CASSIOPEIA LFQ
USER GUIDE

What is Cassiopeia

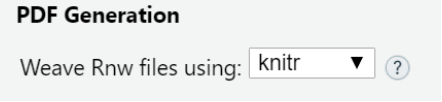
Cassiopeia LFQ is an R script that uses knitr/LaTeX to automatically execute and report standard data analysis steps for label-free proteomics experimental data, such as quality control, differential expression testing (using the limma-trend[1] testing procedure from the R Bioconductor package limma[2]), and other insightful output (PCA, k-means clustering, etc.). It was developed to help out and automatize the LFQ data analysis in the Max Perutz Labs mass spectrometry service facility in Vienna. The script's input is a "proteinGroups.txt" file as produced by database searching using the proteomic software platform MaxQuant[3], as well as manually specified parameters that determine what Cassiopeia is doing. Notably, these parameters allow the downstream data analysis to be uniquely adjusted to the needs of your data – particularly with respect to challenges in data normalization and filtering. Finally, the Cassiopeia output can be exported, including a custom data format for upload in amica, a user-friendly R shiny-app tool for interactively exploring and visualizing omics data. To get an idea of what Cassiopeia can do, check out the example.pdf!

Before running Cassiopeia:

- Install R and R-Studio on your computer.
- Install a LaTeX version your computer (TinyTeX can be installed within R!)
- install required R packages. R-Studio will automatically let you know which packages are missing and suggest their installation.
- Bioconductor packages need to be installed extra by copy-pasting the following code into the R console one at a time (if these packages were already installed some other time, you can skip this step)

```
if (!requireNamespace("BiocManager", quietly = TRUE)){
  install.packages("BiocManager")
}
if (!requireNamespace("limma", quietly = TRUE)){
  install.packages("limma")
}
```

- Open R-Studio -> go to "**Tools** "-> "**Global Options**" -> klick on "**Sweave**" -> tick "**knitr**" on "**Weave RNW files using:**", as it is shown here:



PDF Generation

Weave Rnw files using: knitr ▼ ?

How to run Cassiopeia

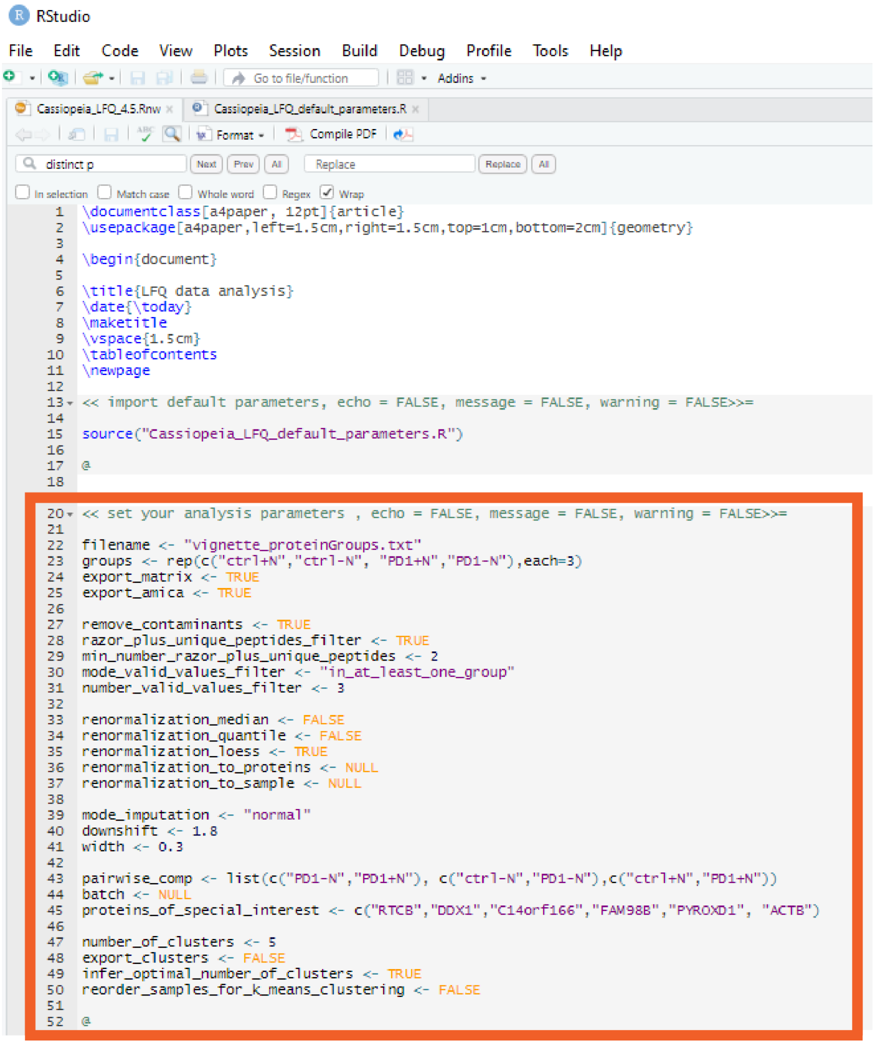
- Create a new folder and copy these files into the folder:

