MSImpute User's Guide

Soroor Hediyeh-zadeh

August 31, 2020

Contents

1 Installation

Install from GitHub

```
> install.packages("devtools") # devtools is required to download and install the package
> devtools::install_github("DavisLaboratory/msImpute")
```

Set up a python environment in R

The python environment is only required if you're willing to compute `gw_dist`, which is a distributional discrepancy measure (this is explained more in Section 4.5). The environment is set up using the `reticulate` package.

```
> install.packages("reticulate")
> library(reticulate)
```

Create a virtual environment

```
> virtualenv_create('msImpute-reticulate')
> py_available() # if this returns TRUE, you've access to python from R.
```

you can additionally run `py_config()` to find out what python version you (or the virtual environment you've set-up) are using. See `reticulate` package if you need to troubleshoot.

Install `scipy`, `cython` and `POT` python packages in this virtual environment

```
> virtualenv_install("msImpute-reticulate","scipy")
> virtualenv_install("msImpute-reticulate","cython")
> virtualenv_install("msImpute-reticulate","POT")
```

If this runs successfully, the installations have been successful:

```
> scipy <- import("scipy")
```

You can now run the `computeStructuralMetrics()` function to compute GW distance. This setup should only be done for the first use. For all subsequent usages, load the virtual environment that you've created using:

```
> library(reticulate)
> use_virtualenv("msImpute-reticulate")
```

you can then run the `computeStructuralMetrics()` function. Note that the `reticulate` package should be loaded before loading *msImpute*.

2 Quick Start

The package consists of the following main functions:

`selectFeatures`: identifies peptides with high biological dropout rate. These peptides have a high abundance and high dropout. Their missingness pattern is used as a diagnostic for MAR/MNAR missingness type, which can then inform later decisions around imputation of the data.

```
> library(reticulate)
> library(msImpute)
> use_virtualenv("msImpute-reticulate")

> selectFeatures(xna)
```

`scaleData`: *msImpute* first scales the data before training a low-rank model

```
> xna <- scaleData(xna)
```

`msImpute`: Main function that imputes missing values by learning a low-rank approximation of the data

```
> xcomplete <- msImpute(xna)
```

`findVariableFeatures`: finds peptides with high biological variance. We use this in `computeStructuralMetrics()`

```
> top.hvp <- findVariableFeatures(xna$E)
```

`computeStructuralMetrics`: returns a number of metrics that measure distortions into the data after imputation.

```
> computeStructuralMetrics(xcomplete,
+                          group,
+                          xna$E[rownames(top.hvp)[1:50],],
+                          k = 2)
```

These functions overall are designed to inform user's decision on adopting an imputation strategy. Datasets used in the case studies are pre-processed according to the code provided at the beginning part of each case study. Raw data are available from [1] as `rds` files.

3 TIMS Case Study: Blood plasma

The aim is to assess the missing values pattern in ion mobility data by Prianichnikov et al. (2020), available from PXD014777. The `evidence` table of MaxQuant output was processed as described below. Rows are Modified Peptide IDs. Charge state variations are treated as distinct peptide species. For peptides with multiple identification types, the intensity is considered to be the median of reported intensity values. Reverse complements and contaminant peptides are discarded. Peptides with more than 4 observed intensity values are retained.

The data was acquired in two batches (over two days). We are interested to know if missing values are evenly distributed across batches, or there is a batch-specific dropout trend. The runs are also labeled by S1, S2 and S4 (source unknown). The aim is to use this information to work out if missing values occur due to technical or biological effects.

```
> library(limma)
> library(tidyr)
> library(imputeLCMD)
> library(impute)
> library(ComplexHeatmap)
```

3.1 Data processing

The following procedures were applied to process the data, which we later load from the package data.

3.1.1 Filter by detection

```
> # when processing from raw files, please replace "path" with the directory
> # where the proteomicscasestudies github repo is downloaded.
>
> # PXD014777_evidence <- readRDS("path/proteomicscasestudies/PXD014777_evidence.rds")
>
>
> # remove contaminants
```

[1] `https://github.com/soroorh/proteomicscasestudies`

```
> table(grepl("CON__|REV__", PXD014777_evidence$Leading.razor.protein))
> PXD014777_evidence <- PXD014777_evidence[grep("CON__|REV__",
+                                               PXD014777_evidence$Leading.razor.protein,
+                                               invert=TRUE),]
> PXD014777_evidence$PeptideID <- paste(PXD014777_evidence$Mod..peptide.ID,
+                                       PXD014777_evidence$Charge, sep="_")
> # multiple identifications types in a experiment, taken median of the values
> #table(PXD014777_evidence$Type[PXD014777_evidence$PeptideID=="0_2"])
>
> y <- aggregate(Intensity ~ Raw.file + PeptideID,
+                FUN = function(x) median(x, na.rm = TRUE),
+                na.action = na.pass, data = PXD014777_evidence)
> # reshape into the form of an intensity matrix
> y <- spread(y, key = Raw.file, value = Intensity)
> rownames(y) <- y[,1]
> y <- y[,-1]
> dim(y)
> # remove peptides with less than 4 observation
> keep <- (rowSums(!is.na(y)) >= 4)
> table(keep)
> y<- y[keep,]
```

The processed data can be accessed via `data(pxd014777)`

```
> data(pxd014777)
> y <- pxd014777
```

Zero values that will be converted to Inf/-Inf after log- transformation. Check if there are valid values in the data before log transformation

```
> table(is.infinite(data.matrix(log2(y))))

 FALSE   TRUE
610515    681
```

There are zero values that will be converted to Inf/-Inf after log- transformation. Add a small offset to avoid infinite values:

```
> y <- log2(y+0.25)
>
> # can explore the data by boxplots
> #boxplot(y , las = 2, outline = FALSE)
```

Before we normalise the data, we explore the mean-CV^2 trend to determine if a intensity-based filtering step is required. Some outlier points are detected on mean-CV^2 plot:

```
> peptideCV2 <- plotCV2(y, trend = FALSE)
```

