Statistical analysis of post-translational modifications quantified by label-free proteomics across multiple biological conditions with R: illustration from SARS-CoV-2 infected cells

Giai Gianetto Quentin

Statistical analysis of post-translational modifications quantified by label-free proteomics across multiple biological conditions with R: illustration from SARS-CoV-2 infected cells

Giai Gianetto Quentin[1]

[1]Institut Pasteur, Bioinformatics and Biostatistics HUB – Department of Computational Biology, CNRS USR 3756 & Institut Pasteur, Proteomic Platform – Mass Spectrometry for Biology Unit, CNRS USR 2000, quentin.giaigianetto@pasteur.fr

May 31, 2021

Abstract

Protein post-translational modifications (PTMs) are essential elements of cellular communication. Their variations in abundance can affect cellular pathways, leading to cellular disorders and diseases. A widely used method for revealing PTM-mediated regulatory networks is their label-free quantitation (LFQ) by high-resolution mass spectrometry. The raw data resulting from such experiments are generally interpreted using specific software, such as MaxQuant, MassChroQ or Proline for instance. They provide data matrices containing quantified intensities for each modified peptide identified. Statistical analyses are then necessary (1) to ensure that the quantified data are of good enough quality and sufficiently reproducible, (2) to highlight the modified peptides that are differentially abundant between the biological conditions under study. The objective of this chapter is therefore to provide a complete data analysis pipeline for analyzing the quantified intensities of modified peptides in presence of two or more biological conditions using the R software. We illustrate our pipeline starting from MaxQuant outputs dealing with the analysis of A549-ACE2 cells infected by SARS-CoV-2 at different time stamps, freely available on PRIDE (PXD020019).

Key words Statistics, R, Data quality control, Clustering, Post-translational modifications, Label-free proteomics, Relative quantification

1 Introduction

Mass spectrometry (MS)-based proteomics allow the identification and quantification of a large number of post-translational protein modifications [1] [2]. A general workflow consists of (i) a proteolytic digestion of proteins into peptides, (ii) an enrichment step to increase the concentration of modified peptides in the samples, and (iii) the analysis of the samples by liquid chromatography-tandem mass spectrometry (LC-MS/MS) [3]. It is often used to study protein phosphorylations [4][5], which is the most commonly studied post-translational modification. This workflow can also be adapted to study other post-translational modifications such as ubiquitinations [6], methylations [7], acetylations [8], or several kind of modifications in unified pipelines [9].

Label-free quantification enables large-scale analyzes and can be applied to experiments composed of many samples and biological conditions, as it can be the case in the field of clinical screening for instance. It allows avoiding drawbacks of labeling methods, whether metabolic (SILAC) or chemical (iTRAQ), which are limited by the cost of labeling reagents, some ineffective labeling, or the limited number of samples that can be analyzed. However, label-free quantification requires careful experimental designs to achieve good reproducibility of quantified values. Indeed, the enrichment step introduces additional variations to the ones traditionally produced by the fluctuation of the liquid chromatography and the ionization conditions in MS-based proteomics. This is the reason why a special care must be taken during this step to carry it out as rigorously as possible in order to maximize the reproducibility of experiments.

The MS/MS spectra obtained after the LC-MS/MS analysis have to be associated to peptides. The identification of spectra is generally carried out by searching the measured MS/MS spectra in a theoretical database, freely downloadable from UniProt website for instance (https://www.uniprot.org/). The localization of a modification on a specific amino-acid of the identified peptide is assessed using a score or a probability, depending on the method applied. It is usually calculated by comparing the measured spectrum with theoretical spectra where the modification is placed on each possible amino acid of the considered peptide [10]. Currently, this research is generally done thanks to specific software, such as MaxQuant [11], MassChroQ [12] or Proline [13], but can also be performed from R (*see* **Note 1**). Depending on the quantification method, either the MS spectra or the MS/MS spectra are next used to quantify the abundance of each modified peptides identified. In the end, such bioinformatics analyses result in data matrices composed of the intensities measured in each of the samples for all the modified peptides identified.

It is common to identify and quantify several tens of thousands peptides in usual experiments. To extract useful information from these large data sets, statistical analyzes of quantified data matrices are required, to check the reproducibility of the analyses and to determine the modified peptides of interest. The objective of this chapter is therefore to provide a data analysis pipeline for analyzing the quantified intensities of modified peptides, in presence of two or more biological conditions using the R software. This analysis pipeline is provided with examples of R codes adapted to a case study. The informed reader can easily change these codes to adapt them to his/her problematic and to what seems relevant to put forward in his/her analysis. We start from outputs of the MaxQuant software [11], which is one of the most popular software tool for this kind of analysis, but the proposed data analysis pipeline can easily be applied to the outputs of various software, as long as they provide the intensities of the modified peptides and the proteins they belong to.

2 Material

2.1 R and RStudio installation

R can be freely installed on most operating systems (Windows, Linux, Mac OS) from https://cran.r-project.org/. The basic R installation includes the language itself as well as packages, most of which are geared towards statistical analysis. One of the strengths of R is the large number of packages that can be downloaded, mainly from the CRAN repository (https://cran.r-project.org/) and, as far as bioinformatics is concerned, from Bioconductor (http://bioconductor.org/). In addition to R, we recommend that the reader install RStudio (https://rstudio.com/products/rstudio/download/) which is a development environment to facilitate the development of R codes and packages. However, all the codes presented herein can be executed from a basic R installation.

Throughout this chapter we assume that the reader has some prior knowledge of R, and we give code examples without dealing with the basics of R, as for example the different types of R objects. For more details on R and its programming language, the reader may refer to the many courses available for free on the R and Bioconductor websites as well as many good books on R and Bioconductor such as [14], [15], [16].

2.2 R packages

In addition to the functions provided in the standard version of R, we will use functions from other R packages in the sample codes provided. These packages are the next: `VennDiagram` [17], `UpSetR` [18], `ggplot2` [19], `grid`, `ggplotify`, `jaccard`, `ggdendro` [20], `ggridges` [21], `limma` [22], `cp4p` [23], `ssize.fdr` [24], `imp4p` [25], `car` [26], `multivariance` [27], `factoextra` [28], `cluster`, `reshape2` [29], `ggpubr` [30].

It is up to the reader to download these packages and to check that they are properly installed on his/her machine.

2.3 Data type

Throughout this chapter we assume that the data are outputs from software used to identify and quantify the abundance of modified peptides and their proteins, such as MaxQuant [11], MassChroQ [12] or Proline [13]. The quantitative data should fit into two matrices: the first contains quantified intensities for PTMs identified on peptides; the second contains quantified intensities for the non-modified proteins to which the modified peptides of the first matrix are associated (i.e. intensities deduced from non-modified peptides of associated proteins).

For instance, when using MaxQuant [11], the first dataset can be extracted from the “Phospho (STY)Sites.txt” (phosphorylation) or “GlyGly (K)Sites.txt“ (ubiquitination), while the second dataset is in the “proteinGroups.txt” file.

The first matrix thus makes it possible to determine the dynamics of the abundance of a PTM between several biological conditions; while the second makes it possible to compare this dynamic with that of the unmodified protein across multiple biological conditions. This comparison is important to ensure that a difference in the abundance of a modified peptide is actually due to the abundance of the modification and not to the dynamic of the associated unmodified protein between compared conditions.

2.4 Case study dataset

To illustrate this protocol, we used a dataset that the reader can freely download from PRIDE (PXD020019). On this server, the reader will find MaxQuant outputs as well as the raw data from the mass spectrometers, such as he/she can reanalyze them using their own workflow and software of interest. This dataset corresponds to the analysis of phosphorylated and ubiquitinated peptides in A549-ACE2 cells infected by SARS-CoV-2 at different time stamps [31]. Hereafter, we will focus on the analysis of phosphorylated peptides, but the same analyses can be performed from the ubiquitinated peptides. To reproduce the results presented along this protocol, the careful reader has to download the MaxQuant outputs "proteinGroups.txt" and "Phospho (STY)Sites.txt" from the PRIDE website.

3 Methods

3.1 Data preprocessing steps

1. Importing data in R: First, the software outputs used to identify and quantify the proteins as well as the peptides have to be saved into a format easily loadable in R. Generally, this can be performed by saving the software outputs into a “txt” or “csv” file. Then, these files can be imported in the R session into a data frame object (*see* **Note 2**). For instance, using MaxQuant outputs, the `read.csv` function can be used for this purpose:
```
> data.prot=read.csv("(path_to_your_file)/proteinGroups.txt",
header=TRUE,sep="\t",quote="")
> data.ptm=read.csv("(path_to_your_file)/Phospho (STY)Sites.txt",
header=TRUE,sep="\t",quote="")
```
2. Filtering peptides which are associated to “Reverse” or “Potential contaminant” peptides: In MaxQuant, “reverse” peptides are artefactual peptides whose amino acid sequence corresponds to the one of peptides of the input database used for the identification, yet in reverse order (*see* **Note 3**). Additionally, “Potential contaminant” peptides are associated with proteins that commonly contaminate samples. Thus, these two kinds of identified peptides can be deleted before subsequent analysis. They are indicated by “+” in the reverse and potential contaminant columns:
```
> data.ptm=data.ptm[which((data.ptm$Reverse!="+")&
(data.ptm$Potential.contaminant!="+")),]
```
3. Filtering peptides which PTM location is not sufficiently well identified: In MaxQuant outputs, the maximum localization probability that has been estimated in one of the analyzed samples is in the “Localization.prob” column. In literature, it is common to keep only the modified peptides for which this probability is greater than 75% (*see* **Note 4**):
```
> data.ptm=data.ptm[which(data.ptm$Localization.prob>0.75),]
```
4. Separating intensities from metadata: The metadata generally contains multiple information, notably related to the identification of the modified peptides and the proteins they belong to. To separate the quantified intensities from the metadata, columns containing quantified intensities can be searched using the `grep` function, and the data can be converted into a matrix object in R with the `as.matrix` function:
```
> int.prot=as.matrix(data.prot[,grep("Intensity.",
  names(data.prot), value=TRUE)])
```
For the proteins, only the columns relative to the quantification of the unmodified proteins can be kept using (*see* **Note 5**):
```
> int.prot=int.prot[,!colnames(int.prot)%in%grep("Phospho",
colnames(int.prot), value=TRUE)]
> int.prot=int.prot[,!colnames(int.prot)%in%grep("Ubi",
colnames(int.prot), value=TRUE)]
```
Similarly, the intensities of the modified peptides can be extracted using (*see* **Note 6**):
```
> int.ptm=as.matrix(data.ptm[,grep("Intensity.Phospho",
names(data.ptm), value=TRUE)])
```
5. Replacing 0 by NA: This step is useful to easily detect missing values in subsequent analyses:
```
> int.prot[int.prot==0]=NA
> int.ptm[int.ptm==0]=NA
```

6. Distinguishing between the different versions of a modified peptide: In MaxQuant outputs, the quantified intensities derived from one two or three and higher PTMs are in columns whose name ends with ___1, ___2, ___3. Thus three lines must be created for each modified peptide of the file.

