PEC 1: Informe Análisis de datos de Fosfoptoteómica

Gabriel Manzano Reche

01 de noviembre de 2024

Índice

1. Introducción

Para esta PEC1 se ha escogido el dataset propuesto llamado **2018-Phosphoproteomics**, el cuál se ha descargado del repositorio proporcionado en el enunciado.

Los daros de este estudio se han obtenido a partir de un experimento de fosfoproteómica. El experimento ha analizado (3 + 3) modelos PDX de dos subtipos diferentes utilizando muestras enriquecidas en fosfopéptidos. Se realizó un análisis LC-MS con 2 réplicas técnicas para cada muestra. El conjunto de resultados consiste en abundancias normalizadas de señales MS para aproximadamente 1400 fosfopéptidos. El objetivo del análisis es identificar fosfopéptidos que permitan diferenciar los dos grupos de tumores. Esto debe realizarse mediante análisis estadístico y visualización. Los datos se han proporcionado en un archivo de Excel: TIO2+PTYR-human-MSS+MSIvsPD.XLSX.

Los grupos están definidos como:

- Grupo MSS: Muestras M1, M5 y T49
- Grupo PD: Muestras M42, M43 y M64

Cada muestra cuenta con dos réplicas técnicas. La primera columna, SequenceModification, contiene los valores de abundancia para los distintos fosfopéptidos. Las demás columnas pueden omitirse.

Objetivos del Estudio

- Identificar diferencias entre los dos subtipos (MSS y PD)
- Explorar patrones de fosforilación en cada subtipo
- Observar si existe variabilidad entre réplicas

Limitaciones del estudio

- Tamaño de muestra reducido. Sólo se disponen de tres muestras con dos réplicas cada una para cada subtipo.
- La propia limitación de la técnica LC-MS: esta técnica puede no capturar todos los fosfopéptidos presentes.
- Los modelos pueden presentar variabilidad biológica, lo que puede influir en los resultados.

2. Carga de los datos

```
if (!(require(limma))){
  source("http://bioconductor.org/biocLite.R")
  biocLite("limma")
}
```

```
## Loading required package: limma
```

```
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
```

```
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following object is masked from 'package:limma':
##
##     plotMA

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows
```

```
## Loading required package: GenomeInfoDb

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
library(readxl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::%within%() masks IRanges::%within%()
## x dplyr::collapse()     masks IRanges::collapse()
## x dplyr::combine()      masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count()        masks matrixStats::count()
## x dplyr::desc()         masks IRanges::desc()
## x tidyr::expand()       masks S4Vectors::expand()
## x dplyr::filter()       masks stats::filter()
## x dplyr::first()        masks S4Vectors::first()
## x dplyr::lag()          masks stats::lag()
## x ggplot2::Position()   masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()       masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()       masks S4Vectors::rename()
## x lubridate::second()   masks S4Vectors::second()
## x lubridate::second<-() masks S4Vectors::second<-()
## x dplyr::slice()        masks IRanges::slice()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Carga de los datos de la hoja 1
original_data <- read_excel("TIO2+PTYR-human-MSS+MSIvsPD.XLSX")
head(original_data)
```

```
## # A tibble: 6 x 18
##   SequenceModifications   Accession Description Score M1_1_MSS M1_2_MSS M5_1_MSS
##   <chr>                   <chr>     <chr>       <dbl>    <dbl>    <dbl>    <dbl>
## 1 LYPELSQYMGLSLNEEEIR[2]~ O00560    Syntenin-1~  48.1     24.3   44476.       0
## 2 VDKVIQAQTAFSANPANPAILS~ O00560    Syntenin-1~  67.0      0     43139.    2102.
```

```
## 3 VIQAQTAFSANPANPAILSEAS~ O00560    Syntenin-1~  77.7   3413.   172143.   77323.
## 4 HADAEMTGYVVTR[6] Oxida~ O15264    Mitogen-ac~  44.9 220431.   145657.  104288.
## 5 HADAEMTGYVVTR[9] Phosp~ O15264    Mitogen-ac~  67.4  18255.     8530.   35956.
## 6 STGPGASLGTGYDR[12] Pho~ O15551    Claudin-3 ~  63.7 644513.   261938.  187023.
## # i 11 more variables: M5_2_MSS <dbl>, T49_1_MSS <dbl>, T49_2_MSS <dbl>,
## #   M42_1_PD <dbl>, M42_2_PD <dbl>, M43_1_PD <dbl>, M43_2_PD <dbl>,
## #   M64_1_PD <dbl>, M64_2_PD <dbl>, CLASS <chr>, PHOSPHO <chr>
```

```
# Carga de los datos de la hoja 1
targets <- read_excel("TIO2+PTYR-human-MSS+MSIvsPD.XLSX", sheet = 2)
```

```
## New names:
## * `Sample` -> `Sample...1`
## * `Sample` -> `Sample...2`
```

```
head(targets)
```

```
## # A tibble: 6 x 4
##   Sample...1 Sample...2 Individual Phenotype
##   <chr>      <chr>           <dbl> <chr>
## 1 M1_1       M1                  1 MSS
## 2 M1_2       M1                  1 MSS
## 3 M5_1       M5                  2 MSS
## 4 M5_2       M5                  2 MSS
## 5 T49_1      T49                 3 MSS
## 6 T49_2      T49                 3 MSS
```

```
# Nombres de las columnas
colnames(original_data)
```

```
##  [1] "SequenceModifications" "Accession"             "Description"
##  [4] "Score"                 "M1_1_MSS"              "M1_2_MSS"
##  [7] "M5_1_MSS"              "M5_2_MSS"              "T49_1_MSS"
## [10] "T49_2_MSS"             "M42_1_PD"              "M42_2_PD"
## [13] "M43_1_PD"              "M43_2_PD"              "M64_1_PD"
## [16] "M64_2_PD"              "CLASS"                 "PHOSPHO"
```