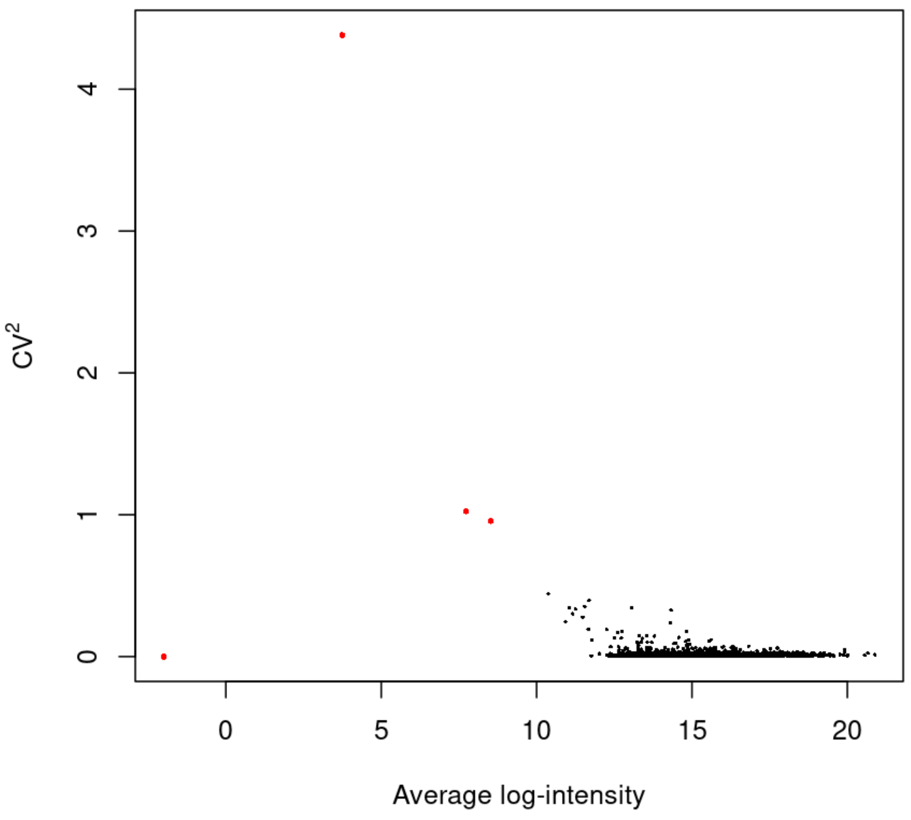


Figure 1: Mean-CV^2 plot

The outlier points tend to have low abundances, and are likely to be false peaks detected in the data as the result of incorrect match between runs. An incorrect match between run typically results in high coefficient of variation (CV) at low intensities, which can be spotted using mean-CV^2 plot. Such peptides have high CV as intensities reported by match between run are variable and typically less consistent with intensities measured by multi MS/MS detection type. This can be confirmed by looking-up the detection type of intensities reported for the peptides highlighted as outlier. The outliers are also reported by `plotCV2` output.

```
> head(peptideCV2[order(peptideCV2$CV, decreasing = TRUE),])

            mean        CV          RBF outlier
561_2   3.740281 4.3812055 0.0003468613    TRUE
370_2   7.734421 1.0247445 0.0006149430    TRUE
```

```
833_2    8.520133 0.9559445 0.0007170983    TRUE
153_2   10.375527 0.4437493 0.0071224722   FALSE
1276_2 11.689774 0.3944568 0.0768031933   FALSE
731_2   11.550239 0.3532837 0.0628256772   FALSE
```

3.1.2 Filter by intensity and normalize

We apply an **intensity-based filtering** in addition to filter by detection applied in the previous step. The mean-CV^2 trend after removing outlier points (peptides) is shown below:

```
> # quantile normalisation
> A <- rowMeans(y, na.rm = TRUE)
> y <- normalizeBetweenArrays(y[A>10,], method = "quantile")
> peptideCV2 <- plotCV2(y)
>
```

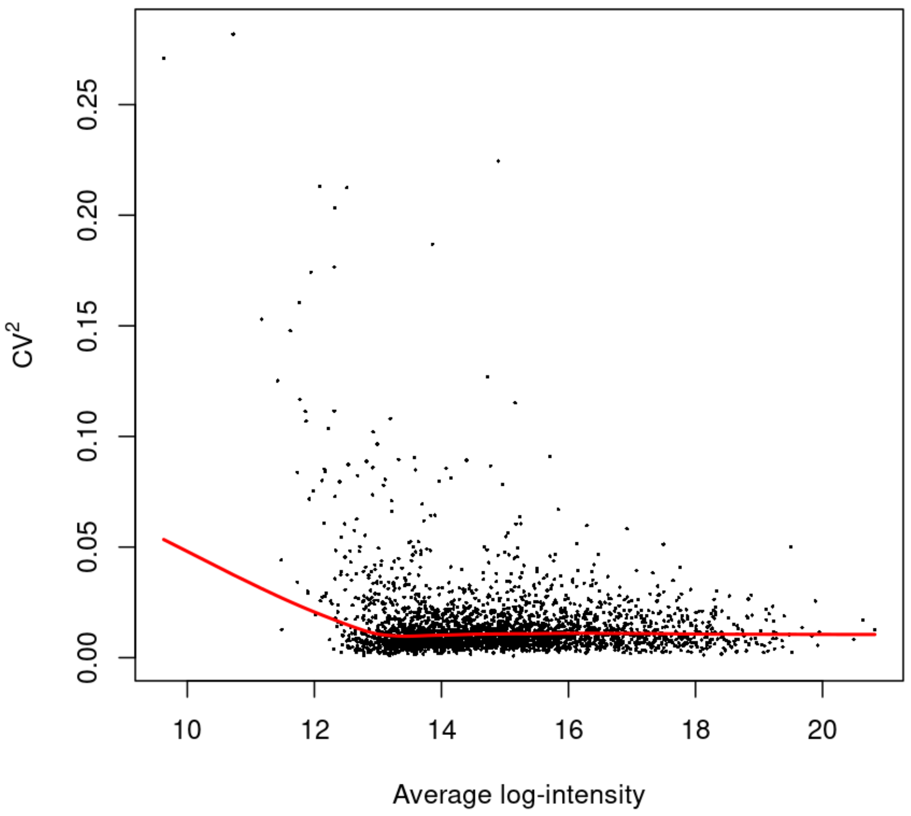
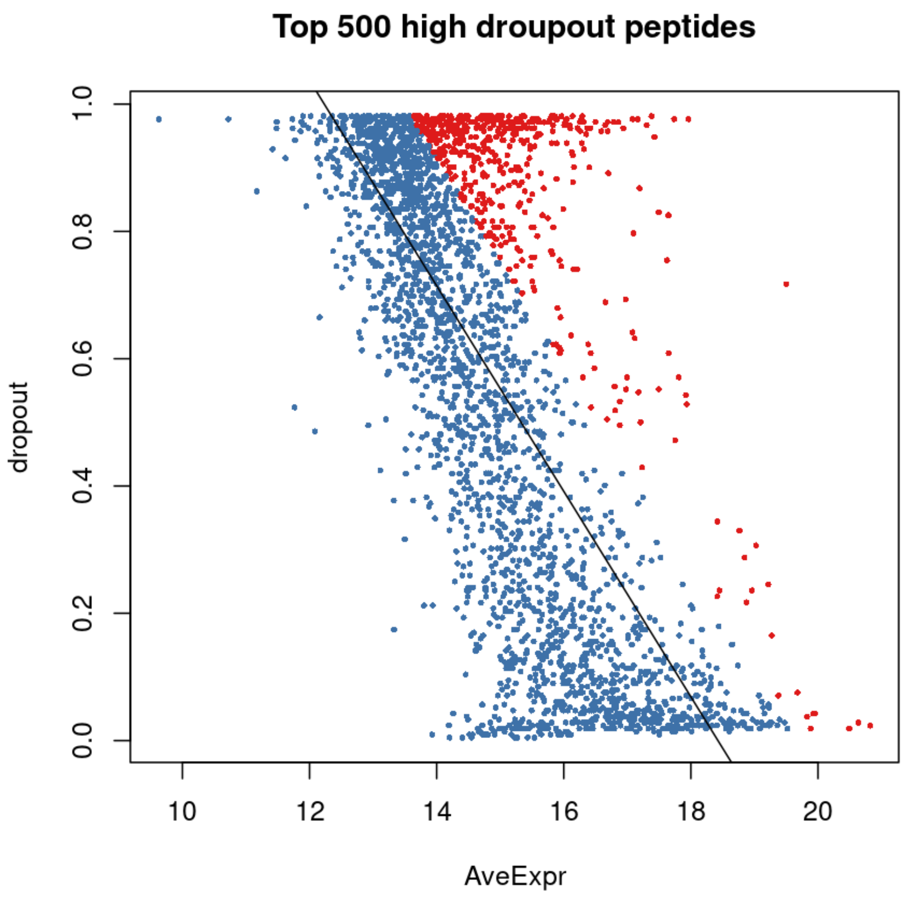