  A) **Cassiopeia_LFQ.Rnw** (the main script)
  B) **Cassiopeia_LFQ_default_parameters.R** (contains default parameter values to be adopted if a parameter is not explicitly specified)
  C) **con_table.txt** (lists common contaminants to fill in information in the Cassiopeia matrix output)

- Copy your **proteinGroups.txt** file (the MaxQuant protein table output) into the same main folder that Cassiopeia_LFQ.Rnw is in. This text file serves as the primary data input for the analysis.

- Open the "Cassiopeia_LFQ.Rnw" file in R-Studio.

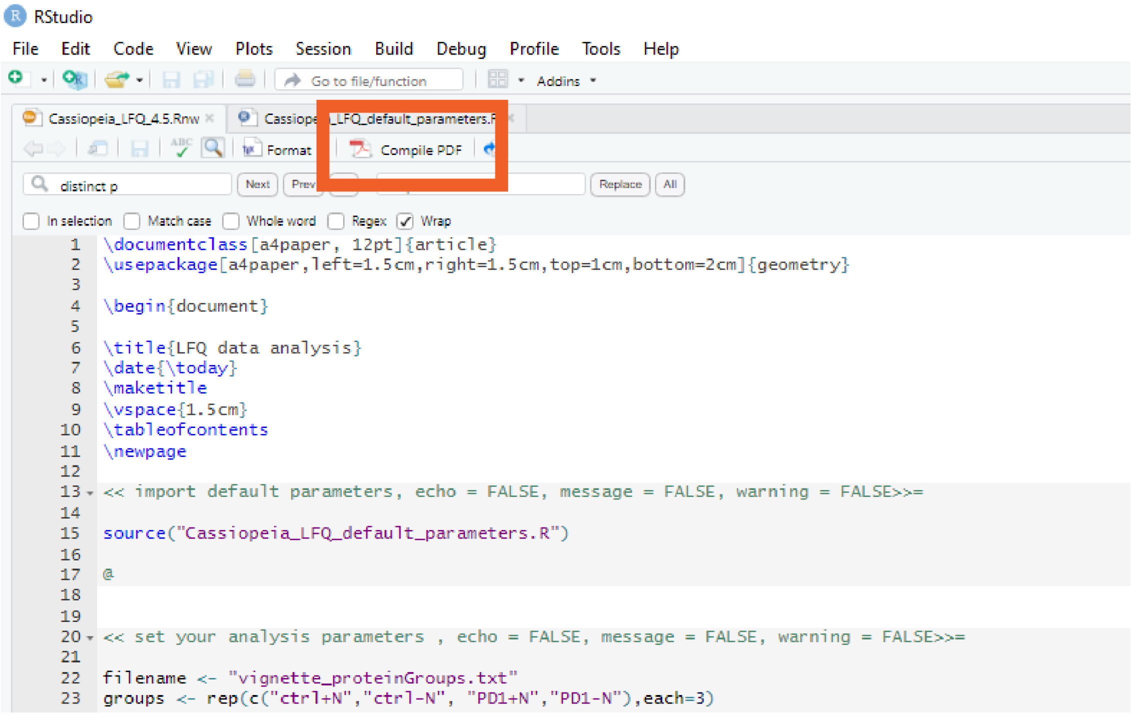
- Specify the parameters that determine what Cassiopeia should be doing. The parameters are to be entered in the section at the beginning of the Rnw file, for example like this:



```
\documentclass[a4paper, 12pt]{article}
\usepackage[a4paper,left=1.5cm,right=1.5cm,top=1cm,bottom=2cm]{geometry}

\begin{document}

\title{LFQ data analysis}
\date{\today}
\maketitle
\vspace{1.5cm}
\tableofcontents
\newpage

<< import default parameters, echo = FALSE, message = FALSE, warning = FALSE>>=

source("Cassiopeia_LFQ_default_parameters.R")

@


<< set your analysis parameters , echo = FALSE, message = FALSE, warning = FALSE>>=

filename <- "vignette_proteinGroups.txt"
groups <- rep(c("ctrl+N","ctrl-N", "PD1+N","PD1-N"),each=3)
export_matrix <- TRUE
export_amica <- TRUE

remove_contaminants <- TRUE
razor_plus_unique_peptides_filter <- TRUE
min_number_razor_plus_unique_peptides <- 2
mode_valid_values_filter <- "in_at_least_one_group"
number_valid_values_filter <- 3

renormalization_median <- FALSE
renormalization_quantile <- FALSE
renormalization_loess <- TRUE
renormalization_to_proteins <- NULL
renormalization_to_sample <- NULL

mode_imputation <- "normal"
downshift <- 1.8
width <- 0.3

pairwise_comp <- list(c("PD1-N","PD1+N"), c("ctrl-N","PD1-N"),c("ctrl+N","PD1+N"))
batch <- NULL
proteins_of_special_interest <- c("RTCB","DDX1","C14orf166","FAM98B","PYROXD1", "ACTB")