   ```
   > int.ptm.mod=rbind(int.ptm[,grep("___1",colnames(int.ptm))],
   int.ptm[,grep("___2",colnames(int.ptm))],
   int.ptm[,grep("___3",colnames(int.ptm))])
   > colnames(int.ptm.mod)=unlist(strsplit(colnames(int.ptm)[
   grep("___1",colnames(int.ptm))],split ="___1"))
   ```

   Each modified peptide is next characterized by its identification number, the proteins to which it may belong (quantified in the "proteinGroups.txt" file) and the number of PTMs used to quantify ("multiplicity" column):

   ```
   > id.ptm=rep(data.ptm$id,3)
   > id.pg=rep(data.ptm$Protein.group.IDs,3)
   > multiplicity=c(rep("___1",nrow(int.ptm)),rep("___2",
   nrow(int.ptm)),rep("___3",nrow(int.ptm)))
   ```

7. Removing modified peptide with no quantification values: The modified peptides associated with no quantification value in any sample are useless for performing subsequent statistical analysis (*see* **Note 7**):

   ```
   #Compute the number of observed values on each row
   > sum_notna=apply(int.ptm.mod,1,function(x)sum(!is.na(x)))

   #Extract peptides with at least one observed value
   #on each row
   > int.ptm.mod=int.ptm.mod[which(sum_notna>0),]
   > id.ptm=id.ptm[which(sum_notna>0)]
   > id.pg=id.pg[which(sum_notna>0)]
   > multiplicity=multiplicity[which(sum_notna>0)]
   ```

8. Renaming the columns of the dataset: It can generally be useful to rename the columns with shorter names for further analyses, in particular to make them appear in graphics:

   ```
   > short.name=unlist(strsplit(colnames(int.ptm.mod),
   split="Intensity.Phospho_"))
   > short.name=short.name[short.name!=""]
   > colnames(int.ptm.mod)=short.name
   > data.ptm.mod=data.frame(id.ptm,id.pg,multiplicity,
   int.ptm.mod)
   ```

9. Defining vectors containing the design of experiments: The experimental design is specific to each experiment. It is extremely important, especially for the subsequent statistical analysis. The vectors containing the experimental design must correspond to the columns containing the intensity values. It is advisable to do one for modified peptides, another for proteins, and one containing both. In our case study, we have two factors, the time (6h or 24h) and the biological conditions (mock or SARS-Cov-2 infected samples), 3 replicated samples have been performed in each condition for studying the modifications and 4 replicates have been used for the unmodified proteins. Experimental design corresponding to the columns of `int.ptm.mod` is defined by:

   ```
   #colnames(int.ptm.mod) can be used to check the column names
   #of int.ptm.mod
   ```

```
> Cond.ptm=factor(c(rep("mock",6),rep("SARS_COV_2",6)),levels=
c("mock","SARS_COV_2"))
> Time.ptm=factor(rep(c(rep("24h",3),rep("6h",3)),2),levels=
c("6h","24h"))
> CondTime.ptm=as.factor(paste(Cond.ptm,Time.ptm,sep="."))
```

Experimental design corresponding to the columns of `int.prot` is defined by:

```
> Cond.prot=factor(c(rep("mock",8),rep("SARS_COV_2",8)),
                 levels=c("mock","SARS_COV_2"))
> Time.prot=factor(rep(c(rep("24h",4), rep("6h",4)),2),levels=
                   c("6h","24h"))
> CondTime.prot=as.factor(paste(Cond.prot,Time.prot,sep="."))
```

Both experimental designs can be combined using:

```
> Cond.ptmprot=unlist(list(Cond.ptm,Cond.prot))
> Time.ptmprot=unlist(list(Time.ptm,Time.prot))
```

Additionally, a vector indicating whether the combined design is for the peptide or protein will be useful in subsequent statistical analysis (*see* Subheading 3.9, **Step 1**):

```
#Comp.ptmprot is equal to 1 if the coordinate is related
#to the modified peptide and 0 otherwise
> Comp.ptmprot=c(rep(1,length(Cond.ptm)),
rep(0,length(Cond.prot)))
```

3.2 Checking the reproducibility of identifications

When the samples are highly reproducible, one can expect that the same modified peptides will be identified in each of the samples. The intersections between the sets of modified peptides identified in the samples can be visualized using Venn diagrams or UpSet graphs to highlight any aberrant sample.

1. Creating a matrix containing the row number of an identified peptide if it is quantified:

```
> tr.int.ptm=int.ptm.mod;
> tr.int.ptm[tr.int.ptm>0]=1;
> vecto=as.matrix(1:nrow(tr.int.ptm));
> row.nb.id=apply(tr.int.ptm,2,function(x) x*vecto);
```

2. Calculating the size of each intersection between samples using the `intersect` function:

```
> peptid=row.nb.id[,grep("SARS_COV2_6h",colnames(row.nb.id))]
> nb.sample=ncol(peptid)
#n2 contains the sizes of intersections between 2 samples
> n2=matrix(0,nb.sample,nb.sample);
#n3 contains the sizes of intersections between 3 samples
> n3=array(0,dim=rep(nb.sample,3));
> for (i in 1:nb.sample){for (j in i:nb.sample){
  inter=intersect(peptid[,i],peptid[,j]);
  n2[i,j]=length(inter[inter!=0]);
  for (k in j:nb.sample){
  inter=intersect(peptid[,i],intersect(peptid[,j],peptid[,k]));
  n3[i,j,k]=length(inter[inter!=0]);
  }
}}
```

3. Using the `draw.triple.venn()` function of R package `VennDiagram` to plot a Venn diagram (*see* **Note 8**): The following code displays the numbers of modified peptides found in common or specific of each SARS-Cov-2 samples at 6h (as illustrated in Fig.1**A**):

```
> require(VennDiagram)

> venn.plot = draw.triple.venn(
area1=n2[1,1], area2=n2[2,2], area3=n2[3,3], n12=n2[1,2],
n13=n2[1,3], n23=n2[2,3], n123=n3[1,2,3],
category = colnames(peptid),
fill = c("dodgerblue", "goldenrod1", "seagreen3"),
cat.col = c("dodgerblue", "goldenrod4", "seagreen4"),
cat.cex = 1.5,cat.dist=0.1,
margin = 0.1,
cex = c(1.5,1.5,1.5,1.5,2,1.5,1.5), ind=TRUE);
```

4. Visualizing the intersections using UpSet graphs: Venn diagrams represent the sets of identified peptides using circles or ellipses. However, the larger the number of samples, the more difficult the Venn diagrams are readable. This is the reason why it may be preferable to use UpSet graphs (see Fig.1**B**) when the number of samples becomes large (*i.e.* superior to 5):

```
> require(UpSetR)

> data.upset=as.data.frame(tr.int.ptm)
> names(data.upset)=colnames(data.upset)
> data.upset[is.na(data.upset)]=0

> upset.plot = upset(data.upset, order.by = "freq",
number.angles=315, main.bar.color=4, nsets=ncol(data.upset),
sets.x.label="Nb", text.scale = c(1.5,1.5,1.5,1.5,1.5,1));

#To visualize
> upset.plot
```

5. {Optional step} Converting the plots into ggplot objects: It can be useful to convert the obtained UpSet plot or the Venn diagram into `ggplot` objects (*see* **Note 9**). For this, the `grid` and `ggplotify` R packages can be used:

```
> require(ggplot2)
> require(ggplotify)
> require(grid)

> grid.newpage();
> p = grobTree(venn.plot);
> venn.plot = as.ggplot(p);
> upset.plot = as.ggplot(upset.plot);
```

6. Clustering samples using Jaccard index-based distances: The reproducibility of the identifications between two samples can be evaluated using Jaccard index. This index can be computed using the `jaccard()` function of the R package `jaccard`:

```
> require(jaccard)

> mat.jaccard=matrix(NA,ncol(data.upset),ncol(data.upset))
> for (i in 1:ncol(data.upset)){
    for (j in 1:ncol(data.upset)){
     mat.jaccard[i,j]=jaccard(data.upset[,i],data.upset[,j]);
}}
> colnames(mat.jaccard)=colnames(data.upset)
> rownames(mat.jaccard)=colnames(data.upset)
```

This matrix can be visualized using shades of blue (*see* **Note 10**). Additionally, a distance matrix based on this index can be computed with the `as.dist()` function, and a hierarchical clustering of samples can be performed with the `hclust()` function:

```
> hv = hclust(as.dist(1-mat.jaccard),method="ward.D2")
```

The clustering can be visualized with the `ggdendrogram` function of the R package `ggdendro`:

```
> require(ggdendro)

> ggd = ggdendrogram(hv, rotate = TRUE, theme_dendro = FALSE)
> ggd = ggd + theme(axis.text.y = element_text(hjust = 1))
> ggd = ggd + xlab("Sample")
> ggd = ggd + ylab("Height")
> ggd = ggd + ggtitle("Ward's method with a \n
Jaccard index-based distance")
```

Additionally, the same approach can be used to evaluate the replication of the non-identifications between samples (*see* **Note 11**).

3.3 Checking the reproducibility of quantified values

When the samples are highly reproducible, one can expect the quantified values of a same modified peptides to be similar across the biological replicates.

1. Log2-transforming the intensity values: Before analyzing quantified values, a log2 transformation is usually applied, in particular to decorrelate the intensity levels of peptides from their variances (see [32] and its supplementary material):

   ```
   > log2.int.ptm=log2(int.ptm.mod);
   ```

2. Plotting the distributions of observed values in all samples: The differences between the distributions of the observed values can be used to see whether samples have globally lower or higher quantified values than in the others (see Fig.1**C**). To check if the distribution of quantified values are the same in different samples, it is important to focus exclusively on modified peptides which are quantified in all samples you want to compare (*see* **Note 12**).