Figure 2: Mean-CV^2 plot after intensity-based filtering

3.2 Determine missing values pattern

Determine dominant patterns of missing values by investigating the dropout pattern of high dropout peptides. We find top 500 peptides with higher than expected dropout rate and high average log-intensity, then make a heatmap of their dropout pattern.

```
> hdp <- selectFeatures(y, n_features = 500)
> # construct matrix M to capture missing entries
> M <- ifelse(is.na(y),1,0)
> M <- M[hdp$msImpute_feature,]
> # plot a heatmap of missingness patterns for the selected peptides
>
>
> batch <- as.factor(gsub("(2018.*)_RF.*","\\1", colnames(y)))
> experiment <- as.factor(gsub(".*(S[1-9]).*","\\1", colnames(y)))
```



```
> ha_column <- HeatmapAnnotation(batch = batch,
+                                experiment = experiment,
+                                 col = list(batch = c('20181023' = "#B24745FF",
+                                                      '20181024'= "#00A1D5FF"),
+                                            experiment=c("S1"="#DF8F44FF",
+                                                         "S2"="#374E55FF",
+                                                         "S4"="#79AF97FF")))
> hm <- Heatmap(M,
+ column_title = "dropout pattern, columns ordered by dropout similarity",
+               name = "Intensity",
+               col = c("#8FBC8F", "#FFEFDB"),
+               show_row_names = FALSE,
+               show_column_names = FALSE,
+               cluster_rows = TRUE,
+               cluster_columns = TRUE,
+               show_column_dend = FALSE,
+               show_row_dend = FALSE,
+               top_annotation = ha_column,
+               row_names_gp =  gpar(fontsize = 7),
+               column_names_gp = gpar(fontsize = 8),
+               heatmap_legend_param = list(#direction = "horizontal",
+               heatmap_legend_side = "bottom",
+               labels = c("observed","missing"),
+               legend_width = unit(6, "cm")),
+          )
> hm <- draw(hm, heatmap_legend_side = "left")
```

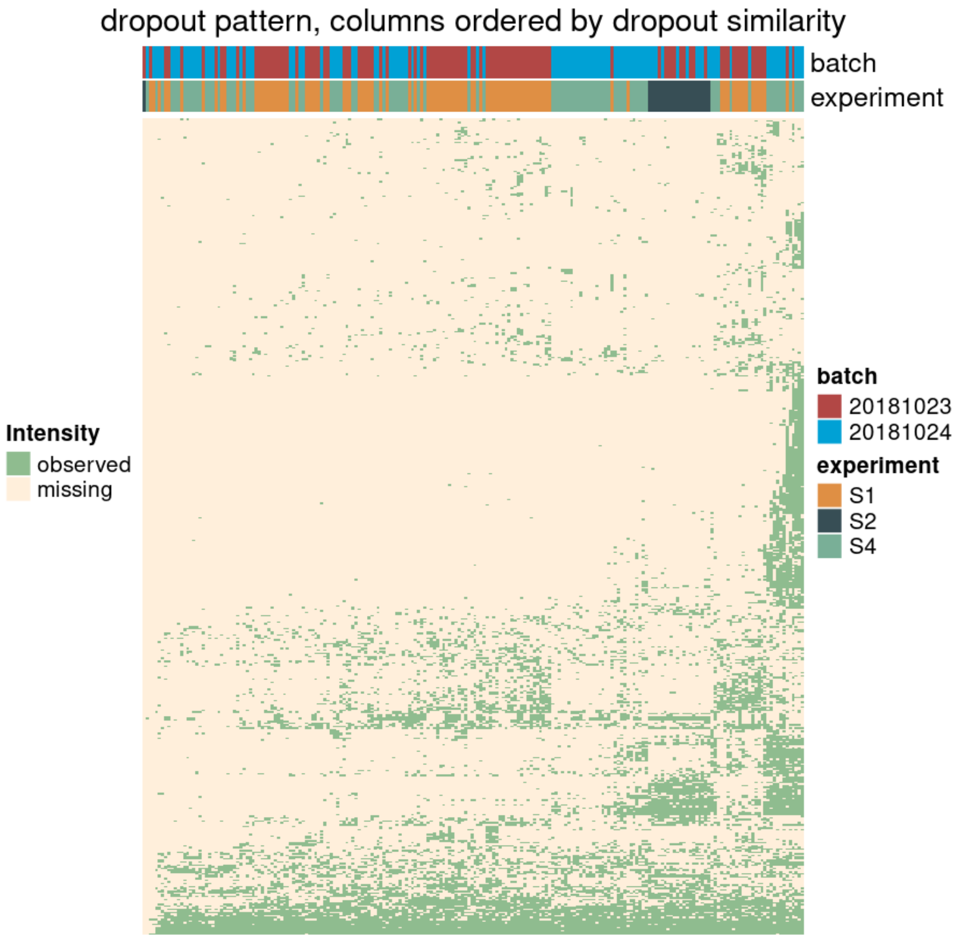


Figure 3: Heatmap of missing value patterns for top 500 high dropout peptides

As it can be seen on the heatmap, missing values are evenly distributed over batches (there is no systematic clustering of samples by batch), however, samples labeled as `S2` in the file name tend to cluster together. It turns out that samples from `S1` and `S4` experiments are blood plasma from individuals, whereas samples labeled as `S2` are labeled as pooled plasma. This is a biological effect, and we just demonstrated that `msImpute` can detect biologically meaningful and informative results.

4 DDA Case Study: Extracellular vesicles isolated from inflammatory bowel disease patients and controls

The study aims to characterize the proteomic profile of extracellular vesicles isolated from the descending colon of pediatric patients with inflammatory bowel disease and control participants.

The following analysis is based on the ‘peptide‘ table from MaxQuant output, available from PXD007959. Rows are Modified Peptide IDs. Charge state variations are treated as distinct peptide species. Reverse complements and contaminant peptides are discarded. Peptides with more than 4 observed intensity values are retained. Additionally, qualified peptides are required to map uniquely to proteins. Two of the samples with missing group annotation were excluded.

4.1 Filter by detection

The following procedures were applied to process the data, which we later load from the package data.

```
> # sample_annot <- readRDS("path/proteomicscasestudies/PXD007959_experimentalDesignTemplate.rds")
> # PXD007959_peptide <- readRDS("path/proteomicscasestudies/PXD007959_peptide.rds")
>
>
> sample_annot$group <- gsub(".*_(Mild|Sever|Control|Moderate)",
+                            "\\1", sample_annot$Experiment)
> sample_annot$group[grep("dCA", sample_annot$group)] <- NA
> table(grepl("CON__|REV__", PXD007959_peptide$Leading.razor.protein))
> table(PXD007959_peptide$Unique..Proteins.)
> # keep non-contaminants and peptides that map to unique proteins
> keep1 <- (!grepl("CON__|REV__", PXD007959_peptide$Leading.razor.protein))
> keep2 <- (PXD007959_peptide$Unique..Proteins. == "yes")
> PXD007959_peptide <- PXD007959_peptide[keep1&keep2, ]
> dim(PXD007959_peptide)
> dim(sample_annot)
> # replace zeros with NAs
> y <- PXD007959_peptide[,grep("Intensity\\.", colnames(PXD007959_peptide))]
> rownames(y) <- paste(PXD007959_peptide$Mod..peptide.IDs,
+                      PXD007959_peptide$Charges, sep="_")
> y[y==0] <- NA
> # remove samples with no group annotation
> y <- y[, !is.na(sample_annot$group)]
> sample_annot <- sample_annot[!is.na(sample_annot$group),]
> table(rowSums(!is.na(y)) >= 4)
> keep3 <- (rowSums(!is.na(y)) >= 4)
> y <- y[keep3,]
```

4.2 Normalization

The sample descriptions can be accessed via the `samples` component and the intensity values can be accessed from the `y` component of `pxd007959`.

```
> data(pxd007959)
> sample_annot <- pxd007959$samples
> y <- pxd007959$y
> y <- log2(y)
```

Here we explore the mean-CV^2 trend to determine if intensity-based filtering is required.

```
> # explore mean-CV^2 trend
> peptideCV2 <- plotCV2(y)
```