Como se puede ver de la columna 5 a la 16, se encuentran las abundancias. Estás se almacenarán en un nuevo dataframe llamado abundancias y se le asignarán el nombre de los fosfopéptidos a las filas, los cuales se encuentran en la columna Accesion. Por otro lado, en la columna SequenceModifications y Description se observa metadata, la cual se incluirá en el objeto *SummarizeData*.

3. Seleccionar columna de metadatos y datos

```
# Seleccionar las columnas de abundancia y almacenarlas en una variable
abundancias <- original_data %>% select(5:16)
abundancias <- as.data.frame(abundancias)

# Asignarle nuevos nombres a las columnas de la matriz de abundancia
accesion_rownames <- make.names(original_data$Accession, unique = TRUE)
rownames(abundancias) <- accesion_rownames
head(abundancias)
```

```
##              M1_1_MSS   M1_2_MSS   M5_1_MSS   M5_2_MSS    T49_1_MSS  T49_2_MSS
## O00560       24.29438  44475.964      0.000   6269.141    1135.8169   21933.90
## O00560.1      0.00000  43138.904   2102.056  50355.051     248.9275    3239.16
```

```
## O00560.2   3412.60332 172143.040  77323.019 307637.429   98442.2773  192982.37
## O15264   220431.17880 145656.887 104287.815  75887.365  773377.4981  481165.54
## O15264.1  18254.77813   8529.755  35955.901  44102.316   57145.1682   34638.01
## O15551   644513.31840 261938.025 187023.484 124867.715 4487443.6920 2572575.27
##             M42_1_PD   M42_2_PD     M43_1_PD    M43_2_PD    M64_1_PD
## O00560         0.000       0.00     772.9056    2136.746    1820.724
## O00560.1    1315.904       0.00       0.0000       0.000       0.000
## O00560.2   24851.344   16547.95    5565.2821       0.000    3264.563
## O15264   1027196.292 1163747.38 4080239.1820 4885818.113 3093786.793
## O15264.1   21231.256   49499.70  666107.0448  379313.615  255792.117
## O15551    535809.187  434645.89   91361.8781   65997.913  243250.439
##              M64_2_PD
## O00560      1727.9098
## O00560.1     892.3565
## O00560.2    5901.9577
## O15264   2759104.5440
## O15264.1  579765.0018
## O15551    206632.6444
```

A continuación, se definen los metadatos, que incluyen información de las muestras como grupo y réplica, y detalles de los fosfopéptidos como modificaciones de secuencia y descripción.