   ```
   #compute number of observed values on each row
   > sum_notna=apply(log2.int.ptm,1,function(x){sum(!is.na(x));})

   #keeping log2 intensities of modified peptides with no
   #missing values
   > log2.int.ptm.notNA=log2.int.ptm[sum_notna==ncol(log2.int.ptm),]

   #formatting into a dataframe to use ggplot
   > df=stack(data.frame(log2.int.ptm.notNA));
   > colnames(df)=c("log2_intensities","Samples");
   ```

```
#plotting using stat_density_ridges function
> require(ggridges)

> dis.obs = ggplot(df, aes(x=log2_intensities, y=Samples, fill=
stat(x)))
> dis.obs=dis.obs+stat_density_ridges(geom="density_ridges_gradient",
scale=3,size=0.3,rel_min_height = 0.01,quantile_lines=TRUE,
quantiles = 0.5, alpha = 0.7)
> dis.obs = dis.obs + scale_fill_viridis_c(name="log2(int.)",
option="C", direction=-1)
> dis.obs = dis.obs + labs(title = "Distribution of intensities
for phosphopeptides\n without missing values")
```

3. Clustering samples using the Pearson correlation matrix: A Pearson correlation matrix using only complete pairs of observations between two samples to compute each Pearson correlation coefficient can be obtained by using:

   ```
   > mat.cor=cor(log2.int.ptm, use = "pairwise.complete.obs");
   ```

   The correlation matrix can next be visualized using shades of blue using the `qplot` function as in Fig.1**D** (*see* **Note 13**). Of note, Principal Component Analysis is strongly related to this matrix (*see* **Note 14**) and can also be used to check the clustering of samples by condition. Clustering methods can be applied from the obtained Pearson correlation-based distance matrix to evaluate if the intensities measured in different samples are close or not:

   ```
   > hv = hclust(as.dist(1-mat.cor),method="ward.D2")

   > require(ggdendro)
   > ggd = ggdendrogram(hv, rotate = TRUE, theme_dendro = FALSE)
   > ggd = ggd + theme(axis.text.y = element_text(hjust = 1))
   > ggd = ggd + xlab("Sample")
   > ggd = ggd + ylab("Height")
   > ggd = ggd + ggtitle("Ward's method with a \nPearson
   correlation-based distance")
   ```

   Such clustering can be useful for determining which biological conditions are similarly distributed, and which are less so (see Fig.1**E**).

3.4 Is the number of replicates sufficient in each condition?

A sufficient number of samples to confidently trust the results of a statistical test can be determined by seeking the minimum number of samples reaching a satisfactory statistical power. The calculation of this statistical power depends on several factors. In case of a classical t-test comparing the average intensities between two conditions, it depends on the minimal average difference intensity between conditions that is expected to be detectable, the variance of the intensities, the type 1 error (threshold on the p-values), and the number of samples in each condition (sample size). That is why it is logical to conduct a power analysis from a first MS-based experiment to assess how many samples are sufficient to obtain robust statistical results in a subsequent analysis (*see* **Note 15**). This analysis can also be performed after the final data have been acquired to evaluate what is the confidence in the results.

The R package `ssize.fdr` contains several functions that can be used to calculate minimum sample sizes [24]. The purpose of each of these functions is to estimate an average statistical power over all the tests carried out on all the peptides in function of the sample size. In case of more than two conditions, the functions `ssize.F` and `ssize.Fvary` can be used by defining a level of false discovery rate, a user-specified proportion of non-differentially abundant peptides, a design matrix and a standard deviation (or the parameters of inverse gamma distribution followed by variances of peptides for `ssize.Fvary`). We detailed hereafter how to use the latter one.

1. Creating a design matrix from the vectors defining the design of experiment: Additional explanations on this matrix and the contrast matrix can be found in *see* Subheading 3.9, **Step 1** and *see* Subheading 3.9, **Step 2**.

   ```
   > dat.design=data.frame(Cond.ptm,Time.ptm)
   > design=model.matrix(~Cond.ptm+Time.ptm,data=dat.design)
   ```

2. Creating a contrast matrix specific to a statistical test from which average statistical power will be computed:

   ```
   #test if the second coefficient related to biological
   #condition (mock or SARS-CoV-2) is null
   # ct=cbind(c(0,1,0))
   #test if the second coefficient related to biological
   #condition (mock or SARS-CoV-2)) is null
   #and the third coefficient related to the time is also null
   > ct=cbind(c(0,1,-1),c(0,0,1))
   ```

3. Using `limma` [22] to estimate the model and determine prior variance and degrees of freedom related to the test:

   ```
   > require(limma)
   > fit=eBayes(contrasts.fit(lmFit(log2.int.ptm.notNA,design),ct),
   robust=TRUE)
   > s02=fit$s2.prior;
   > d0=mean(fit$df.prior);
   ```

4. Estimating the proportion of non-differentially abundant peptides: Histogram and calibration plot of p-values [23][32] are displayed in Fig.2**A**-**B**.

   ```
   > pval_ct=fit$F.p.value
   #histogram of pvalues
   > hist(pval_ct,50,main="Histogram of pvalues",col=4)

   > require(cp4p)
   #Calibration plot of pvalues to choose a method
   > calibration.plot(pval_ct,"ALL")

   #The proportion of non-differentially abundant peptides is
   #estimated using the distribution of the p-values related
   #to the contrasted test with default method ("pounds")
   > prop=estim.pi0(pval_ct)
   ```

5. Using the `ssize.Fvary` function: Additional explanations can be found in [24].

   ```
   #FDR level required
   > fdr=0.01

   #Minimum statistical power required
   > pwr=0.75
   ```

```
#Function computing the degrees of freedom related to the
#design of experiment: here we have 4 groups (two conditions
#at two time points) and two parameters estimated in the model
> df=function(n){4*n-2;}

#parameters of the inverse gamma distribution of the variances
> alph=d0/2
> beta=d0*s02/2

#eps represents values for the model parameters related to
#biological conditions and the time. Because we test the
#nullity of the parameters, more this value is close to 0
#and the harder it will be on the statistical test to know
#if the coefficients are indeed zero or not from data:
#the average statistical power will logically be lower.
> eps=0.1

> require(ssize.fdr)
> ftv=ssize.Fvary(X=design,beta=c(1,eps,-eps),L=ct,dn=df,a=alph,
b=beta,fdr=fdr,power=pwr,pi0= prop$pi0.est$pi0.Pounds,maxN=20)
```

Fig.2**C**-**D** represents the average statistical power in function of the sample size for different values of `eps`. The closer it is to 0, and lower the statistical power will be.

3.5 Normalisation step

As explained in the introduction, the enrichment step induces a lower reproducibility among the reported intensities. That is why a normalization step is of prime importance to correct the variations of intensities. In literature, several methods have been proposed [34][35]. Several normalization methods can be used from the `wrapper.normalizeD` function of the R package `DAPAR` [36] after creating an object of class MSnSet. A classical approach consists to apply a median-centering function in the samples which are a replication of a same biological condition.

1. Creating a median-centering function:

```
> median.norm=function(intensities,condition){
  int.norm = intensities
  lev=levels(condition)
  medianes=rep(0,ncol(intensities))
  for (i in 1:length(lev)){
   #column of the condition
   col_cond=which(condition == lev[i])
   #nb missing values in the condition for proteins/peptides
    nbna_cond=apply(intensities[,col_cond],1,
     function(x){sum(is.na(x));})
    #medianes from proteins/peptides without missing values
    medianes[col_cond]=apply(intensities[which(nbna_cond==0),
     col_cond],2,median,na.rm=T);
    moy=mean(medianes[col_cond]);
    for (j in col_cond){
      #computing differences between each median and
      #the average median in each condition
      int.norm[,j] = intensities[,j] - medianes[j] + moy
    }}
  return(int.norm)
}
```

2. Performing the normalization on intensities of modified peptides:

```
> log2.int.ptm.norm=median.norm(intensities=log2.int.ptm,
condition= CondTime.ptm)
```

3. Performing the normalization method on the protein intensities:

```
> log2.int.prot=log2(int.prot)
> log2.int.prot.norm= median.norm(intensities=log2.int.prot,
condition=CondTime.prot)
```

3.6 Dealing with missing values

In case of multiple biological conditions, modified peptides can have similar detection profiles: they are detected, or not detected, under the same biological conditions. These detection profiles may be of greatest interest. Indeed, non-detection is generally synonymous either with an absence of the modified peptide in the biological condition; or with its low abundance, as missing values mainly result of the limit of detection of the mass spectrometer. The detection profile of the modified peptide have to be compared with those of their belonging protein in order to highlight over- or under-abundance of the modification relatively to the one of the unmodified protein they belong to (*see* **Note 16**).

1. Creating a function returning detection profiles (1 if the modified peptide is detected and 0 if not):

```
> detect.prof=function(mat,condition){
lev=levels(condition)
detect=matrix(0,nrow(mat),length(lev))
for (k in 1:nrow(mat)){
    for (i in 1:length(lev)){
        for (j in which(condition ==lev[i])){
            if (!is.na(mat[k,j])){
                detect[k,i]=1
            }
        }
    }
}
colnames(detect)=levels(condition)
return(detect)
}
```

2. Computing detection profiles for modified peptides and their proteins:

```
> detect.ptm=detect.prof(log2.int.ptm.norm,CondTime.ptm)
> detect.prot=detect.prof(log2.int.prot.norm,CondTime.prot)
```

Missing values can be replaced within a same condition using imputation algorithms. If no value have been measured along replications of a same experiment in a condition, then either a value can be inferred assuming that there is a relationship to the values measured in the other conditions, or no value can be inferred (*see* **Note 17**). Two main kinds of missing values arise in MS-based proteomics: either missing not at random (MNAR) values or missing completely at random (MCAR) values [37]. The R package `imp4p` proposes a multiple imputation strategy to deal with these two kinds of missing values and functions to analyze missing value mechanisms [25].

3. Analyzing missing value mechanisms:

   ```
   > require(imp4p)
   > log2.int.ptm.impMCAR=impute.mle(log2.int.ptm.norm,
   conditions=CondTime.ptm)
   > resmod=estim.mix(tab=log2.int.ptm.norm,
   tab.imp=log2.int.ptm.impMCAR,conditions= CondTime.ptm)
   > resmod$pi.na
   > resmod$pi.mcar
   ```

   The proportion of missing values is almost 30% in each sample. The proportion of MCAR values is estimated between 0 and 30% depending on the considered sample.