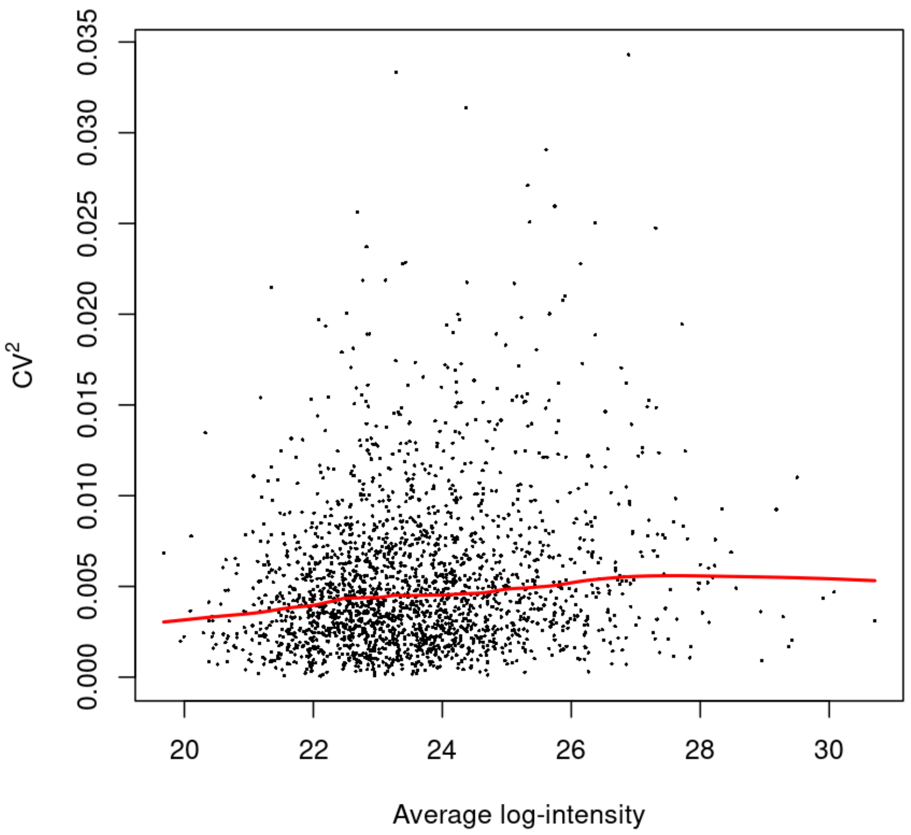


Figure 4: Mean-CV^2 plot

No outlier points are detected, hence, no intensity-based filtering is necessary.
To help us determine the best normalization method, we use `MA plots` in a diagnostic approach. For each sample (run) in the study, log fold-change of peptides in that versus all other samples (`M values`) in plotted against peptide average log-intensity (`A values`). The blue line represent median log-intensity for the peptides, and the red line is a loess fit to `M` and `A` values. One can see that in some samples, there is a (mean-) trended bias: M values are higher (or lower) for high abundance peptides. This indicates a composition bias. We apply *cyclic loess* normalisation from `limma` to correct for this trended bias.

```
> A <- rowMeans(y, na.rm = TRUE)
> o <- order(A)
> par(mfrow=c(4,4))
> for(i in 1:ncol(y)){
+   limma::plotMA(y, array = i, ylab="M", x="A", cex=0.6)
+   M <- y[,i]-rowMeans(y[,-i], na.rm = TRUE)
+
+   abline(h=median(M, na.rm = TRUE), col = "blue")
+   fit <- loessFit(x=A, y=M)
+   lines(A[o], fit$fitted[o], col="red", lwd=2)
+ }
>
```

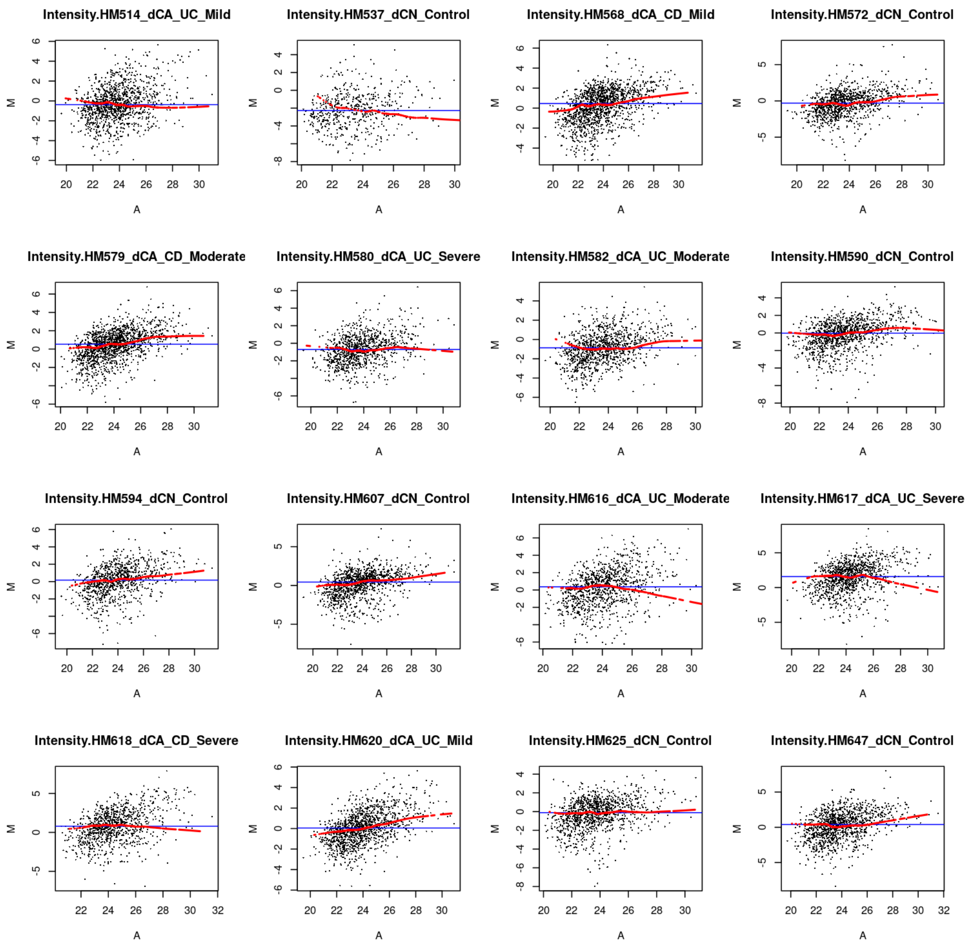


Figure 5: MA plots before normalisation

```
> y <- normalizeBetweenArrays(y, method = "cyclicloess")
> A <- rowMeans(y, na.rm = TRUE)
> o <- order(A)
> par(mfrow=c(4,4))
> for(i in 1:ncol(y)){
+   limma::plotMA(y, array = i, ylab="M", x="A", cex=0.6)
+   M <- y[,i]-rowMeans(y[,-i], na.rm = TRUE)
+
+   abline(h=median(M, na.rm = TRUE), col = "blue")
+   fit <- loessFit(x=A, y=M)
+   lines(A[o], fit$fitted[o], col="red", lwd = 2)
+ }
```