number_of_clusters <- 5
export_clusters <- FALSE
infer_optimal_number_of_clusters <- TRUE
reorder_samples_for_k_means_clustering <- FALSE

@
```

  Parameters not specified in this section will adopt their default values, which are defined in the file "Cassiopeia_LFQ_default_parameters.R". All parameters (as well as their default values) are described in detail in the next section of the user guide. Importantly, the parameter section is the only (!!!) section where the user is required to enter their own input.

- finally, click on "compile PDF" on top to run the code and generate your report:


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Cassiopeia_LFQ_4.5.Rnw
Cassiopeia_LFQ_default_parameters.R
Format
Compile PDF
distinct p
Next
Prev
Replace
All
□ In selection □ Match case □ Whole word □ Regex ☑ Wrap

```
\documentclass[a4paper, 12pt]{article}
\usepackage[a4paper,left=1.5cm,right=1.5cm,top=1cm,bottom=2cm]{geometry}

\begin{document}

\title{LFQ data analysis}
\date{\today}
\maketitle
\vspace{1.5cm}
\tableofcontents
\newpage

<< import default parameters, echo = FALSE, message = FALSE, warning = FALSE>>=

source("Cassiopeia_LFQ_default_parameters.R")

@


<< set your analysis parameters , echo = FALSE, message = FALSE, warning = FALSE>>=

filename <- "vignette_proteinGroups.txt"
groups <- rep(c("ctrl+N","ctrl-N", "PD1+N","PD1-N"),each=3)
```

- Your finished pdf report will automatically be created in the folder where the script is located. Any additional output will also be created there. If specified in the parameter section, the final data frame (including all the additionally created columns) will be exported as a text file. Similarly, folders containing amica and k-means clustering results will be created. A folder containing all figures (saved as pdf graphics) will always be created automatically. Before you run Cassiopeia again, make sure to close all output files – else they cannot be overwritten and the script returns an error.