4. Imputing missing values of peptides using a multiple imputation strategy combining MCAR and MNAR values with the `impute.mi()` function:

   ```
   > log2.int.ptm.imp=impute.mi(tab = log2.int.ptm.norm,
   conditions = CondTime.ptm)
   > colnames(log2.int.ptm.imp)=colnames(log2.int.ptm)
   ```

5. Imputing missing values of proteins using a MCAR-devoted imputation algorithm:

   ```
   > log2.int.prot.impMCAR=impute.mle(log2.int.prot.norm,
   conditions=CondTime.prot)
   > colnames(log2.int.prot.impMCAR)=colnames(log2.int.prot.norm)
   ```

3.7 Mapping the quantified values of the unmodified protein to the ones of modified peptides.

After the steps of data pre-processing, normalization and imputation, it is useful to create a dataset containing all the modified peptides retained on rows with their values by columns, as well as the values of the potential proteins to which they belong for subsequent statistical analysis. For each modified peptide, these intensities can be retrieved using the identification numbers of their protein groups in MaxQuant: they are in the column "id" of the "proteinGroups.txt" file.

The following R code creates a dataset containing all the information necessary for subsequent analysis:

```
> int.ptm.prot=NULL;id.ptm.mod=NULL;
> id.pg.mod=NULL;multiplicity.mod=NULL;
> prof.ptm=NULL;prof.prot=NULL;id.prot.mod=NULL;
> for (i in 1:nrow(data.ptm.mod)){
  id_prot=unlist(strsplit(data.ptm.mod$id.pg[i],split =";"))
  for (j in 1:length(id_prot)){
    id.ptm.mod=c(id.ptm.mod,data.ptm.mod$id.ptm[i])
    id.prot.mod=c(id.prot.mod,id_prot[j])
    id.pg.mod=c(id.pg.mod,data.ptm.mod$id.pg[i])
    prof.ptm=rbind(prof.ptm,detect.ptm[i,])
    multiplicity.mod=c(multiplicity.mod,
                       data.ptm.mod$multiplicity[i])
```

```
    if (length(data.prot$id%in%id_prot[j])!=0){
     prof.prot=rbind(prof.prot,
            detect.prot[which(data.prot$id%in%id_prot[j]),])
     int.ptm.prot=rbind(int.ptm.prot,c(log2.int.ptm.imp[i,],
     log2.int.prot.impMCAR[which(data.prot$id%in%id_prot[j]),]))
    }else{
     prof.prot=rbind(prof.prot,rep(NA,ncol(detect.prot)))
     int.ptm.prot=rbind(int.ptm.prot,c(log2.int.ptm.imp[i,],
                         rep(NA,ncol(log2.int.prot.impMCAR))))
    }
}}
> data.ptm.prot=data.frame(id.ptm.mod,id.pg.mod,id.prot.mod,
multiplicity.mod,prof.ptm,prof.prot,int.ptm.prot)
```

3.8 Extracting modified peptides with specific detection profiles relatively to their unmodified protein.

Before analyzing the quantified intensities, specific detection profiles can be extracted to highlight modified peptides that are detected while their unmodified protein is not quantified for instance. For instance, this can be performed with the following steps for the “mock” condition of our case study dataset:

1. Putting aside peptides with no quantified values in the considered condition:
   ```
   > nb.mock.ptm=apply(prof.ptm[,grep("mock",colnames(prof.ptm))],
   1,function(x){sum(x==1)})
   > ind_noval=which(nb.mock.ptm==0)
   ```
2. Putting aside peptides with associated proteins having no quantified values in the considered condition:
   ```
   > nb.mock.prot=apply(prof.prot[,grep("mock",colnames(prof.prot))],
   1,function(x){sum(x==1)})
   > ind_noprot=which(nb.mock.prot==0)
   ```
3. Putting aside peptides quantified at time stamps where their protein is not quantified:
   ```
   #peptides quantified at time stamps where their protein is
   #not quantified.
   > ind=NULL
   > for (i in 1:nrow(prof.ptm)){
       if (length(which(
           prof.ptm[i,grep("mock",colnames(prof.ptm))]==
           prof.prot[i,grep("mock",colnames(prof.prot))]
           ))==0){
          ind=c(ind,i)
       }
   }
   > ind_diffprof=ind[!ind%in%c(which(nb.mock.prot==0),
   which(nb.mock.ptm==0))]
   ```
4. {Optional steps} Exporting data in a txt file: Next, peptides of interest can be exported outside R in "txt" files using the `write.table` function for instance. For peptides quantified at time stamps where their protein is not quantified, they can be exported using:

```
> write.table(data.ptm.prot[ind_diffprof,],
"(path_to_your_file)/pept_wo_prot.txt",sep="\t",row.names=F)
```

3.9 Extracting modified peptides evolving significantly differently from their unmodified protein.

A common strategy aims to focus on the evolution of the intensities of modified peptides at different time stamps and under different biological conditions, notably to highlight PTM-mediated pathways involved in diseases or cellular disorders. For example, this can happen if one has samples related to patients belonging to different categories, and whose modified proteomes are measured at different time stamps. In this framework, the measured intensity of a modified peptide in a sample can be explained by three factors: the abundance of its membership protein, the patient's category and the time stamps.

When no value is missing in the dataset, the following linear model can easily be estimated in each condition using R:

$$y_j = \alpha + \beta \mathbb{1}_{comp(j)=pept} + \sum_{t \in [t_1,...,t_n]} \theta_t \mathbb{1}_{time(j)=t} + \delta_t \mathbb{1}_{time(j)=t} \mathbb{1}_{comp(j)=pept} + \epsilon_j \quad (1)$$

where ϵ_j is a Gaussian white noise.

In this equation, $Y = (y_j)_{j \in [1,J]}$ is a vector containing the intensities of both the modified peptide and its unmodified protein, and the function $\mathbb{1}_{comp(j)=pept}$ is equal to 1 if the j^{th} coordinate of the vector Y is an intensity related to the modified peptide and 0 if it is related to the protein. Similarly, the function $\mathbb{1}_{time(j)=t}$ is equal to 1 if the j^{th} coordinate of the vector y is an intensity related to the time stamp t and 0 otherwise. Here, the measured intensity y_j is explained in function of four parameters: the intercept α which represents the average intensity level of the protein to which the peptide belongs at time t_0; the β parameter which represents the gap between this average at t_0 and the one of the intensities of the modified peptide; the list of θ_t which are the effects of each other time point regardless of whether the intensity is that of the peptide or the protein; and the list of δ_t that correspond to the interaction effects between the peptide intensities and the time stamps.

Testing the nullity of a linear combination of the model coefficients allow to select peptides of interest. For this, a design matrix (defining the linear model) and a constrast matrix (defining the linear combination of model coefficients that have to be tested) has to be specified in R.

1. Defining the design matrix: The model (1) can also be represented as a product between a design matrix X, composed of ones and zeros, and a vector of parameters $B = (\alpha, \beta, \theta_1, \ldots, \theta_T, \delta_1, \ldots, \delta_T)^T$, such as $Y = XB + \epsilon$, where $\epsilon = (\epsilon_j)_{j \in [1,J]}$. In R, the design matrix can be defined by using the `model.matrix` function (*see* **Note 18**):

   ```
   > design = model.matrix(~Comp.ptmprot*Time.ptmprot)
   ```

2. Defining the contrast matrix: This matrix (referred to as C) is used to test if a linear combination of coefficients composing the estimated model is null. In this framework, the null hypothesis of the test is defined by $H_0 : C^T B = 0$ and the alternative hypothesis by $H_1 : C^T B \neq 0$.
   To test if the modified peptide has a dynamic similar to that of the protein over time, we have to test that all the interaction parameters δ_t are equal to 0, *i.e.* $\delta_1 = 0$ and $\delta_1 - \delta_2 = 0$, $\delta_2 - \delta_3 = 0$, etc. For instance, this contrast matrix will test $\delta_1 = 0$ and

$\delta_1 - \delta_2 = 0$:

$$C^T B = 0 \iff \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 & \cdots & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \theta_1 \\ \vdots \\ \theta_T \\ \delta_1 \\ \vdots \\ \delta_T \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_1 - \delta_2 \end{pmatrix} = 0 \tag{2}$$

In our case study dataset of two time stamps, only one interaction parameter is in the model defined by the design matrix. Using this model, the contrast matrix testing the nullity of this parameter is defined by only one column:

```
> ct=cbind(c(0,0,0,1))
```

Alternatively, if it is desired to remove all the peptides which are constant over time, it is advisable to test the nullity of all the coefficients in relation to time, *i.e.* the nullity of the (θ_t) and (δ_t) parameters. In our case study, the contrast matrix to be used is:

```
> ct=cbind(c(0,0,1,0),c(0,0,1,-1))
```

3. For the following steps, we consider a dataset without missing values in order to be able to estimate linear models and perform statistical tests:

```
nbna=apply(int.ptm.prot,1,function(x)sum(is.na(x)))
data.ptm.prot=data.ptm.prot[which(nbna==0),]
int.ptm.prot=int.ptm.prot[which(nbna==0),]
```

4. a. Applying a classical Fisher's F test: Once the design matrix and the contrast matrix have been defined, a classical Fisher's F test can be applied in a condition (here, "mock") for all the modified peptides of our dataset using the `linearHypothesis` function of the `car` R package [26]:

```
> require(car)
> condition="mock"
#condition="SARS_COV_2"
> p.value_ANOVA=rep(NA, nrow(int.ptm.prot))
> for (i in 1:nrow(int.ptm.prot)){
Int=int.ptm.prot[i,Cond.ptmprot==condition]
Comp=Comp.ptmprot[Cond.ptmprot==condition]
Time=Time.ptmprot[Cond.ptmprot==condition]
mod <- lm(Int ~ Comp * Time)
p.value_ANOVA[i]=linearHypothesis(mod,t(ct))$‘Pr(>F)‘[2]
}
```

b. Applying a Fisher's F test using limma: The `limma` R package can alternatively be used to test the nullity of the linear combination of coefficients with Fisher's F test. It uses a regularisation of the variances based on the assumption that they follow an inverse Gamma distribution [22]:

```
> require(limma)
> design=model.matrix(~Comp.ptmprot[Cond.ptmprot==condition]
*Time.ptmprot[Cond.ptmprot== condition])
> fit=eBayes(contrasts.fit(lmFit(int.ptm.prot[,Cond.ptmprot==
condition], design),ct),robust=TRUE)
```

```
> p.value_LIMMA=fit$F.p.value
```

5. Selecting peptides with a significantly different dynamic than their parent unmodified protein with a chosen FDR threshold: For this, an adaptive FDR control procedure can be applied to get adjusted pvalues using the `cp4p` R package [23]:

   ```
   > require(cp4p)
   > a=adjust.p(p.value_ANOVA,pi0.method = "pounds")
   > a.pvalue_ANOVA=a$adjp$adjusted.p
   > ptm.sel.mock=which(a.pvalue_ANOVA<0.01)
   ```

   Here, a 1% FDR threshold has been used and the proportion of true null hypotheses has been estimated using the method of Pounds and Cheng [33]. Likewise, modified peptides which have a significantly different dynamic from their unmodified protein in the SARS-CoV-2 condition can be recovered by using `condition="SARS_COV_2"` and listing them in a vector `ptm.sel.sarscov2` from previous step (*see* Subheading 3.9, **Step 4**).