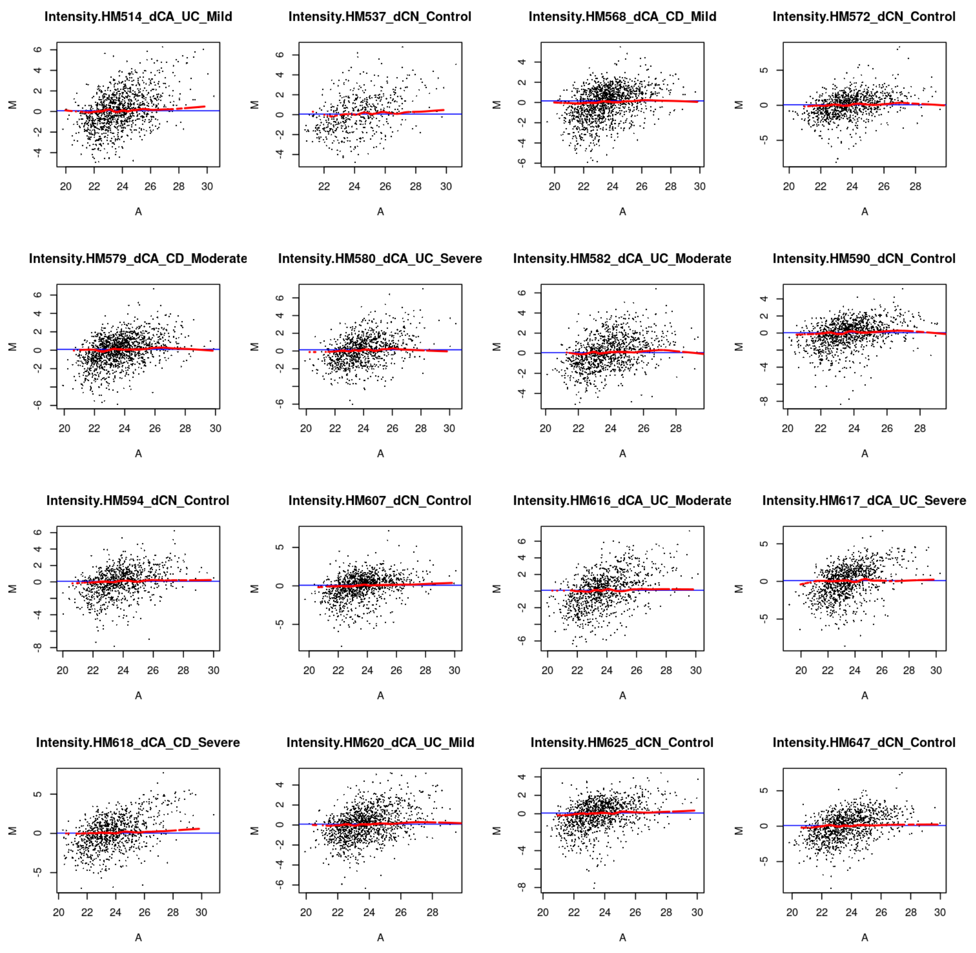
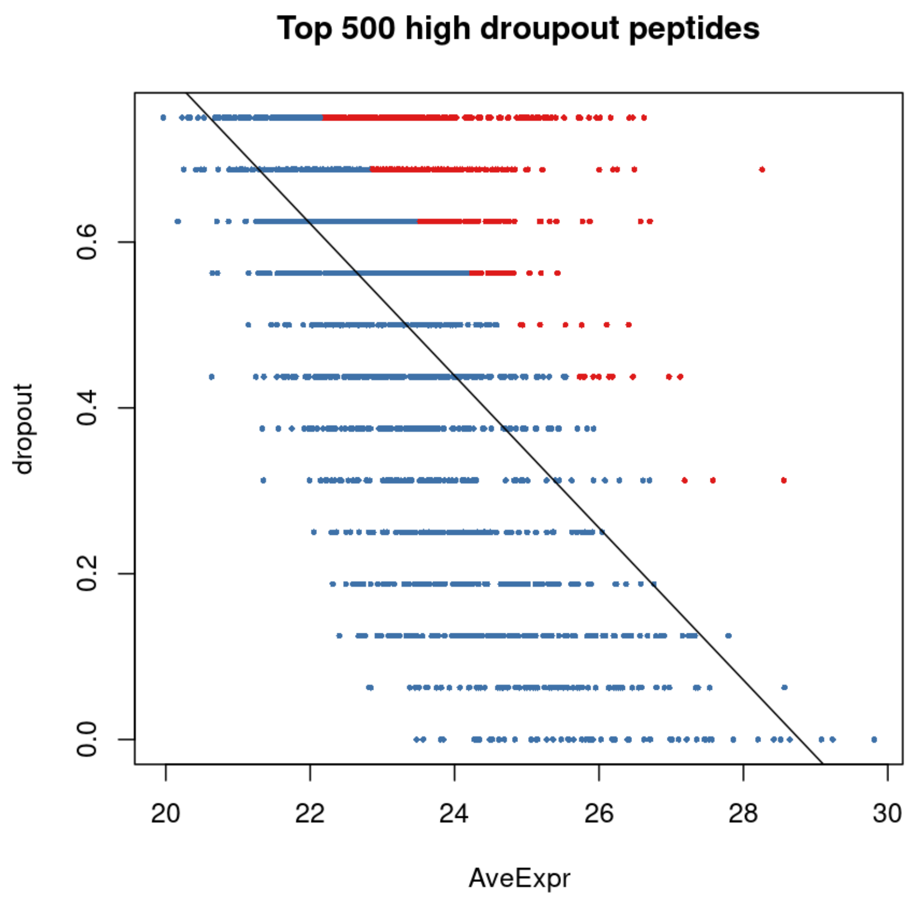


Figure 6: MA plot after normalisation

It can be seen that the trend disappears after *cyclic loess* normalisation.

4.3 Determine missing values pattern

```
> # determine missing values pattern
> hdp <- selectFeatures(y, n_features = 500)
```



```
> # construct matrix M to capture missing entries
> M <- ifelse(is.na(y),1,0)
> M <- M[hdp$msImpute_feature,]

> # plot a heatmap of missingness patterns for the selected peptides
>
> ha_column <- HeatmapAnnotation(group = as.factor(sample_annot$group),
+                                col=list(group=c('Control' = "#E64B35FF",
+                                                 'Mild' = "#3C5488FF",
+                                                 'Moderate' = "#00A087FF",
+                                                 'Severe'="#F39B7FFF")))
> hm <- Heatmap(M,
+ column_title = "dropout pattern, columns ordered by dropout similarity",
+               name = "Intensity",
+               col = c("#8FBC8F", "#FFEFDB"),
+               show_row_names = FALSE,
+                 show_column_names = FALSE,
+                 cluster_rows = TRUE,
+                 cluster_columns = TRUE,
+                 show_column_dend = FALSE,
+                 show_row_dend = FALSE,
+                 top_annotation = ha_column,
+                 row_names_gp =  gpar(fontsize = 7),
+                 column_names_gp = gpar(fontsize = 8),
+                 heatmap_legend_param = list(#direction = "horizontal",
+                 heatmap_legend_side = "bottom",
+                 labels = c("observed","missing"),
+                 legend_width = unit(6, "cm")),
+            )
> hm <- draw(hm, heatmap_legend_side = "left")
```