Parameters

filename:

per default: "proteinGroups.txt".
This parameter specifies the file path of the proteinGroups.txt file to be analyzed. Assuming the file is in the same folder as the script, the filename is sufficient. If your proteinGroups.txt file is called differently, specify it in the parameter section as:

filename <- "differentname.txt"

groups:

has to be specified!
This parameter requires a character vector denoting the group identity for each sample that was searched. Importantly, the order of entries has to match the sample column order as they are arranged in the proteinGroups.txt file (more specifically, the intensity column order). For example: In an LFQ experiment of 2 groups with 3 replicates, i.e. 6 samples in total, this parameter could look like this:

groups <- c("group1", "group1", "group1", "group2", "group2", "group2")

or shorter:

groups <- rep(c("group1", "group2"), each=3)

The group names can be chosen freely.

export_matrix:

per default: FALSE
If set to TRUE, Cassiopeia will export its analysis results as a tab-delimited text file with the prefix “Matrix_Export”. This exported table will include all additional variables created by Cassiopeia (i.e. limma statistical output such as p-values and fold changes, k-means cluster centers, and normalized & imputed intensities prefixed with “norm Intensity”). Proteins that were filtered during the analysis will still be present in this table; their filtering status will be indicated by the column “Valid Values Filter (removed)”. Set this parameter to TRUE by:

export_matrix <- TRUE

In addition, the columns in the output will be ordered to make them easy to read and navigate. All the original columns will be kept, barring certain columns that cause problems when opened in Excel. These are: “Peptide IDs", “Peptide is razor”, “Mod peptide.IDs”, “MS.MS.IDs”, “Best MS MS” und “EvidenceIDs” (note: this script was developed for a mass spectrometry facility to create customer-friendly output). When the parameter export_matrix is set to TRUE and the script is run again, make sure that the previously exported table is not open in any program so it can be overwritten.

export_amica:

per default: FALSE
If set to TRUE, Cassiopeia will export its analysis results in a custom amica file format allowing direct upload to amica (will be cited once published), a user-friendly R shiny app to interactively explore, visualize and analyse omics results. Further, one of its strengths is the implementation of network analysis based on the IntAct protein interaction database[4], as well as functional enrichment analysis.

remove_contaminants: per default: TRUE

if TRUE: For the analysis, contaminants (as indicated by the column “Potential contaminant”) will be discarded. Else, set to FALSE.

razor_plus_unique_peptides_filter: per default: TRUE

if TRUE: Cassiopeia filters out rows (proteins) which do not reach the minimum required razor + unique peptide count in the column “Razor + unique peptides”. The required threshold is specified by the parameter called “min_number_razor_plus_unique_peptides” (see next parameter). Else, set to FALSE

min_number_razor_plus_unique_peptides: per default: 2

mode_valid_values_filter: per default: "in_at_least_one_group"

Like in Perseus[5]: This parameter specifies how filtering based on valid values within groups is to be conducted. The three options are:

"in_at_least_one_group"
"in_total"
"in_each_group"

To specify this parameter differently, set this parameter to the desired mode as character, for example:

mode_valid_values_filter <- "*in_total*"

If no normalization strategy is employed, this filtering step is based on NAs in the MaxQuant LFQ intensity columns. If instead any of the available normalization strategies is employed, this filtering step will instead be based on the raw intensity columns (these have a different NA-structure compared to the LFQ-intensities — LFQ-intensities tend to have more NAs due to their specific calculation).

number_valid_values_filter: per default: 2

Like in Perseus[5]: This parameter specifies the minimum number of valid values used in the filtering step based on valid values. In other words, the two parameters “mode_valid_values_filter” and “number_valid_values_filter” therefore work in concert to apply a distinct filtering criterion based on valid values in the intensity data.

renormalization_quantile: per default: FALSE

if TRUE: performs quantile renormalization on log2 protein raw intensities after the removal of contaminants and filtering for valid values. For this step, the script uses the normalizeBetweenArrays function from the R Bioconductor package limma[2].