6. Extracting peptides of interest: Finally, a dataset containing all the modified peptides evolving significantly differently than their unmodified proteins in at least one condition can be extracted from `ptm.sel.mock` and `ptm.sel.sarscov2` with:

   ```
   > data.ptm.prot.sel=data.ptm.prot[unique(c(ptm.sel.mock,
   ptm.sel.sarscov2)),]
   > int.ptm.prot.sel=int.ptm.prot[unique(c(ptm.sel.mock,
   ptm.sel.sarscov2)),]
   ```

   They can next be exported outside R using the `write.table` function for instance (*see* Subheading 3.8, **Step 4**).

3.10 Clustering of modified peptides using their dynamics relatively to their unmodified protein.

When prior information is not available to perform groupings of peptides, unsupervised clustering methods can be applied to highlight clusters of peptides behaving similarly for subsequent biological interpetation.

In label-free proteomics, many physico-chemical properties can lead to higher or lower intensity levels for different peptides with similar abundance in different conditions. Consequently, the cluster analysis of peptides has to be performed without considering the global intensity levels of the peptide, or the one of its protein, but rather their deviation from a reference intensity level. With this purpose, the intensities of the modified peptide and of its unmodified protein can be modeled by a linear model with interaction parameters using the three factors that are time stamps, biological conditions, and the studied component (protein or peptide):

$$\begin{gathered} y_j = \alpha + \beta_0 \mathbb{1}_{comp(j)=pept} \\ + \sum_{k\in[1,K]} \beta_k \mathbb{1}_{cond(j)=k} + \gamma_k \mathbb{1}_{cond(j)=k} \mathbb{1}_{comp(j)=pept} \\ + \sum_{t\in[t_1,t_n]} \theta_t \mathbb{1}_{time(j)=t} + \vartheta_t \mathbb{1}_{time(j)=t} \mathbb{1}_{comp(j)=pept} \\ + \sum_{k\in[1,K]} \sum_{t\in[t_1,t_n]} \delta_{tk} \mathbb{1}_{time(j)=t} \mathbb{1}_{cond(j)=k} + \varphi_{tk} \mathbb{1}_{time(j)=t} \mathbb{1}_{cond(j)=k} \mathbb{1}_{comp(j)=pept} \\ + \epsilon_j \end{gathered} \tag{3}$$

where ϵ_j is a Gaussian white noise.

Thus, the dynamics of the modified peptide compared to the one of its unmodified protein is characterized by all the parameters multiplied by $\mathbb{1}_{comp(j)=pept}$, with the exception of the β_0 parameter which represents the overall difference between the intensity level of the modified peptide and that of its unmodified protein. Therefore, the dynamic of the modified peptide relative to that of its protein can be characterized by the parameter vector $(\gamma_1, \cdots, \gamma_K, \vartheta_{t_1}, \cdots, \vartheta_{t_n}, \varphi_{t_1 1}, \cdots, \varphi_{t_n K})^T$, while the dynamic of its protein is characterized by the parameter vector $(\beta_1, \cdots, \beta_K, \theta_{t_1}, \cdots, \theta_{t_n}, \delta_{t_1 1}, \cdots, \delta_{t_n K})^T$. A clustering of these vectors can be performed to cluster peptides with similar dynamics.

1. Extracting model parameters using a classical approach:

```
> param=matrix(NA, nrow(int.ptm.prot.sel),6)
> for (i in 1:nrow(int.ptm.prot.sel)){
Int=int.ptm.prot.sel[i,]
mod <- lm(Int ~ Cond.ptmprot*Time.ptmprot*Comp.ptmprot)
param[i,]=mod$coefficients[unique(c(grep("Cond",
names(mod$coefficients)),
grep("Time",names(mod$coefficients))))]
}
> colnames(param)=names(mod$coefficients[unique(
c(grep("Cond",names(mod$coefficients)),
grep("Time",names(mod$coefficients))))])

#Extract coefficients only related to the dynamics of the
#modified peptides relatively to their protein
> param_pept=param[,grep(":Comp.ptmprot",colnames(param))]
```

2. Extracting model parameters using `limma` R package:

```
> require(limma)
> design=model.matrix(~Cond.ptmprot*Time.ptmprot*Comp.ptmprot)
> fit=eBayes(lmFit(int.ptm.prot.sel, design),robust=TRUE)
> param_LIMMA=fit$coefficients[,unique(c(grep("Cond",
names(mod$coefficients)), grep("Time",
names(mod$coefficients))))]
```

3. Choosing the clustering algorithm: Many unsupervised clustering algorithms proposed in the literature could be applied on our dataset. We propose in the next to use the classical "hard" version of the k-means algorithm (*see* **Note 19**). This algorithm depends on two main parameters that have to be shortlisted by the user: the distance measure and the number of clusters.

4. Distance measure and dissimilarity matrix: Generally, Euclidean distances or Pearson correlation-based distances between observed values are used (*see* **Note 20**). The Euclidean distance aims to measure the absolute deviations between the vector coordinates, while the Pearson correlation-based distance characterizes the linear evolution between the vector coordinates. In our case, we search to cluster parameter vectors and no dynamic is expected between these parameters. Thus, a classical Euclidean distance seems to be an appropriate choice. Once the distance chosen, a dissimilarity matrix can be computed. To this end, the `multivariance` R package [27] provides a function `fastdist` allowing fast computation of Euclidean distance matrix, which is of definite interest with respect to the data set size (several tens of thousands of peptides):

```
> require(multivariance)
> diss.mat=fastdist(param)
```

5. Determining the optimal number of clusters: Many indices can be used to determine an optimal number of clusters [38]. A classical method consists in using a Gap statistic [39]. For this, the `fviz_nbclust` function of the R package `factoextra` can be used:

```
> require(factoextra)
> gap.stat=fviz_nbclust(param,diss = diss.mat, kmeans,
nstart = 25, method = "gap_stat", nboot = 10, k.max = 30)
> gap.stat=gap.stat+labs(subtitle = "Gap statistic method")
> gap.stat
```

   Fig.3**A**-**B** display the Gap statistics in function of the number of clusters for `param` and `param_pept` vectors *see* Subheading 3.10, **Step 1**.

6. Performing the clustering: The `eclust` function of `factoextra` R package can be used for enhancing the workflow of clustering analyses and `ggplot2`-based elegant data visualization.

```
#Using a k-means clustering with 20 clusters.
> res.km=eclust(param, hc_metric = "euclidean", "kmeans",
k=20, nstart = 10, graph=TRUE)
```

   Alternatively, the k-medioids algorithm can be used using the `pam` R function from `cluster` R package, while a visualization of this clustering using Principal Component Analysis is obtained with the `fviz_cluster` function:

```
> require(cluster)
> res.pam=pam(x=diss.mat,k=20,diss=TRUE)
> res.pam$data=param
> fviz_cluster(res.pam,main = "K MEDIOIDS Clustering")
```

7. Visualization of results: There is two main ways to visualize cluster results. One way is to use dimension reduction methods as Principal Component Analysis or Multidimensional Scaling Methods. The `graph=TRUE` option of the `eclust` function or the `fviz_cluster` function can be used with this purpose. A second way is to plot the profiles of the model parameters used to cluster. The clusters associated to strong deviations from 0 of these parameters are synonym of profiles strongly discriminant, and are thus of main interest. These clusters can be highlighted by computing the average norms of parameter vectors.

   a. Estimating the average norms of parameter vectors by cluster:

```
> dataplot=data.frame(param_pept,cluster=res.pam$clustering,
peptID=1:nrow(param))
#compute the norms of parameter vectors for param_pept
> normv=apply(param_pept,1,function(x)sqrt(sum(x^2)))
#average norm for each cluster
> mc=NULL
> for (i in 1:length(levels(as.factor(dataplot$cluster)))){
   mc=c(mc,mean(normv[dataplot$cluster==i]))
}
> names(mc)=levels(as.factor(dataplot$cluster))
#rank clusters by their average norms
> mc=mc[order(-mc)]
```

b. Displaying the two clusters with the highest average norms as in Fig.3**E**:

```
> require(reshape2) #for melt
> require(ggpubr) #for facet_grid

> nb_to_plot=2
> cl_to_plot=as.numeric(names(mc)[1:nb_to_plot])
> dataplot=dataplot[dataplot$cluster%in%cl_to_plot,]
> datap=melt(dataplot,id.var=c("cluster","peptID"))

> gg= ggplot(datap, aes(x = variable, y = value) )
> gg=gg+geom_boxplot(aes(fill = variable), alpha = 0.5)
> gg=gg+facet_grid(cluster~variable,scales="free_x")
> gg=gg+scale_x_discrete(labels = "")
> gg=gg+theme(legend.position = "none")
> gg
```

3.11 Subsequent functional analysis

1. Motif enrichment analysis: Motif analysis can be used to know the main sequence of amino acids surrounding modification sites of interest. For instance, this kind of analysis can be useful to find out which kinases are involved in phosphorylation. In R, motif analysis can be done using the `rmotifx` R package [40] and the `ggseqlogo` R package can be used to visualize the results [41] (*see* **Note 21**).