En primer lugar, se definen los metadatos de las muestras que se almacenarán en la variable sample_info. Esta variable contiene el SampleID de cada muestra, el grupo y el número de réplica.

```
# Crear el DataFrame de metadatos de las muestras
sample_info <- DataFrame(
    SampleID = colnames(abundancias),
    Group = c(rep("MSS", 6), rep("PD", 6)),
    Replicate = rep(c(1, 2), times = 6)
)
```

A continuación se definen los metadatos de los fosfopéptidos, una descripción de estos y la modificación en su secuencia.

```
# Crear el DataFrame con metadatos de los fosfopéptidos
row_data <- DataFrame(
    SequenceModifications = original_data$SequenceModifications,
    Description = original_data$Description,
    row.names = rownames(abundancias)
)
```

3. Crear y explorar el objeto *SummarizedData*

Finalmente, se crea el objeto *SummarizedData* que permite organizar la matriz de abundancia junto con los metadatos en un solo objeto, facilitando el análisis de los datos.

```
# Crear el objeto SummarizedExperiment
se <- SummarizedExperiment(
    assays = list(counts = as.matrix(abundancias)),  # Matriz de abundancia
    colData = sample_info,  # Metadata de las muestras
    rowData = row_data  # Metadata de los fosfopéptidos
)

se
```

```
## class: SummarizedExperiment
```

```
## dim: 1438 12
## metadata(0):
## assays(1): counts
## rownames(1438): O00560 O00560.1 ... Q13283.1 Q9NYF8.12
## rowData names(2): SequenceModifications Description
## colnames(12): M1_1_MSS M1_2_MSS ... M64_1_PD M64_2_PD
## colData names(3): SampleID Group Replicate
```

