contrast specific results.

The qPLEX-RIME experiment can consist of IgG mock samples to discriminate non-specific binding. The controlGroup argument within `getContrastResults` function allows you to specify this group (such as IgG). It then uses the mean intensities from the fitted linear model to compute log2 fold change between IgG and each of the groups. The maximum log2 fold change over IgG control from the two groups being compared is reported in the controlLogFoldChange column. This information can be used to filter non-specific binding. A controlLogFoldChange more than 1 can be used as a filter to discover specific interactors.

The results of the differential protein analysis can be visualized using `maVolPlot` function. It plots average log2 protein intensity to log2 fold change between groups compared. This enables quick visualization (Figure 11) of significantly abundant proteins between groups. `maVolPlot` could also be used to view differential protein results in a volcano plot (Figure 12) to compare the size of the fold change to the statistical significance level.

```
contrasts <- c(DSG.FA_vs_FA = "DSG.FA - FA")
diffstats <- computeDiffStats(MSnset_Pnorm, contrasts = contrasts)
```

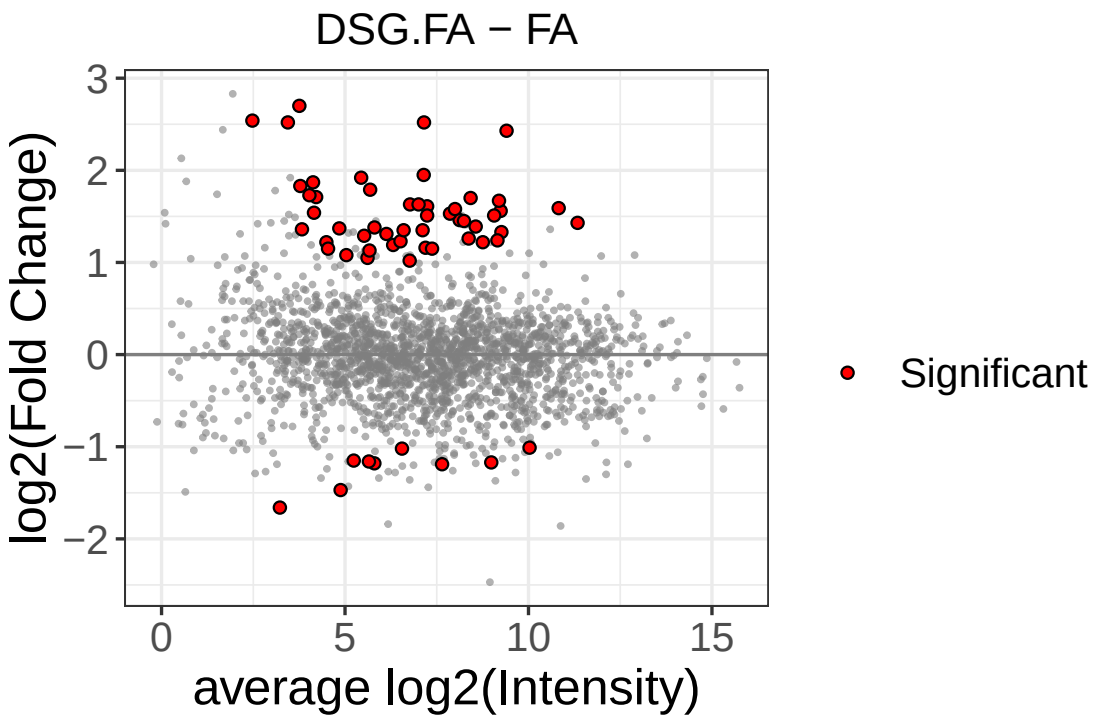

Figure 11: MA plot of the quantified proteins

```
diffexp <- getContrastResults(
  diffstats = diffstats,
  contrast = contrasts,
  controlGroup = "IgG"
)
maVolPlot(diffstats, contrast = contrasts, plotType = "MA", title = contrasts)
```

```
maVolPlot(diffstats,
          contrast = contrasts,
          plotType = "Volcano",
          title = contrasts)
```