2. Enrichment analysis of Gene Ontology terms and Pathways: An analysis of the enrichment of Gene Ontology terms or biological pathways referenced in databases such as KEGG or Reactome is often carried out to highlight biological processes or specific signaling pathways enriched in the lists of modified peptides of interest coming from statistical analysis. Two important points have to be kept in mind when performing these analyses in our framework:
   i) This enrichment is generally carried out in relation to a "background" using hypergeometric tests. In our context, it is very important to choose the set of modified proteins identified by mass spectrometry as background. Indeed, the enrichment protocols can display a bias in favor of enrichment for peptides with a single modification versus peptides with several modifications, or towards enrichment for hydrophobic versus hydrophilic modified peptides. Such biases will not be taken into account if the whole proteome or genome of the organism studied is used as background.
   ii) Most enrichment tools are based on gene-centric databases. However, two modified peptides of the same protein can sometimes evolve differently between two conditions, for example if the abundance of one proteoform of the protein decreases while that of another increases. In such a case, using the annotations of the reference protein without taking the modification into account may lead to falsely highlighting certain biological processes that are associated with a particular proteoform and not with another. Fortunately, recent resources have been developed to take this aspect into account [42].

3. Visualizing networks of PTMomics data: This step can be insightful to find modified proteins with known interactions. To do so, the reader can start from the statistical analyzes carried out from R, export them and then use the Cytoscape software [43] to visualize interaction networks. Two Cytoscape applications freely downloadable from

the Cytoscape App Store are particularly interesting for visualizing PTMomics data: stringApp [44] and Omics Visualizer [45]. stringApp is an application allowing to query the protein-protein interaction database of STRING [46] from Cytoscape, while Omics Visualizer allows to optimize the visualization of the obtained network. The latter is particularly suitable for viewing information on each node of the network, for instance the number of modified peptides regulated for each protein. Specific tutorials are freely available online (https://jensenlab.org/training/).

Notes

1. A number of recently proposed packages allow one to consider pipelines of analyzes performed entirely from R: for example, the `rawR` package can read RAW files from mass spectrometers [47], while the Bioconductor packages `rTandem` [48] or `MSGFplus` [49] [50] can be used to perform identification of mass spectra from these RAW files. Additionally, the `RforProteomics` package offer useful functionnalities to manipulate and visualize mass spectra [51].
2. The first line of the input file has to contain the column names of the data table. Decimal separator for quantitative values has to be the dot ".". Other R functions, such as `read.xlsx()` , from the R package `openxlsx` ; or `read_excel()` from the R package `readxl` can also be used if the file is exported from Excel.
3. These peptides are used to calculate a false discovery rate linked to the identification of peptides [52].
4. The higher the probability of localization, the more certain it is that the measured spectrum is associated with the peptide containing the modification.
5. The "proteinGroups.txt" contains intensities for the proteins using either their unmodified peptides or their modified peptides in different columns. In the case study dataset, samples enriched in phosphorylation contain "Phospho", while the ones enriched in ubiquitination contain "Ubi". However, these names can be different with other datasets depending on how they have been defined in MaxQuant or in the used software. Thus, the reader has to adapt these names in such cases. The `colnames` function can be used to check these names.
6. Similarly to previous note, the names of enriched samples can be different with other datasets depending on how they have been defined in MaxQuant or in the used software: the reader has to adapt these names in such cases.
7. These peptides are generally present because they have been identified. However, no quantified values are available because of the quantification algorithm used in MaxQuant.
8. The same approach can be used with 2, 4 or 5 samples using the `draw.pairwise.venn` , `draw.quad.venn` and `draw.quintuple.venn` functions of the `VennDiagram` package.
9. It is generally useful to convert a plot into a `ggplot` object, for example to easily include it in a pdf or powerpoint document created automatically with R Markdown or the `officer` R package.
10. This can be performed using the `qplot` function of R package `ggplot2` with

the following R code:

```
> require(reshape2)
> gradientRate=1.2;
> text=element_text(colour="black",size=8,face="bold");
> plot.j=qplot(x=Var1,y=Var2,data=melt(mat.jaccard),fill=value,
geom="tile")
> plot.j=plot.j+theme(axis.text=element_text(size=10),
axis.title=element_text(size=10,face="bold"),
axis.text.x=element_text(angle=90,vjust=1,hjust=1),legend.text=text)
> plot.j=plot.j+theme(legend.title=element_blank())
> plot.j=plot.j+labs(x="",y="")
> plot.j=plot.j+scale_fill_gradientn(colours=colorRampPalette(
c("white","lightblue","darkblue"))(101),values=c(pexp(seq(0,1,0.01),
rate=gradientRate),1),limits=c(0,1))
```

11. For this, the same approach has to be applied from vectors equal to 1 is the modified peptide is not identified and 0 if it is identified. Of note, the Jaccard index does not give the same value when evaluating mutual presence or mutual absence, such that the result of this second approach will be different from the first. An alternative to the Jaccard index consists to use the Rand index, this one will lead to a single value measuring at the same time mutual presence and absence.

12. In this way, the differences observed between the various distributions will highlight whether if the intensities measured in one sample are lower or higher than in others, without confounding effects related to different sets of peptides used to plot the distributions.

13. This can be performed similary as in Note 10, except that `data=melt(mat.cor)` in the `qplot` function. Also, we advise to choose `gradientRate=` 3.5 for correlation matrix based on quantified values to see more clearly clusters of samples in the matrix.

14. Because the Principal Component Analysis (PCA) is based on the eigendecomposition of the covariance matrix, it will summarize the same information that appears in the correlation matrix. The `prcomp` function can be used to perform PCA in R.

15. In general, it is necessary to carry out a first MS-based experiment with a minimum number of replicates per biological condition (at least 3 to be able to estimate the variance of intensities for each modified peptide in each condition). This kind of test experiment is generally intended to check that the enrichment step has worked correctly but can also be used to assess how many samples are sufficient to obtain robust statistical results in a subsequent analysis.

16. In each condition, there are therefore two possibilities: either the modified peptide is detected in one of the samples of the condition, or it is not. In the case of N conditions, $2^N - 1$ detection profiles are possible. These detection profiles should be compared with the ones associated with the corresponding proteins, which further increases the number of possible cases. Hopefully, it can generally be expected that many detection profiles will not be encountered in the experiment. It should be noted that some cases lead to not being able to conclude on the over or under abundance of the modified peptide compared to the unmodified protein. For example, if they

are detected in the same condition and not in another, nothing can be concluded by comparing these conditions.
17. For example, with conditions representing different time stamps, a value can be deduced by applying methods developped for time series data [53]. In the case where no value can be inferred in the condition from observed ones, then an analysis of the detection profiles can be conducted to highlight interesting modified peptides.
18. Here, the "*" operator is used to specify that interaction parameters has to be included in the model. If one does not want interactor parameters, one uses only the "+" operator.
19. Either hierarchical clustering methods or partitional clustering methods (such as the k-means algorithm) can be used [54]. Moreover, either hard clustering versions of these methods, assigning a unique cluster for each peptide, or fuzzy clustering versions, assigning probabilities of belonging to each cluster for each peptide could be used [55]. The main advantage of hierarchical methods is that they does not need to specify a number of clusters. Their results are generally represented in the form of a dendrogram. However, in our context, it is preferable to cluster the peptides into a finite number of clusters in order to facilitate the subsequent biological interpretation of the results. Partitional clustering methods seem thus more appropriate. The most famous one is the k-means algorithm. The fuzzy version of this algorithm has serious limitations in high dimensional spaces, as it is generally the case in peptidomics datasets [56].
20. However, alternative distances can also be investigated as, in case of time series, Dynamic Time Warping distances [57] or Short time series distances [58].
21. Pre-alignment of the modified peptides around the detected sites of modification has to be performed before to use these packages.

Acknowledgements: The author wants to acknowledge Mariette Matondo, Thibaut Douché, Thibault Chaze and Magalie Duchateau of the Proteomics platform of the Institut Pasteur for fruitful discussions about proteomics and phosphoproteomics.

References

[1] Witze ES, Old WM, Resing KA, Ahn NG (2007) Mapping protein post-translational modifications with mass spectrometry. Nature methods 4(10):798–806, https://doi.org/10.1038/nmeth1100

[2] Zhao Y, Jensen ON (2009) Modification-specific proteomics: strategies for characterization of post-translational modifications using enrichment techniques. Proteomics 9(20):4632–4641, https://doi.org/10.1002/pmic.200900398

[3] Pieroni L, Iavarone F, Olianas A, Greco V, Desiderio C, Martelli C, Manconi B, Sanna MT, Messana I, Castagnola M, et al. (2020) Enrichments of post-translational modifications in proteomic studies. Journal of separation science 43(1):313–336, https://doi.org/10.1002/jssc.201900804

[4] Ficarro SB, Adelmant G, Tomar MN, Zhang Y, Cheng VJ, Marto JA (2009) Magnetic bead processor for rapid evaluation and optimization of parameters for phosphopeptide

enrichment. Analytical chemistry 81(11):4566–4575, https://doi.org/10.1021/ac9004452

[5] Pinkse MW, Lemeer S, Heck AJ (2011) A protocol on the use of titanium dioxide chromatography for phosphoproteomics. In: Gel-Free Proteomics, Springer, pp 215–228, https://doi.org/10.1007/978-1-61779-148-2_14

[6] Udeshi ND, Svinkina T, Mertins P, Kuhn E, Mani D, Qiao JW, Carr SA (2013) Refined preparation and use of anti-diglycine remnant (k-ε-gg) antibody enables routine quantification of 10,000 s of ubiquitination sites in single proteomics experiments. Molecular & Cellular Proteomics 12(3):825–831, https://doi.org/10.1074/mcp.O112.027094

[7] Carlson SM, Moore KE, Green EM, Martín GM, Gozani O (2014) Proteome-wide enrichment of proteins modified by lysine methylation. Nature protocols 9(1):37–50, https://doi.org/10.1038/nprot.2013.164

[8] Kim SC, Sprung R, Chen Y, Xu Y, Ball H, Pei J, Cheng T, Kho Y, Xiao H, Xiao L, et al. (2006) Substrate and functional diversity of lysine acetylation revealed by a proteomics survey. Molecular cell 23(4):607–618, https://doi.org/10.1016/j.molcel.2006.06.026

[9] Mertins P, Qiao JW, Patel J, Udeshi ND, Clauser KR, Mani D, Burgess MW, Gillette MA, Jaffe JD, Carr SA (2013) Integrated proteomic analysis of post-translational modifications by serial enrichment. Nature methods 10(7):634, https://doi.org/10.1038/nmeth.2518

[10] Chalkley RJ, Clauser KR (2012) Modification site localization scoring: strategies and performance. Molecular & Cellular Proteomics 11(5):3–14, https://doi.org/10.1074/mcp.R111.015305

[11] Tyanova S, Temu T, Cox J (2016) The maxquant computational platform for mass spectrometry-based shotgun proteomics. Nature protocols 11(12):2301, https://doi.org/10.1038/nprot.2016.136