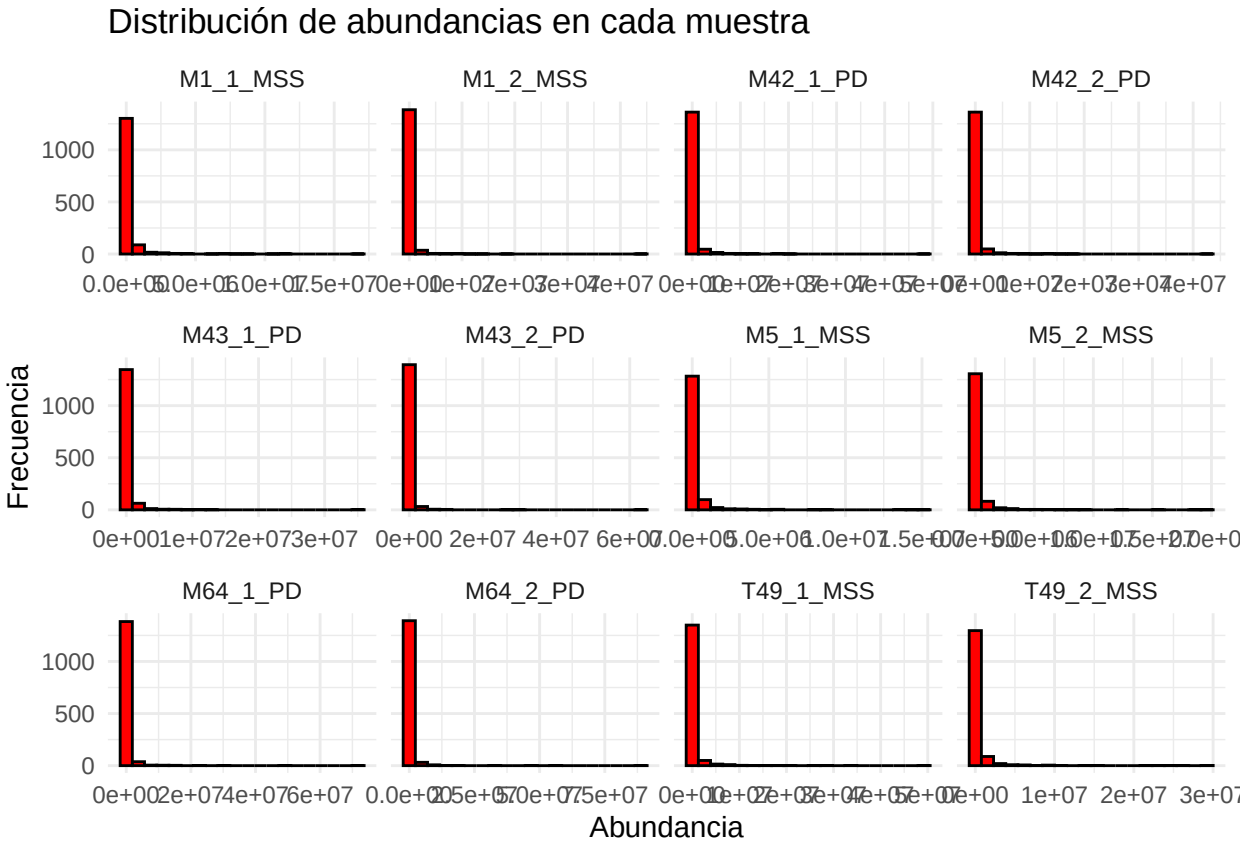
4. Análisis de los datos

A continuación, se procederá al análisis de los datos con el objetivo de identificar diferencias significativas entre los dos subtipos de muestras presentes en el experimento.

```
# Resumen estadístico de la matriz de abundancia
summary(assay(se, "counts"))
```

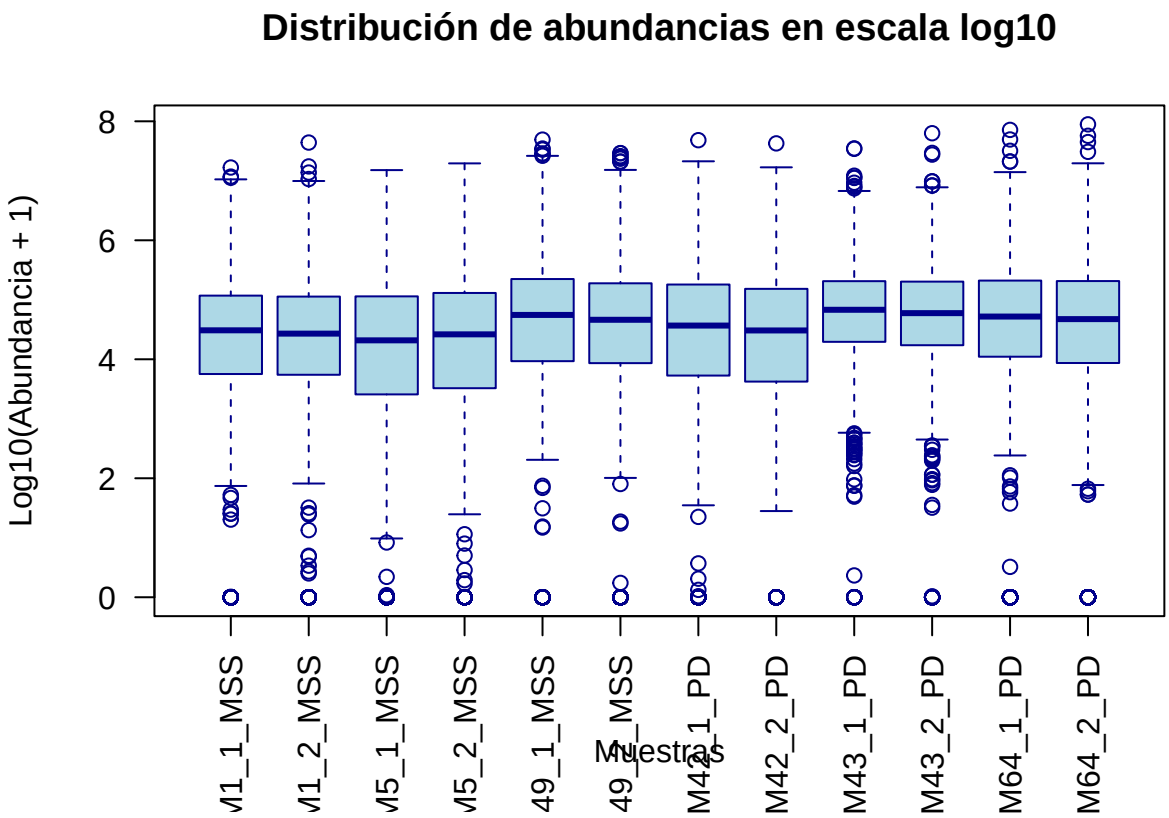
```
##     M1_1_MSS           M1_2_MSS           M5_1_MSS           M5_2_MSS
##  Min.   :       0   Min.   :       0   Min.   :       0   Min.   :       0
##  1st Qu.:    5653   1st Qu.:    5497   1st Qu.:    2573   1st Qu.:    3273
##  Median :   30682   Median :   26980   Median :   20801   Median :   26241
##  Mean   :  229841   Mean   :  253151   Mean   :  232967   Mean   :  261067
##  3rd Qu.:  117373   3rd Qu.:  113004   3rd Qu.:  113958   3rd Qu.:  130132
##  Max.   :16719906   Max.   :43928481   Max.   :15135169   Max.   :19631820
##    T49_1_MSS          T49_2_MSS          M42_1_PD           M42_2_PD
##  Min.   :       0   Min.   :       0   Min.   :       0   Min.   :       0
##  1st Qu.:    9306   1st Qu.:    8611   1st Qu.:    5341   1st Qu.:    4216
##  Median :   55641   Median :   46110   Median :   36854   Median :   30533
##  Mean   :  542449   Mean   :  462616   Mean   :  388424   Mean   :  333587
##  3rd Qu.:  223103   3rd Qu.:  189141   3rd Qu.:  180252   3rd Qu.:  152088
##  Max.   :49218872   Max.   :29240206   Max.   :48177680   Max.   :42558111
##     M43_1_PD           M43_2_PD           M64_1_PD           M64_2_PD
##  Min.   :       0   Min.   :       0   Min.   :       0   Min.   :       0
##  1st Qu.:   19641   1st Qu.:   17299   1st Qu.:   11038   1st Qu.:    8660
##  Median :   67945   Median :   59607   Median :   52249   Median :   47330
##  Mean   :  349020   Mean   :  358822   Mean   :  470655   Mean   :  484712
##  3rd Qu.:  205471   3rd Qu.:  201924   3rd Qu.:  209896   3rd Qu.:  206036
##  Max.   :35049402   Max.   :63082982   Max.   :71750330   Max.   :88912734
```

```
# Convertir los datos de abundancia a formato largo
abundancias_long <- abundancias %>%
  pivot_longer(cols = everything(), names_to = "Muestra", values_to = "Abundancia")

# Crear el histograma para cada muestra
ggplot(abundancias_long, aes(x = Abundancia)) +
  geom_histogram(bins = 20, fill = "red", color = "black") +
  facet_wrap(~ Muestra, scales = "free_x") +
  labs(title = "Distribución de abundancias en cada muestra",
       x = "Abundancia",
       y = "Frecuencia") +
  theme_minimal()
```