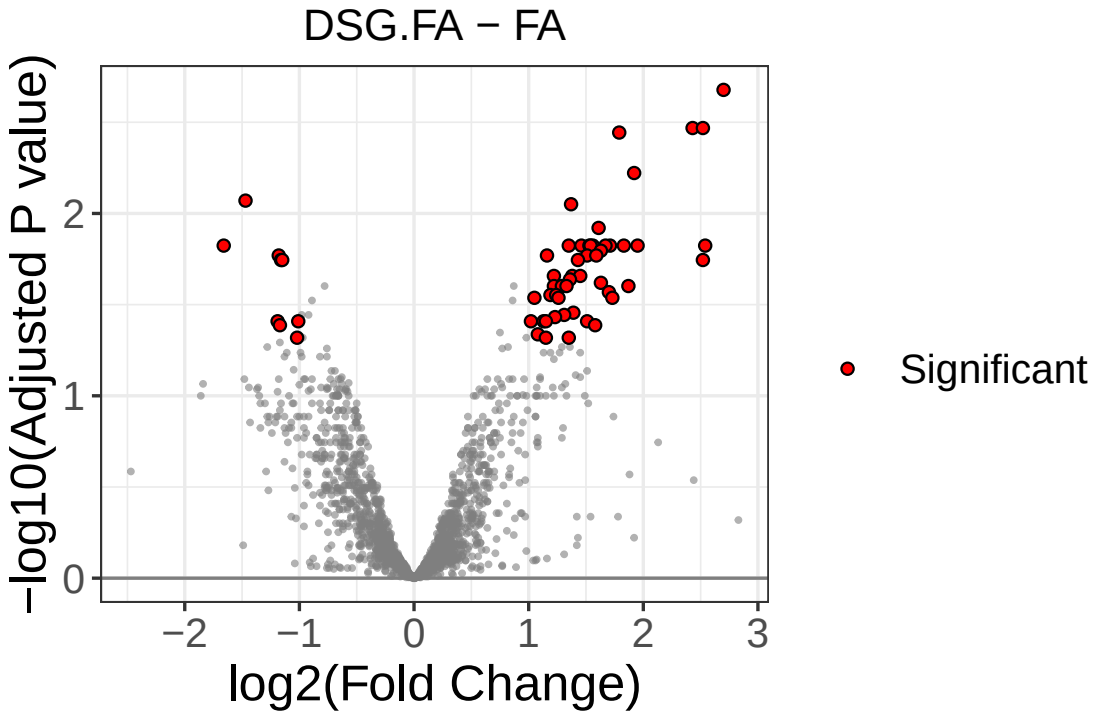

Figure 12: Volcano plot of the quantified proteins

8 Session Information

```
sessionInfo()

## R version 3.5.3 (2019-03-11)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2012 R2 x64 (build 9600)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=C
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils
## [7] datasets  methods   base
##
## other attached packages:
##  [1] dplyr_0.8.0.1       gridExtra_2.3       qPLEXanalyzer_1.0.4
##  [4] MSnbase_2.8.3       ProtGenerics_1.14.0 S4Vectors_0.20.1
##  [7] mzR_2.16.2          Rcpp_1.0.1          statmod_1.4.30
## [10] Biobase_2.42.0      BiocGenerics_0.28.0
```

```
##
## loaded via a namespace (and not attached):
##  [1] lattice_0.20-38        tidyr_0.8.3
##  [3] Biostrings_2.50.2      utf8_1.1.4
##  [5] assertthat_0.2.1       digest_0.6.18
##  [7] foreach_1.4.4          R6_2.4.0
##  [9] GenomeInfoDb_1.18.2    plyr_1.8.4
## [11] mzID_1.20.1            evaluate_0.13
## [13] highr_0.8              ggplot2_3.1.1
## [15] pillar_1.3.1           zlibbioc_1.28.0
## [17] rlang_0.3.4            lazyeval_0.2.2
## [19] preprocessCore_1.44.0  splines_3.5.3
## [21] labeling_0.3           BiocParallel_1.16.6
## [23] stringr_1.4.0          RCurl_1.95-4.12
## [25] munsell_0.5.0          compiler_3.5.3
## [27] xfun_0.6               pkgconfig_2.0.2
## [29] pcaMethods_1.74.0      tidyselect_0.2.5
## [31] tibble_2.1.1           GenomeInfoDbData_1.2.0
## [33] IRanges_2.16.0         codetools_0.2-16
## [35] XML_3.98-1.19          fansi_0.4.0
## [37] crayon_1.3.4           MASS_7.3-51.3
## [39] bitops_1.0-6           grid_3.5.3
## [41] gtable_0.3.0           affy_1.60.0
## [43] magrittr_1.5           scales_1.0.0
## [45] ncdf4_1.16.1           cli_1.1.0
## [47] stringi_1.4.3          impute_1.56.0
## [49] XVector_0.22.0         affyio_1.52.0
## [51] doParallel_1.0.14      limma_3.38.3
## [53] ggdendro_0.1-20        RColorBrewer_1.1-2
## [55] iterators_1.0.10       tools_3.5.3
## [57] glue_1.3.1             purrr_0.3.2
## [59] colorspace_1.4-1       BiocManager_1.30.4
## [61] vsn_3.50.0             GenomicRanges_1.34.0
## [63] MALDIquant_1.19.2      knitr_1.22
```