[12] Valot B, Langella O, Nano E, Zivy M (2011) Masschroq: a versatile tool for mass spectrometry quantification. Proteomics 11(17):3572–3577, https://doi.org/10.1002/pmic.201100120

[13] Bouyssié D, Hesse AM, Mouton-Barbosa E, Rompais M, Macron C, Carapito C, Gonzalez de Peredo A, Couté Y, Dupierris V, Burel A, et al. (2020) Proline: an efficient and user-friendly software suite for large-scale proteomics. Bioinformatics 36(10):3148–3155, https://doi.org/10.1093/bioinformatics/btaa118

[14] Gentleman R (2008) R programming for bioinformatics. CRC Press, https://doi.org/10.18637/jss.v029.b08

[15] Chambers J (2008) Software for data analysis: programming with R. Springer Science & Business Media, https://doi.org/10.1007/978-0-387-75936-4

[16] Millot G (2011) Comprendre et réaliser les tests statistiques à l'aide de r. Brussels: De Boeck, ISBN 9782807302914

[17] Chen H, Boutros PC (2011) Venndiagram: a package for the generation of highly-customizable venn and euler diagrams in r. BMC bioinformatics 12(1):1–7, https://doi.org/10.1186/1471-2105-12-35

[18] Conway JR, Lex A, Gehlenborg N (2017) Upsetr: an r package for the visualization of intersecting sets and their properties. Bioinformatics 33(18):2938–2940, https://doi.org/10.1093/bioinformatics/btx364

[19] Wickham H (2016) ggplot2: elegant graphics for data analysis. springer, https://doi.org/10.1080/15366367.2019.1565254

[20] de Vries A, Ripley BD (2020) ggdendro: Create dendrograms and tree diagrams using 'ggplot2'. R package version 0122 https://cran.r-project.org/web/packages/ggdendro/index.html

[21] Wilke CO (2021) ggridges: Ridgeline plots in 'ggplot2'. R package version 053 https://cran.r-project.org/web/packages/ggridges/index.html

[22] Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015) limma powers differential expression analyses for rna-sequencing and microarray studies. Nucleic acids research 43(7):e47–e47, https://doi.org/10.1093/nar/gkv007

[23] Giai Gianetto Q, Combes F, Ramus C, Bruley C, Couté Y, Burger T (2016) Calibration plot for proteomics: A graphical tool to visually check the assumptions underlying fdr control in quantitative experiments. Proteomics 16(1):29–32, https://doi.org/10.1002/pmic.201500189

[24] Liu P, Hwang JG (2007) Quick calculation for sample size while controlling false discovery rate with application to microarray analysis. Bioinformatics 23(6):739–746, https://doi.org/10.1093/bioinformatics/btl664

[25] Gianetto QG, Wieczorek S, Couté Y, Burger T (2020) A peptide-level multiple imputation strategy accounting for the different natures of missing values in proteomics data. bioRxiv https://doi.org/10.1101/2020.05.29.122770

[26] Fox J, Weisberg S, Adler D, Bates D, Baud-Bovy G, Ellison S, Firth D, Friendly M, Gorjanc G, Graves S, et al. (2020) car: Companion to applied regression. R package version 30-10 https://cran.r-project.org/web/packages/car/index.html

[27] Böttcher B (2020) Copula versions of distance multivariance and dhsic via the distributional transform–a general approach to construct invariant dependence measures. Statistics pp 1–18, https://doi.org/10.1080/02331888.2020.1748029

[28] Kassambara A, Mundt F (2020) factoextra: Extract and visualize the results of multivariate data analyses. R package version 107 https://cran.r-project.org/web/packages/factoextra/index.html

[29] Wickham H (2020) reshape2: Flexibly reshape data: a reboot of the reshape package. R package version 144 https://cran.r-project.org/web/packages/reshape2/index.html

[30] Kassambara A (2020) ggpubr:"ggplot2" based publication ready plots. https://cran.r-project.org/web/packages/ggpubr/index.html

[31] Stukalov A, Girault V, Grass V, Bergant V, Karayel O, Urban C, Haas DA, Huang Y, Oubraham L, Wang A, et al. (2020) Multi-level proteomics reveals host-perturbation strategies of sars-cov-2 and sars-cov. BioRxiv https://doi.org/10.1101/2020.06.17.156455

[32] Wieczorek S, Gianetto QG, Burger T (2019) Five simple yet essential steps to correctly estimate the rate of false differentially abundant proteins in mass spectrometry analyses. Journal of proteomics 207:103441, https://doi.org/10.1016/j.jprot.2019.103441

[33] Pounds S, Cheng C (2006) Robust estimation of the false discovery rate. Bioinformatics 22(16):1979–1987, https://doi.org/10.1093/bioinformatics/btl328

[34] Kauko O, Laajala TD, Jumppanen M, Hintsanen P, Suni V, Haapaniemi P, Corthals G, Aittokallio T, Westermarck J, Imanishi SY (2015) Label-free quantitative phosphoproteomics with novel pairwise abundance normalization reveals synergistic ras and cip2a signaling. Scientific reports 5:13099, https://doi.org/10.1038/srep13099

[35] Saraei S, Suomi T, Kauko O, Elo LL (2018) Phosphonormalizer: an r package for normalization of ms-based label-free phosphoproteomics. Bioinformatics 34(4):693–694, https://doi.org/10.1093/bioinformatics/btx573

[36] Wieczorek S, Combes F, Lazar C, Giai Gianetto Q, Gatto L, Dorffer A, Hesse AM, Coute Y, Ferro M, Bruley C, Burger T (2017) Dapar & prostar: software to perform statistical analyses in quantitative discovery proteomics. Bioinformatics 33(1):135–136, https://doi.org/10.1093/bioinformatics/btw580

[37] Lazar C, Gatto L, Ferro M, Bruley C, Burger T (2016) Accounting for the multiple natures of missing values in label-free quantitative proteomics data sets to compare imputation strategies. Journal of proteome research 15(4):1116–1125, https://doi.org/10.1021/acs.jproteome.5b00981

[38] Charrad M, Ghazzali N, Boiteau V, Niknafs A (2014) Nbclust: An r package for determining the relevant number of clusters in a data set. Journal of statistical software 61:1–36, https://doi.org/10.18637/jss.v061.i06

[39] Tibshirani R, Walther G, Hastie T (2001) Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63(2):411–423, https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00293

[40] Wagih O, Sugiyama N, Ishihama Y, Beltrao P (2016) Uncovering phosphorylation-based specificities through functional interaction networks. Molecular & Cellular Proteomics 15(1):236–245, https://doi.org/10.1074/mcp.M115.052357

[41] Wagih O (2017) ggseqlogo: a versatile r package for drawing sequence logos. Bioinformatics 33(22):3645–3647, https://doi.org/10.1093/bioinformatics/btx469

[42] Krug K, Mertins P, Zhang B, Hornbeck P, Raju R, Ahmad R, Szucs M, Mundt F, Forestier D, Jane-Valbuena J, et al. (2019) A curated resource for phosphosite-specific signature analysis. Molecular & cellular proteomics 18(3):576–593, https://doi.org/10.1074/mcp.TIR118.000943

[43] Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome research 13(11):2498–2504, https://doi.org/10.1101/gr.1239303

[44] Doncheva NT, Morris JH, Gorodkin J, Jensen LJ (2018) Cytoscape stringapp: network analysis and visualization of proteomics data. Journal of proteome research 18(2):623–632, https://doi.org/10.1021/acs.jproteome.8b00702

[45] Legeay M, Doncheva NT, Morris JH, Jensen LJ (2020) Visualize omics data on networks with omics visualizer, a cytoscape app. F1000Research 9, https://doi.org/10.12688/f1000research.22280.2

[46] Szklarczyk D, Gable AL, Lyon D, Junge A, Wyder S, Huerta-Cepas J, Simonovic M, Doncheva NT, Morris JH, Bork P, et al. (2019) String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic acids research 47(D1):D607–D613, https://doi.org/10.1093/nar/gky1131

[47] Kockmann T, Panse C (2020) rawr-direct access to raw mass spectrometry data in r. bioRxiv https://doi.org/10.1101/2020.10.30.362533

[48] Fournier F, Joly Beauparlant C, Paradis R, Droit A (2014) rtandem, an r/bioconductor package for ms/ms protein identification. Bioinformatics 30(15):2233–2234, https://doi.org/10.1093/bioinformatics/btu178

[49] Kim S, Pevzner PA (2014) Ms-gf+ makes progress towards a universal database search tool for proteomics. Nature communications 5:5277, https://doi.org/10.1038/ncomms6277

[50] Pedersen TL (2020) Msgfplus: An interface between r and ms-gf+. R package version 1240 http://www.bioconductor.org/packages/release/bioc/html/MSGFplus.html

[51] Gatto L, Breckels LM, Naake T, Gibb S (2015) Visualization of proteomics data using r and bioconductor. Proteomics 15(8):1375–1389, https://doi.org/10.1002/pmic.201400392

[52] Couté Y, Bruley C, Burger T (2020) Beyond target–decoy competition: Stable validation of peptide and protein identifications in mass spectrometry-based discovery proteomics. Analytical Chemistry 92(22):14898–14906, https://doi.org/10.1021/acs.analchem.0c00328

[53] Pratama I, Permanasari AE, Ardiyanto I, Indrayani R (2016) A review of missing values handling methods on time-series data. In: 2016 International Conference on Information Technology Systems and Innovation (ICITSI), IEEE, pp 1–6, https://doi.org/10.1109/ICITSI.2016.7858189

[54] Gan G, Ma C, Wu J (2020) Data clustering: theory, algorithms, and applications. SIAM, https://doi.org/10.1137/1.9780898718348

[55] Schwämmle V, Jensen ON (2018) Vsclust: feature-based variance-sensitive clustering of omics data. Bioinformatics 34(17):2965–2972, https://doi.org/10.1093/bioinformatics/bty224

[56] Winkler R, Klawonn F, Kruse R (2011) Fuzzy c-means in high dimensional spaces. International Journal of Fuzzy System Applications (IJFSA) 1(1):1–16, https://doi.org/10.4018/IJFSA.2011010101

[57] Giorgino T, et al. (2009) Computing and visualizing dynamic time warping alignments in r: the dtw package. Journal of statistical Software 31(7):1–24, https://doi.org/10.18637/jss.v031.i07

[58] Mori U, Mendiburu A, Lozano JA (2016) Distance measures for time series in r: the tsdist package. The R Journal 8(2):451, https://doi.org/10.32614/RJ-2016-058