renormalization_median: per default: FALSE

if TRUE: performs median renormalization on log2 protein raw intensities after the removal of contaminants and filtering for valid values.

renormalization_loess: per default: FALSE

if TRUE: performs cyclic loess renormalization on protein raw intensities after the removal of contaminants and filtering for valid values. See the normalizeBetweenArrays function in package limma[2] for more details. Default method = "fast".

renormalization_to_proteins: per default: NULL

Allows for renormalization to a set of proteins within the data (specified via the column "Gene names"). Works by performing median normalization of log2 raw intensities of all the proteins specified by this parameter, resulting in equal medians for these proteins across all samples (intensity columns). Intensities of proteins (rows) not included in this parameter are shifted accordingly. For example:

renormalization_to_proteins <- c("POM121C", "TAX1BP3", "USH2A", "FTL", "FTH1")

Note: The input requires the full gene name as listed in the MaxQuant proteinGroups column "Gene names".

You can also renormalize to a set of proteins/genes that clustered together by a previous run of Cassiopeia. To accomplish this, read in the respective k-means cluster output (which is to be copied somewhere else so it is not overwritten) by specifying the respective file path and using read.delim, for example:

renormalization_to_proteins <- read.delim("C:\\Users\\madern\\Desktop\\Cassiopeia_LFQ 1.9\\ Gene_names_cluster_1.txt", stringsAsFactors = FALSE, header=FALSE, sep="\t")[,1]

To quickly get the file path of a file, you can type and enter the function choose.files() in the R-console, and browse manually.

renormalization_to_sample: per default: NULL

Allows for renormalization to a specific sample, say sample x. More specifically, this performs a renormalization based on the median of log2 raw intensities of all the proteins that were measured in sample x with non-zero intensity (i.e. no missing values), resulting in equal medians for all those proteins detected in sample x across all samples (intensity columns). Works by specifying a sample index (e.g. 1 would be first sample as ordered in the proteinGroups.txt file). Intensities of proteins (rows) not found in sample x are shifted accordingly. For example:

renormalization_to_sample <- 1

You can also enter more than one sample index. In this case, the normalization is based on the the row medians of all detected proteins in the indexed samples (for example, control samples in an experiment). For example:

renormalization_to_sample <- c(1,2,3)

If you give this parameter all sample indices possible, this will result in a standard median normalization.

mode_imputation: per default: "normal"

determines how remaining NA intensities (after filtering based on valid values) should be imputed.

1) “constant”
   If “constant”: All missing values will be substituted with the minimum over all valid log2 intensity values, rounded down to an integer value (so it can be recognized in the output)

2) “normal”
   If “normal”: All missing values of a sample will be imputed with randomly generated values from a normal distribution that is calculated from the distribution of log2 intensities of the respective sample. In detail:
   First the median and the standard deviation of valid log2 intensities of the respective sample are calculated, here denoted as median_valid and sigma_valid. Then, parameters for the distribution of imputed values will be calculated as:

   mu_imputed = median_valid – downshift*sigma_valid
   sigma_imputed = width*sigma_valid

   This is identical to the calculation in Perseus[5]. The parameters **downshift** and **width** can be adjusted. Per default they are defined as: downshift = 1.8, width = 0.3.

3) “global”
   if “global”: All missing values of a sample will be imputed with randomly generated values from a normal distribution calculated from the distribution of log2 intensities of all samples. Else, it is analogous to mode “normal”.

4) “none”
   if “none”: All missing values remain as they are. However, some of the downstream analysis steps may be compromised and produce errors!

   To change the parameter, just redefine it, for example:
   mode_imputation <- "*constant*"

pairwise_comp: per default: NULL

if NULL: Cassiopeia will not conduct comparisons between groups.

In case pairwise comparisons are desired, this parameter has to be defined as a list of 2-element character vectors (“strings”). The respective 2 entries should denote the two croups to be compared in each comparison, for example *c("group1","group2").* This particular constellation implies fold changes calculated as group2-group1. Note that the group specification of this parameter has to match the specification in the parameter “groups”.