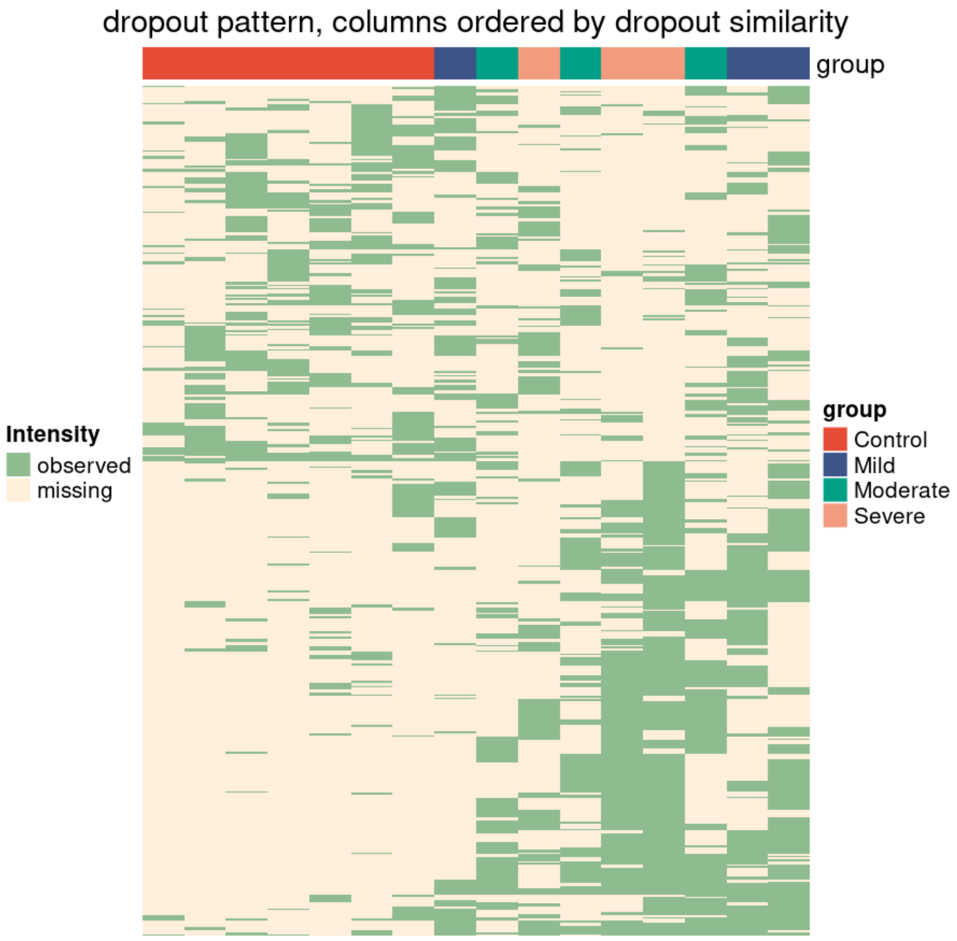


Figure 7: Dropout pattern of top 500 high dropout peptides

As it can be seen, samples from the control group cluster together. There is a structured, block-wise pattern of missing values (i.e. missing values occur in all samples from control individuals, but are measured in disease group). This suggests that missing in not at random. This is an example of **MNAR** dataset. Given this knowledge, we impute using `KNN`, `QRILC` and `msImpute`. We then compare these methods by preservation of local (within experimental group) and global (between experimental group) similarities.

4.4 Imputation

```
> # imputation
>
> y_knn <- impute::impute.knn(y, k = 10)
> y_qrilc <- impute.QRILC(y)[[1]]
> y_msImpute <- scaleData(y)

bi-scaling ...
data scaled

> y_msImpute <- msImpute(y_msImpute)

maximum rank is 15
computing lambda0 ...
lambda0 is 44.79365
fit the low-rank model ...
model fitted.
Imputting missing entries ...
Imputation completed

> group <- as.factor(sample_annot$group)
```

4.5 Assessment of preservation of local and global structures

If you've installed python, and have set up a python environment in your session, you can run this section to compute the GW distance.

Withinness, betweenness and Gromov-Wasserstein (GW) distance

`computeStructuralMerics` returns three metrics that can be used to compare various imputation procedures:

- `withinness` is the sum of the squared distances between samples from the same experimental group (e.g. control, treatment, Het, WT). More specifically the similarity of the samples is measured by the distance of the (expression profile of the) sample from group centroid. This is a measure of preservation of local structures.

- `betweenness` is the sum of the squared distances between the experimental groups, more specifically the distance between group centroids. This is a measure of preservation of global structures.
- `gw_dist` is the Gromov-Wasserstein distance computed between Principal Components of imputed and source data. It is a measure of how well the structures are overall preserved over all principal axis of variation in the data. Hence, it captures preservation of both local and global structures. PCs of the source data are computed using highly variable peptides (i.e. peptides with high biological variance).

An ideal imputation method results in smaller `withinness`, larger `withinness` and smaller `gw_dist` among other imputation methods.

```
> top.hvp <- findVariableFeatures(y)
```

```
> computeStructuralMetrics(y_msImpute, group, y[rownames(top.hvp)[1:50],], k = 16)
```

```
Computing GW distance using k= 16 Principal Components
$withinness
    Mild  Control Moderate   Severe
8.615747 9.455356 8.692705 8.586571

$betweenness
[1] 17.50648

$gw_dist
[1] 0.04294607
```

```
> computeStructuralMetrics(y_knn$data, group, y[rownames(top.hvp)[1:50],], k = 16)
```

```
Computing GW distance using k= 16 Principal Components
$withinness
    Mild  Control Moderate   Severe
8.958293 9.751338 8.973504 8.966409

$betweenness
[1] 17.0302

$gw_dist
[1] 0.04763233
```

```
> computeStructuralMetrics(y_qrilc, group, y[rownames(top.hvp)[1:50],], k = 16)
```

```
Computing GW distance using k= 16 Principal Components
$withinness
    Mild  Control Moderate   Severe
```

```
10.33614 11.81887 10.64143 10.78648

$betweenness
[1] 18.60394

$gw_dist
[1] 0.00998869
```

`Withinness` is smaller by `msImpute` and `KNN`, which indicates that local structures are better preserved by these two methods. However, the `gw_dist` is smaller for `QRILC` over all PCs. This suggests that `QRILC` is likely a better approach compared to the other two methods. Given that the dominant patterns of missing values in this dataset is MNAR, `QRILC` is indeed a reasonable choice, as missing values are likely to be left-censored MNAR.

5 SWATH-DIA Case Study: SWATH-MS analysis of Gfi1-mutant bone marrow neutrophils

This study investigates the proteomic alterations in bone marrow neutrophils isolated from 5-8 week old Gfi1+/-, Gfi1K403R/-, Gfi1R412X/-, and Gfi1R412X/R412X mice using the SWATH-MS technique. This dataset consists of 13 DIA (for SWATH) runs on a TripleTOF 5600 plus (SCIEX). Data available from PXD010943. Peak areas extracted from `13DIAs_SWATHprocessing_area_score_FDR_observedRT.xlsx` accessible via *ProteomXchange*. Rows are peptides. Charge state variations are treated as distinct peptide species. Peptides with more than 4 observed intensity values are retained.

5.1 Data processing

The following code was used to prepare the peptide intensity matrix:

```
> # PXD010943_peakArea_peptides <- readRDS("path/proteomicscasestudies/PXD010943_peakArea_peptides.rds")
>
>
>
> PXD010943_peakArea_peptides$PeptideID <- paste(PXD010943_peakArea_peptides$Protein,
+                                                PXD010943_peakArea_peptides$Peptide,
+                                                PXD010943_peakArea_peptides$Precursor.Charge,
+                                                sep="_")
> y <- PXD010943_peakArea_peptides[,grep("_[1234]", colnames(PXD010943_peakArea_peptides))]
> table(complete.cases(y))
> rownames(y) <- PXD010943_peakArea_peptides$PeptideID
> table(rowSums(!is.na(y)) >= 4 ) # no need to remove any peptides, all are detected in a sufficient number of samples
```

Here we explore the mean-CV^2 trend. No outlier points are detected.

```
> peptideCV2 <- plotCV2(y)
```