Como se puede apreciar en los histogramas de arriba, todas las muestras están sesgadas hacia la izquierda. Esto sugiere que puede ser útil normalizar los datos (aplicando log10) para visualizar mejor las diferencias entre las muestras. A continuación, se viualizarán las datos normalizados en base 10

```
# Visualizar distribuciones de abundancia en escala logarítmica
boxplot(log10(assay(se, "counts") + 1),
        main = "Distribución de abundancias en escala log10",
        xlab = "Muestras",
        ylab = "Log10(Abundancia + 1)",
        las = 2,
        col = "lightblue",
        border = "darkblue")
```



En este gráfico se puede observar la distribución de las abundacias de péptidos en las muestras en escala logarítmica, donde no se observa una separación clara entre las muestras de los grupos MSS y PD.

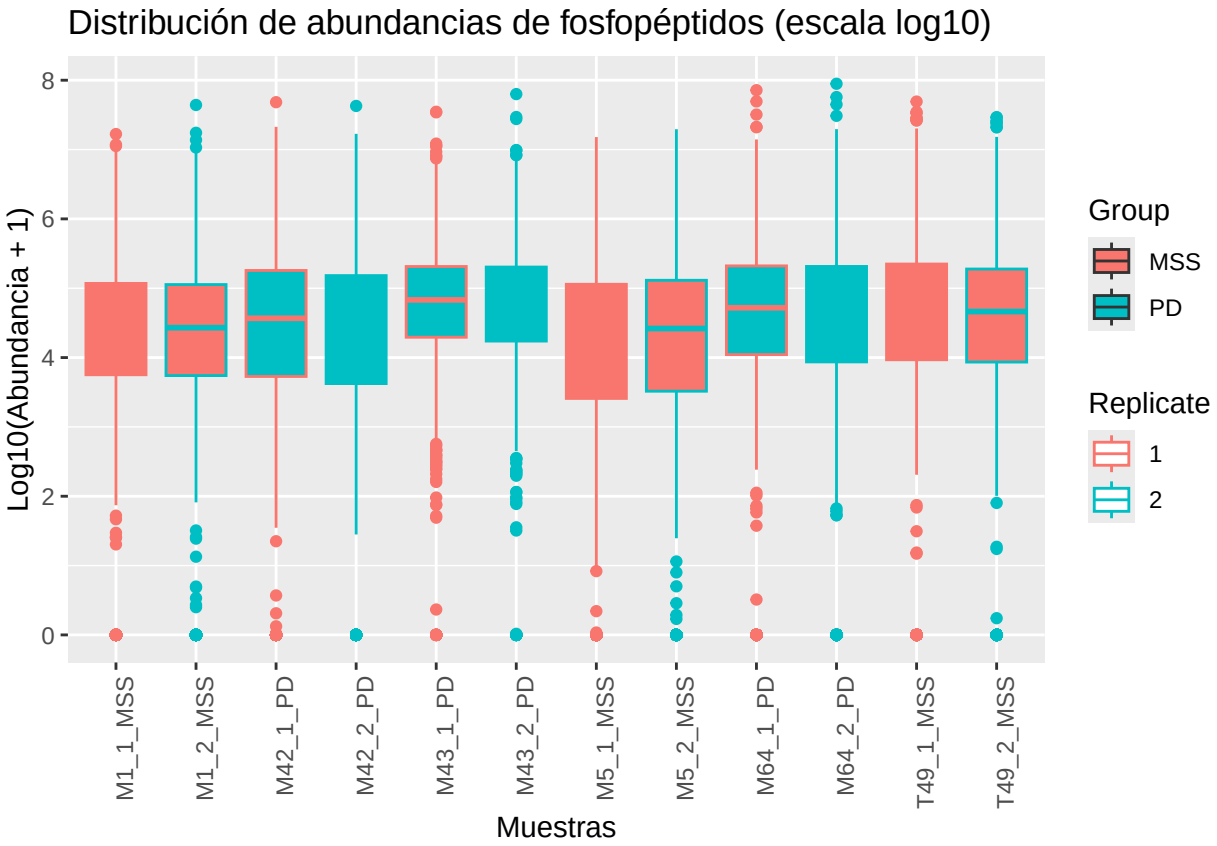
A continuación, se normalizarán los datos de la matriz de abundancias y se extraerá el grupo y réplica a la que pertenece cada muestra, con el objetivo de identificar diferencias significativas entre grupos o entre réplicas.

```
# Extraer la matriz de abundancia del objeto SummarizedExperiment
abundancias <- assay(se, "counts")

# Convertir la matriz de abundancia a formato largo y aplicar transformación logarítmica
logDat <- as.data.frame(abundancias) %>%
  pivot_longer(cols = everything(), names_to = "Muestra", values_to = "Abundancia") %>%
  mutate(log_abundance = log10(Abundancia + 1))

# Extraer información de Grupo y Réplica desde el nombre de la muestra
covs <- str_split(logDat$Muestra, "_", simplify = TRUE)
colnames(covs) <- c("Sample", "Replicate", "Group")

# Añadir las nuevas columnas (Sample, Replicate, Group) al dataframe logDat
logDat2 <- cbind(logDat, covs)

# Verificar la estructura de los datos transformados
head(logDat2)
```

```
##     Muestra  Abundancia log_abundance Sample Replicate Group
## 1  M1_1_MSS    24.29438      1.403024     M1         1   MSS
```

```
## 2  M1_2_MSS 44475.96381      4.648135     M1         2   MSS
## 3  M5_1_MSS     0.00000      0.000000     M5         1   MSS
## 4  M5_2_MSS  6269.14095      3.797277     M5         2   MSS
## 5 T49_1_MSS  1135.81687      3.055691    T49         1   MSS
## 6 T49_2_MSS 21933.89963      4.341136    T49         2   MSS
```

```
# Crear el boxplot con ggplot2 usando el grupo y la réplica como variables estéticas
ggplot(logDat2, aes(x = Muestra, y = log_abundance, fill = Group, colour = Replicate)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  labs(title = "Distribución de abundancias de fosfopéptidos (escala log10)",
       x = "Muestras",
       y = "Log10(Abundancia + 1)")
```



Este gráfico muestra la distribución de abundacias de fosfopéptidos en escala logarítmo para cada muestra, representando también el grupo y la réplica a la que pertenece esa muestra. No se aprecia diferencia significativa entre los grupos ni entre las réplicas.