For example, this parameter could look like:

pairwise_comp <- list (c("group1", "group3"), c("group2", "group3"))

which would perform two distinct pairwise comparisons; first group3 vs group1, and then group 3 vs group2.

Statistical testing is done via the limma-trend[1] testing procedure included in the package limma[2]. It is similar to a t-test, but compensates for low sample sizes by considering a variance prior that is calculated from the empirical mean-variance trend of log2 transformed intensities in the experiment (i.e. empirical Bayes). It tests the two-sided hypothesis:

H0: $\mu(\text{group1}) = \mu(\text{group2})$ vs H1: $\mu(\text{group1}) \neq \mu(\text{group2})$

The limma results will be included in the matrix export. Specifically, for each pairwise comparison that is conducted, the following four columns will be added:

- logFC	(log2 fold change)
- AveExpr	(average expression)
- P.Value	(original p-value)
- adj.P.Val	(p-value adjusted for multiple testing by the Benjamini Hochberg procedure)

The respective column names will contain a distinct suffix that is specific to the respective group comparison in the form of "group1__vs__group2". Additionally, the results will be displayed in volcano plots and MA plots in the pdf report.

batch: per default: NULL

If NULL, no batch effect is assumed. Else, this parameter allows the consideration of batch effects (i.e. the correlation between intensity values of two or more samples from different experimental groups) into the limma statistical model used for differential expression testing. Batch effects occur for example when: some (but not all) of the samples come from the same patient, some (but not all) of the samples were prepared on the same day (e.g. cell harvesting), some (but not all) of the samples where prepared by the same operator. In short: batch effects denote any systematic bias during sample preparation and measurement. The limma model can account for this bias by including batch effects as random effects into the model (i.e. correlated error structure).

When including batch effects, specify them the same way as the parameter "groups" is specified; i.e. a character vector with each individual entry denoting the respective batch, and of a total vector length equal to the number of samples in the experiment. Again, the order has to match the order of samples as reported in the MaxQuant proteinGroups.txt file's intensity columns. For example:

batch <- c("batch1", "batch2", "batch1", "batch2", "batch1", "batch2")

or shorter:

batch <- rep(c("group1","group2"), times=3)

proteins_of_special_interest: per default: NULL

This parameter enables a separate highlighting of distinct proteins in the experiment (e.g. bait proteins in pulldown experiments, or KO proteins, or any other proteins of special interest). The specified proteins will be highlighted in extra-generated volcano, MA and profile plots for each pairwise comparison conducted. Also, the proteins will be marked in the k-means clustering visual output. To specify proteins of special interest, string their respective gene names (as listed in the column "Gene names") together into a character vector:

proteins_of_special_interest <- c("RTCB","DDX1","C14orf166","FAM98B","PYROXD1")

Importantly, here, if a gene name entry consists of multiple entries separated by ";", just enter the first name for this parameter!

number_of_clusters: per default: NULL

if NULL, no k-means clustering is performed. If instead a number (e.g. 7) is specified: Cassiopeia performs k-means clustering with a total number of clusters as specified. For the clustering, the intensities of each proteinGroup (row) are transformed to reach same mean intensity across all samples. This makes the clustering invariant to overall differences in expression levels and allows the clustering to pick up similar expression patterns. In the created visual output, each plot shows the center of the respective cluster (as coloured points), as well as the expression pattern of individual proteins by thin black lines in the background. In addition, in the matrix export, a column named "k Means Cluster" will contain information on the respective cluster each proteinGroup (row) was allocated to.

export_clusters: per default: FALSE

if TRUE, Cassiopeia creates an extra folder containing text files that list gene names attributed to specific clusters. Further, it creates a text-file that simply lists all gene names found in the experiment. This list can be used as a reference list to perform functional and GO enrichment analyses on specific clusters (i.e. target vs a background/reference gene list using Fisher's exact test). Some online tools to upload these lists and perform enrichment analyses are:

PANTHER[6]:
http://pantherdb.org/tools/compareToRefList.jsp

GOrilla[7]:
http://cbl-gorilla.cs.technion.ac.il

Further, you can explore networks of clustered proteins to unveil functional associations using the STRING[8] database:
https://string-db.org/

infer_optimal_number_of_clusters: per default: FALSE

if TRUE: Cassiopeia performs a preliminary clustering of the data by varying the total number of allowed clusters. The result is then displayed as an "elbow-plot" that displays the remaining variance on the y-axis (total sum of squares within) versus different total numbers of clusters. This plot can help to assess the total number of distinct clusters you want to use (specified via the parameter "number_of_clusters"). Typically, you want to choose k (the total number of clusters) such that any further increase in k only results in a marginal reduction in the remaining variance.

reorder_samples_for_k_means_clustering: per default: FALSE

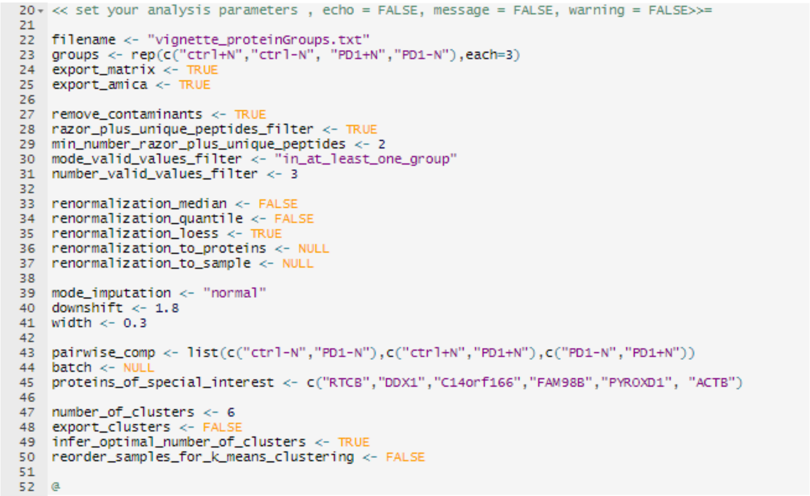
if TRUE: before k-means clustering, samples will be reordered based on group identity. This ensures that samples of the same group will appear next to each other in the k-means cluster output. This parameter therefore only changes the visual output.

Vignette

Before you use Cassiopeia on your own data, I highly recommend running the vignette dataset to check if all the required packages etc. are installed on your computer. To run the vignette, follow these steps:

- Download the the "**vignette_proteinGroups.txt**" from GitHub and copy it into the folder where the script is (see above "How to run Cassiopeia"). This particular proteinGroups.txt file corresponds to data that was recently published (2021) by Asanovićet *et al.*[9] (PMID 33930333). The respective raw data was downloaded from Pride and searched with MaxQuant v1.6.17.0.

- Open the script ("Cassiopeia_LFQ.Rnw") and go to the parameter section. There, copy-paste the parameters written in the file "**vignette_parameters.txt**" (also on GitHub), like this:



```
<< set your analysis parameters , echo = FALSE, message = FALSE, warning = FALSE>>=

filename <- "vignette_proteinGroups.txt"
groups <- rep(c("ctrl+N","ctrl-N", "PD1+N","PD1-N"),each=3)
export_matrix <- TRUE
export_amica <- TRUE

remove_contaminants <- TRUE
razor_plus_unique_peptides_filter <- TRUE
min_number_razor_plus_unique_peptides <- 2
mode_valid_values_filter <- "in_at_least_one_group"
number_valid_values_filter <- 3

renormalization_median <- FALSE
renormalization_quantile <- FALSE
renormalization_loess <- TRUE
renormalization_to_proteins <- NULL
renormalization_to_sample <- NULL

mode_imputation <- "normal"
downshift <- 1.8
width <- 0.3

pairwise_comp <- list(c("ctrl-N","PD1-N"),c("ctrl+N","PD1+N"),c("PD1-N","PD1+N"))
batch <- NULL
proteins_of_special_interest <- c("RTCB","DDX1","C14orf166","FAM98B","PYROXD1", "ACTB")

number_of_clusters <- 6
export_clusters <- FALSE
infer_optimal_number_of_clusters <- TRUE
reorder_samples_for_k_means_clustering <- FALSE

@
```

- Klick on "compile PDF", which should run Cassiopeia LFQ on the vignette dataset. The pdf-report you get should match the "**example.pdf**" on GitHub.