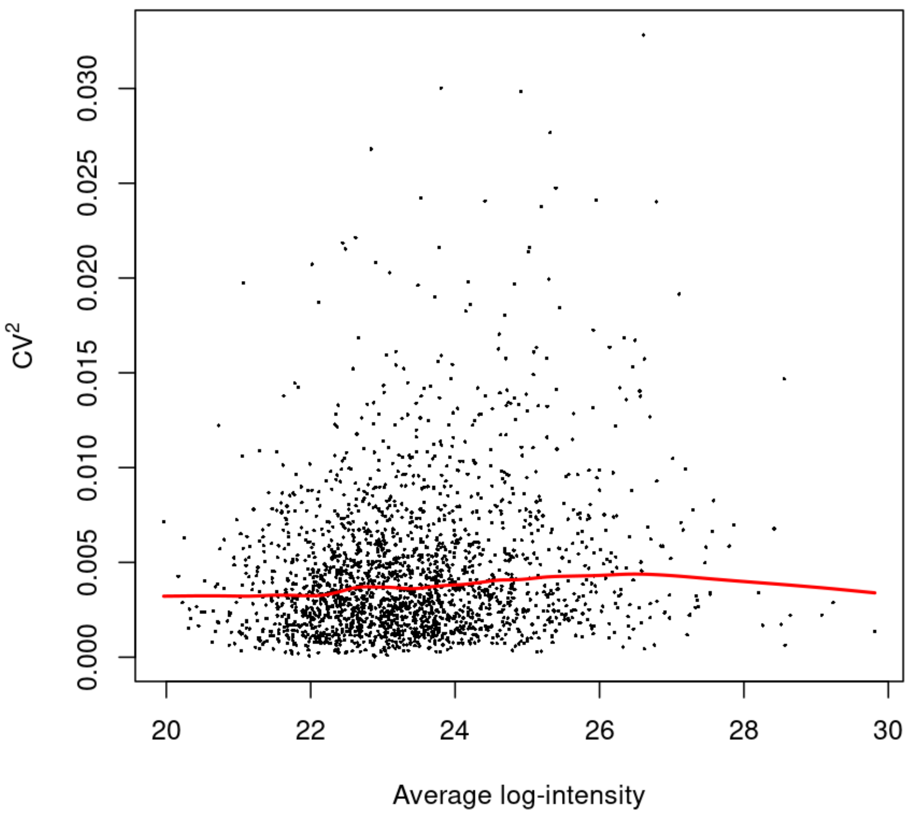


Figure 8: Mean-CV^2 plot

5.1.1 Normalization

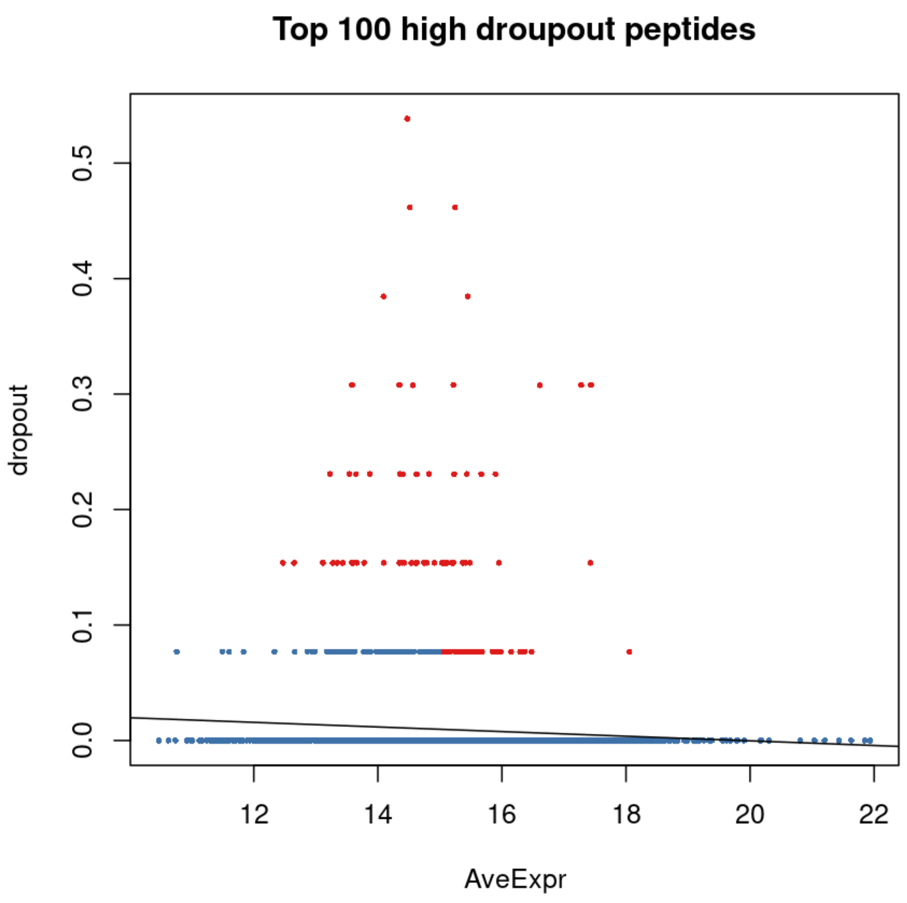
We normalize using *quantile normalization.*

```
> data(pxd010943)
> y <- pxd010943
> # no problematic values for log- transformation
> table(is.infinite(data.matrix(log2(y))))

FALSE
30641

> y <- log2(y)

> y <- normalizeBetweenArrays(y, method = "quantile")
```

5.2 Determine missing values pattern

We use top 100 high dropout peptides as there are 180 partially observed peptides overall

```
> hdp <- selectFeatures(y, n_features = 100)
```



```
> # construct matrix M to capture missing entries
> M <- ifelse(is.na(y),1,0)
> M <- M[hdp$msImpute_feature,]
> # plot a heatmap of missingness patterns for the selected peptides
>
> group <- as.factor(gsub("_[1234]", "", colnames(y)))
> group

 [1] Gfi1      Gfi1      Gfi1      Gfi1      R412Xhet  R412Xhet  R412Xhet
 [8] R412Xhet  R412Xhomo R412Xhomo K403R     K403R     K403R
Levels: Gfi1 K403R R412Xhet R412Xhomo
```

```
> ha_column <- HeatmapAnnotation(group = group)
> hm <- Heatmap(M,
+ column_title = "dropout pattern, columns ordered by dropout similarity",
+               name = "Intensity",
+               col = c("#8FBC8F", "#FFEFDB"),
+               show_row_names = FALSE,
+               show_column_names = FALSE,
+               cluster_rows = TRUE,
+               cluster_columns = TRUE,
+               show_column_dend = FALSE,
+               show_row_dend = FALSE,
+               top_annotation = ha_column,
+               row_names_gp =  gpar(fontsize = 7),
+               column_names_gp = gpar(fontsize = 8),
+               heatmap_legend_param = list(#direction = "horizontal",
+               heatmap_legend_side = "bottom",
+               labels = c("observed","missing"),
+               legend_width = unit(6, "cm")),
+          )
> hm <- draw(hm, heatmap_legend_side = "left")
```