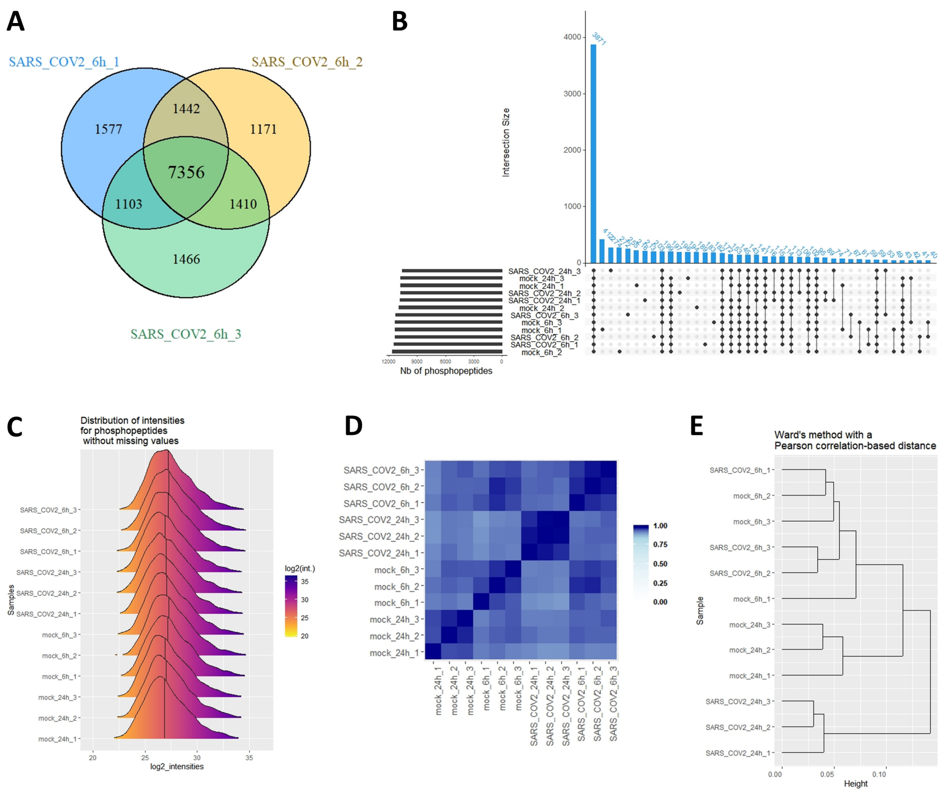

Figure 1: Graphs to check the reproducibility of experiments. **A**: Venn diagram checking the reproducibility of modified peptides identified in SARS-CoV-2 samples at 6h. **B**: UpSet graph checking the reproducibility of modified peptides identified in all samples (3871 are found in common). **C**: Estimated distributions of log2(intensities) of modified peptides found in all samples. This makes it possible to detect shifts in all the intensity values of one or more samples. **D**: Pearson correlation matrix between all samples using shades of blue. Darker blues (higher correlation values) should appear between replicates of the same condition. **E**: Hierarchical clustering of samples from Pearson correlation values. Samples of a same condition should cluster together, and not with the one of another condition if the conditions proteomes are not close. If they are, clustering may have a hard time distinguishing them, but they will appear close (with darker blue) in the correlation matrix plotted in **D**.

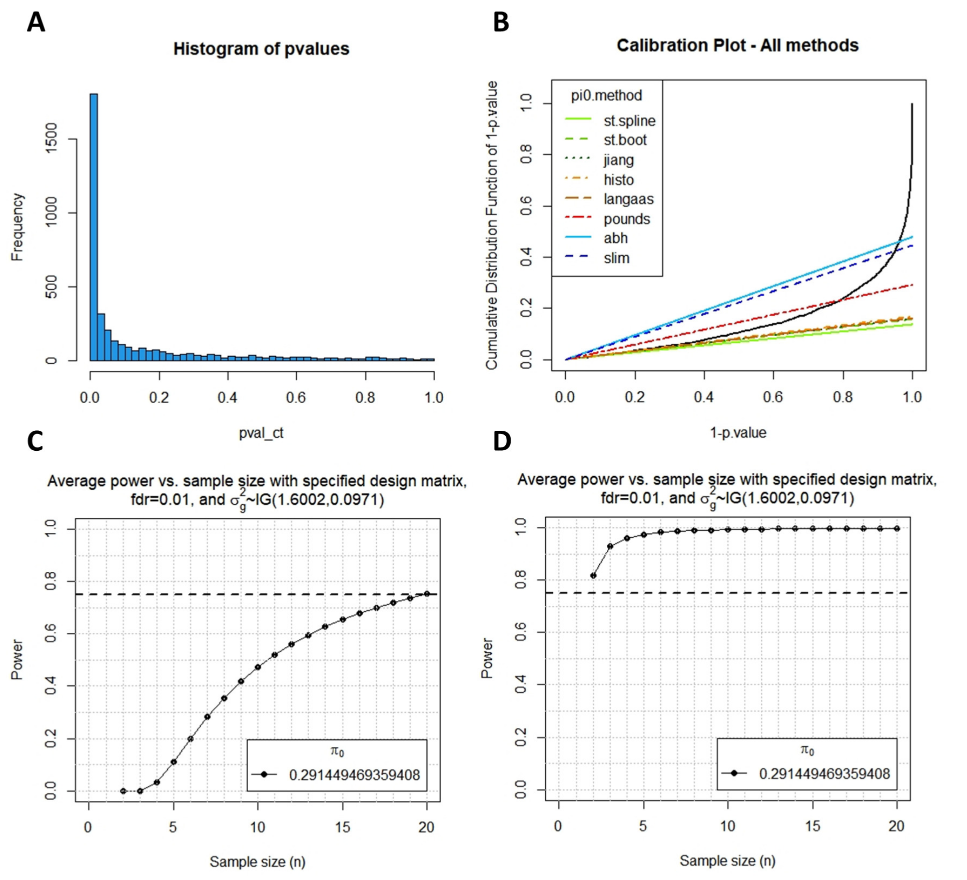

Figure 2: **A**: Histogram of pvalues related to the used test. **B**: Calibration plot to choose a method to estimate the proportion of true null hypotheses. **C**: Average power in function of the sample size with eps=0.1 and the proportion of true null hypotheses estimated by the method of Pounds and Cheng [33]. **D**: Average power in function of the sample size with eps=0.5 and the proportion of true null hypotheses estimated by the method of Pounds and Cheng [33].

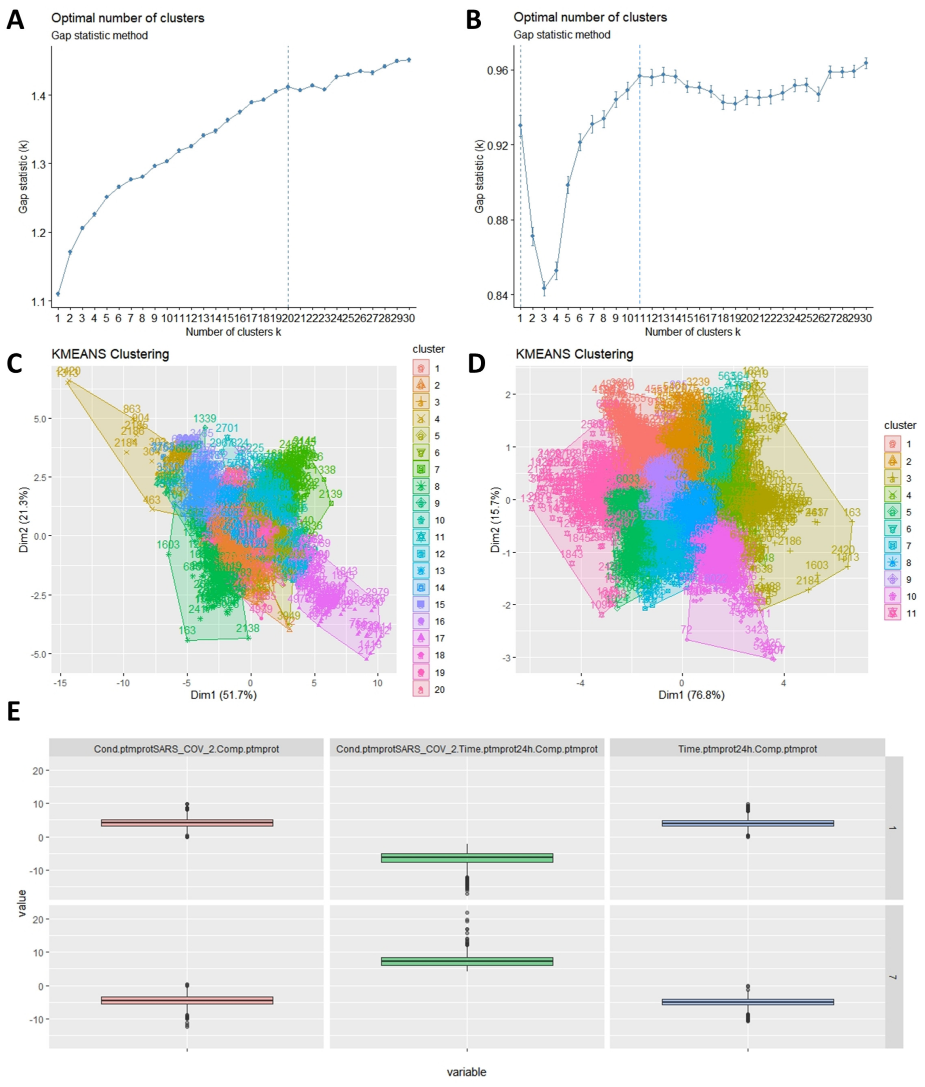

Figure 3: **A**: Gap statistic in function of the number of clusters for param vectors, *see* Subheading 3.10, **Step 1**. **B**: Gap statistic in function of the number of clusters for param_pept vectors, *see* Subheading 3.10, **Step 1**. **C**: Visualization using PCA of 20 clusters of param vectors found with kmeans. **D**: Visualization using PCA of 11 clusters of param_pept vectors found with kmeans. **E**: Profiles of parameters for the clusters with the highest average norms for param_pept vectors. Here, param_pept contains coefficients only related to the dynamics of the modified peptides relatively to their protein. The cluster 1 is characterized by peptides with positive coefficients for Time 24h and SARS-Cov-2 conditions, as well as negative coefficients for their interaction.