Additional remarks

- The file "**con_table.txt**" serves the purpose to fill all missing entries of contaminant proteins in the columns "Gene names" and "Protein names" of the matrix export. Per default, MaxQuant keeps these entries blank. To combat this, we replace the MaxQuant build-in contaminant.fasta file with a custom contaminant.fasta file that includes proper Uniprot fasta headers for protein sequences. Yet despite Uniprot headers in contaminants.fasta, MaxQuant still omits "Gene names" and "Protein names" entries for contaminants. This is where Cassiopeia helps out. The script extracts the Uniprot accession number contained within the column "Protein IDs" of the proteinGroups.txt file and matches them with entries in "con_table.txt".

- The script also contains some less relevant parameters which I generally recommend to stay unchanged. Their default values are specified in the file "**Cassiopeia_LFQ_default_parameters.R**"

- Colors for proteins are always chosen at random. If you don't like the current ones in your visual output, just rerun Cassiopeia.

- Cassiopeia LFQ was written for MaxQuant version 1.6.17.0 output. I cannot guarantee that the script will conform to the output of future MaxQuant versions, as it is always possible that column names will be changed or that new columns will be added that mess up the pattern finding of the code (which happened before).

- Finally, the code is not terribly efficient, nor is it particularly compact. This script was as much an effort to help out the mass spectrometry service facility as it was a way for me to solidify my programming skills and improve on the way. I am putting it on GitHub so it can be referenced, as well as used by others.

Acknowledgements

This project would not have been possible without the support and interest of the Max Perutz Lab's mass spectrometry facility of the University of Vienna. Special thanks go to Markus Hartl (the facility head), as well as Chen Weiqiang for extensive testing and bug-reporting.

References

1. Phipson, B., Lee, S., Majewski, I. J., Alexander, W. S. & Smyth, G. K. ROBUST HYPERPARAMETER ESTIMATION PROTECTS AGAINST HYPERVARIABLE GENES AND IMPROVES POWER TO DETECT DIFFERENTIAL EXPRESSION. *Ann. Appl. Stat.* **10**, 946–963 (2016).
2. Ritchie, M. E. *et al.* Limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.* **43**, e47 (2015).
3. Tyanova, S., Temu, T. & Cox, J. The MaxQuant computational platform for mass spectrometry-based shotgun proteomics. *Nat. Protoc.* **11**, 2301–2319 (2016).
4. Hermjakob, H. *et al.* IntAct: An open source molecular interaction database. *Nucleic Acids Res.* **32**, 452–455 (2004).
5. Tyanova, S. *et al.* The Perseus computational platform for comprehensive analysis of (prote)omics data. *Nat. Methods* **13**, 731–740 (2016).

6. Mi, H. *et al.* PANTHER version 16: A revised family classification, tree-based classification tool, enhancer regions and extensive API. *Nucleic Acids Res.* **49**, D394–D403 (2021).

7. Eden, E., Navon, R., Steinfeld, I., Lipson, D. & Yakhini, Z. GOrilla: A tool for discovery and visualization of enriched GO terms in ranked gene lists. *BMC Bioinformatics* **10**, 1–7 (2009).

8. von Mering, C. *et al.* STRING: A database of predicted functional associations between proteins. *Nucleic Acids Res.* **31**, 258–261 (2003).

9. Asanović, I. *et al.* The oxidoreductase PYROXD1 uses NAD(P)+ as an antioxidant to sustain tRNA ligase activity in pre-tRNA splicing and unfolded protein response. *Mol. Cell* **81**, 2520-2532.e16 (2021).