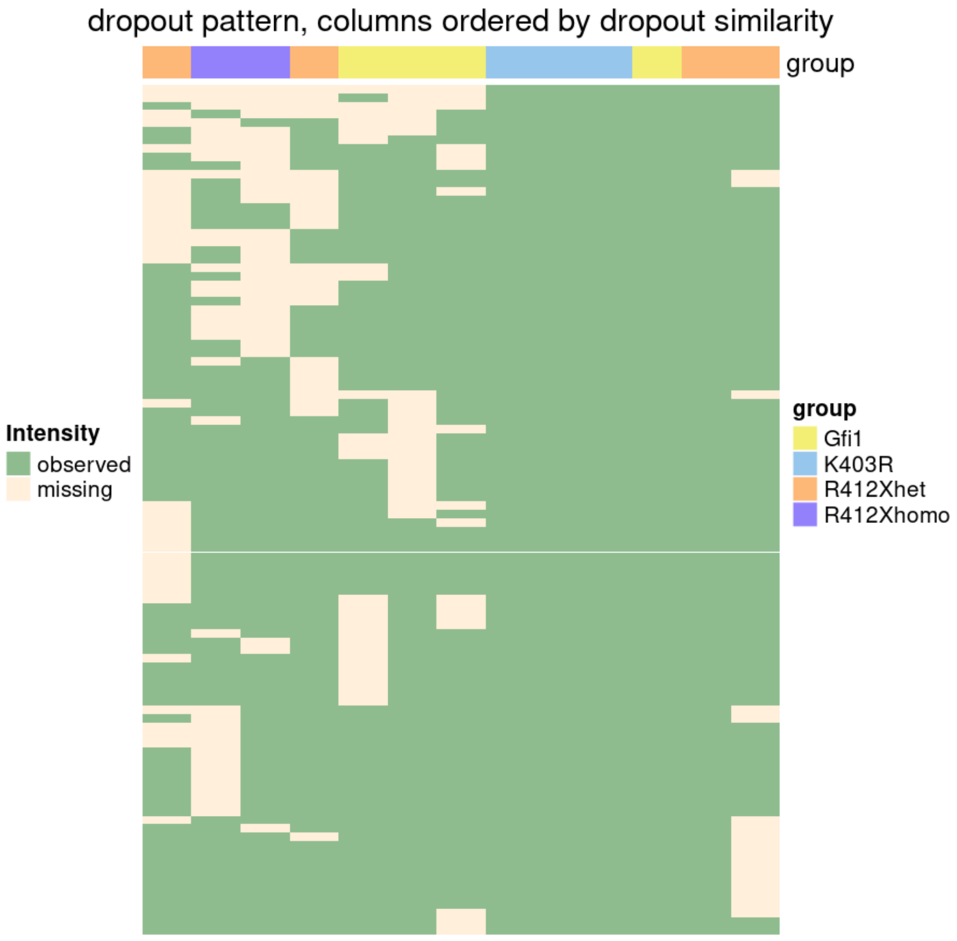


Figure 9: Dropout pattern of top 100 high dropout peptides

It can be seen that missing values are evenly distributed across the groups, and there is no systematic clustering of samples by experimental group. Hence, the dominant pattern of missingness is likely to be **MAR**.

```
> top.hvp <- findVariableFeatures(y)
```

5.3 Imputation

```
> # imputation
>
> y_knn <- impute::impute.knn(y, k = 10)
```

```
Cluster size 2356 broken into 961 1395
Done cluster 961
Done cluster 1395
```

```
> y_qrilc <- impute.QRILC(y)[[1]]
```

Note that the scaling algorithm in `msImpute` does not converge at default iterations in this case. The number of iterations is increased to achieve convergence.

```
> # an example where iteration does not
> # converge at default iteration number
>
> y_msImpute <- scaleData(y, maxit = 50)

bi-scaling ...
data scaled

> y_msImpute <- msImpute(y_msImpute)

maximum rank is 12
computing lambda0 ...
lambda0 is 113.9174
fit the low-rank model ...
model fitted.
Imputting missing entries ...
Imputation completed
```

5.4 Assessment of preservation of local and global structures

`k` is set to the number of samples to capture full information

```
> computeStructuralMetrics(y_msImpute, group, y[rownames(top.hvp)[1:50],], k = 12)

Computing GW distance using k= 12 Principal Components
$withinness
     Gfi1  R412Xhet R412Xhomo     K403R
 8.892049  9.145670  7.279971  6.446897

$betweenness
[1] 18.00916

$gw_dist
[1] 0.09390733

> computeStructuralMetrics(y_knn$data, group, y[rownames(top.hvp)[1:50],], k = 12)

Computing GW distance using k= 12 Principal Components
$withinness
     Gfi1  R412Xhet R412Xhomo     K403R
 8.919586  9.169191  7.261661  6.446843
```

```
$betweenness
[1] 18.02618

$gw_dist
[1] 0.09428275

> computeStructuralMetrics(y_qrilc, group, y[rownames(top.hvp)[1:50],], k = 12)

Computing GW distance using k= 12 Principal Components
$withinness
     Gfi1  R412Xhet R412Xhomo     K403R
 9.041823  9.301998  7.493760  6.459096

$betweenness
[1] 18.05546

$gw_dist
[1] 0.09504425
```

Note `withinness` tend to be smaller in `KNN` and `msImpute`, while `gw_dist` is almost equal to three approaches selected here (up to 3 decimal place). Note that since such a small proportion of peptides are missing, and the dominant missingness pattern appear to be MAR, one can also choose to not impute depending on the downstream analyses.

6 References

Prianichnikov, Nikita, et al. "MaxQuant software for ion mobility enhanced shotgun proteomics." Molecular & Cellular Proteomics 19.6 (2020): 1058-1069. `https://doi.org/10.1074/mcp.TIR119.001720`

Zhang, X., Deeke, S.A., Ning, Z. et al. Metaproteomics reveals associations between microbiome and intestinal extracellular vesicle proteins in pediatric inflammatory bowel disease. Nat Commun 9, 2873 (2018). `https://doi.org/10.1038/s41467-018-05357-4`

Muench, D.E., Olsson, A., Ferchen, K. et al. Mouse models of neutropenia reveal progenitor-stage-specific defects. Nature 582, 109114 (2020). `https://doi.org/10.1038/s41586-020-2227-7`

7 Session Info

```
R version 4.0.0 (2020-04-24)
Platform: x86_64-pc-linux-gnu (64-bit)
```

```
Running under: CentOS Linux 7 (Core)

Matrix products: default
BLAS:   /stornext/System/data/apps/R/R-4.0.0/lib64/R/lib/libRblas.so
LAPACK: /stornext/System/data/apps/R/R-4.0.0/lib64/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
 [1] grid      parallel  stats4    stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
 [1] ComplexHeatmap_2.4.3 imputeLCMD_2.0       impute_1.62.0
 [4] pcaMethods_1.80.0    Biobase_2.48.0       BiocGenerics_0.34.0
 [7] norm_1.0-9.5         tmvtnorm_1.4-10      gmm_1.6-5
[10] sandwich_2.5-1       Matrix_1.2-18        mvtnorm_1.1-1
[13] tidyr_1.1.1          limma_3.44.3         msImpute_0.99.3
[16] reticulate_1.16

loaded via a namespace (and not attached):
 [1] viridis_0.5.1               edgeR_3.30.3
 [3] BiocSingular_1.4.0          jsonlite_1.7.0
 [5] viridisLite_0.3.0           DelayedMatrixStats_1.10.1
 [7] statmod_1.4.34              dqrng_0.2.1
 [9] GenomeInfoDbData_1.2.3      vipor_0.4.5
[11] pillar_1.4.6                lattice_0.20-41
[13] glue_1.4.1                  GenomicRanges_1.40.0
[15] RColorBrewer_1.1-2          XVector_0.28.0
[17] colorspace_1.4-1            pkgconfig_2.0.3
[19] GetoptLong_1.0.2            zlibbioc_1.34.0
[21] purrr_0.3.4                 scales_1.1.1
[23] BiocParallel_1.22.0         tibble_3.0.3
[25] generics_0.0.2              IRanges_2.22.2
[27] ggplot2_3.3.2               pdist_1.2
[29] ellipsis_0.3.1              SummarizedExperiment_1.18.2
[31] magrittr_1.5                crayon_1.3.4
[33] beeswarm_0.2.3              data.table_1.12.8
```

```
[35] tools_4.0.0                scater_1.16.2
[37] softImpute_1.4             GlobalOptions_0.1.2
[39] lifecycle_0.2.0            matrixStats_0.56.0
[41] S4Vectors_0.26.1           munsell_0.5.0
[43] locfit_1.5-9.4             cluster_2.1.0
[45] DelayedArray_0.14.1        irlba_2.3.3
[47] compiler_4.0.0             GenomeInfoDb_1.24.2
[49] rsvd_1.0.3                 rlang_0.4.7
[51] RCurl_1.98-1.2             BiocNeighbors_1.6.0
[53] rappdirs_0.3.1             circlize_0.4.10
[55] rjson_0.2.20               SingleCellExperiment_1.10.1
[57] igraph_1.2.5               bitops_1.0-6
[59] gtable_0.3.0               R6_2.4.1
[61] rdetools_1.0               gridExtra_2.3
[63] zoo_1.8-8                  dplyr_1.0.2
[65] clue_0.3-57                shape_1.4.4
[67] ggbeeswarm_0.6.0           Rcpp_1.0.5
[69] scran_1.16.0               vctrs_0.3.2
[71] png_0.1-7                  tidyselect_1.1.0
```