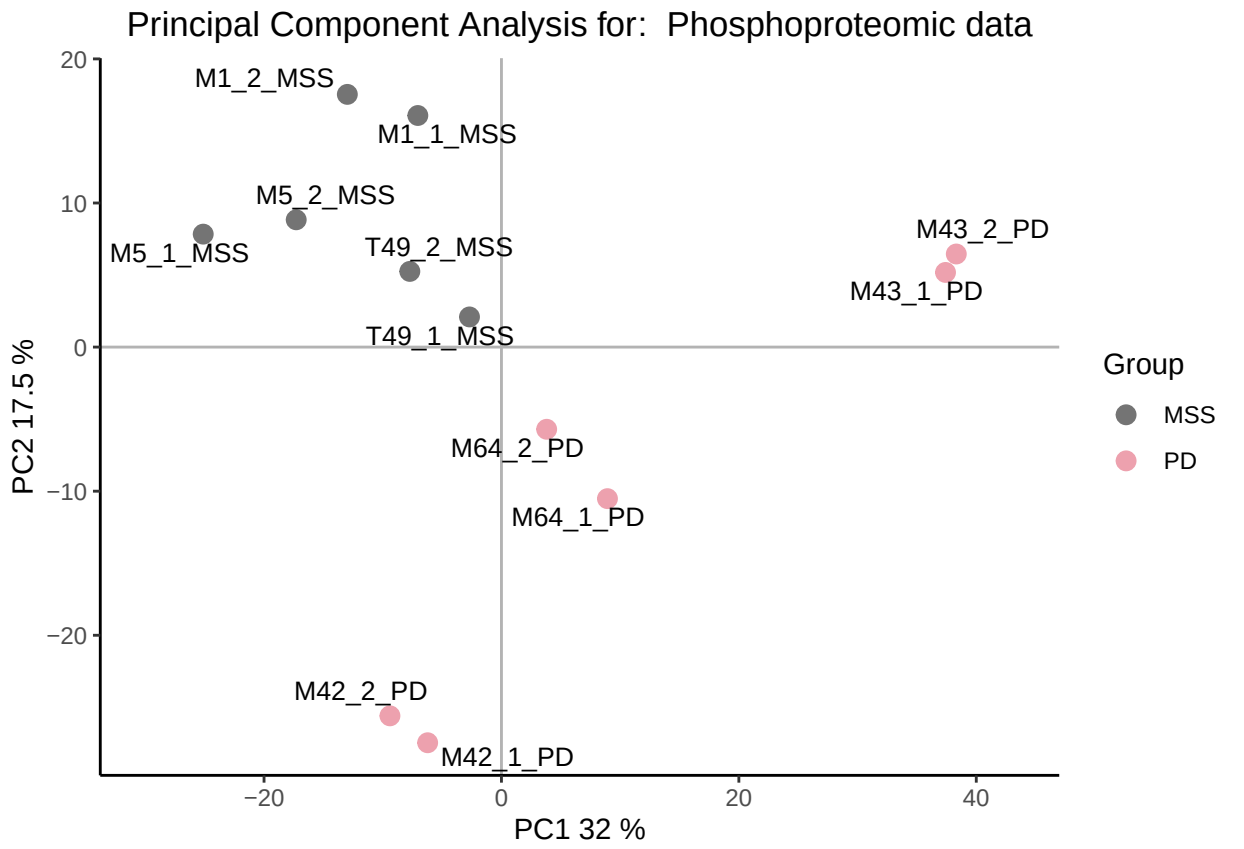
Análisis de Componentes Principales

A continuación, se procederá a aplicar sobre la matriz de abundancia el análisis de componentes principales. Esto permitirá reducir la dimensionalidad del dataset (se disponen de cientos de cientos de variables) y permitirá además identificar patrones o diferencias entre los grupos.

```
# Análisis de Componentes Principales
source("https://raw.githubusercontent.com/uebvhir/UEB_PCA/master/UEB_plotPCA3.R")
```

```
## Loading required package: ggrepel
```

```
plotPCA3(datos=as.matrix(log10(abundancias+1)), labels=colnames(abundancias),
         factor=targets$Phenotype, title ="Phosphoproteomic data",
         scale=FALSE, colores=1:2, size = 3.5, glineas = 2.5)
```



Este gráfico muestra el resultado de aplicar el análisis de componentes principales sobre los datos. En el eje X se representa la componente principal 1 y en el eje y la componente principal 2.

Se puede apreciar una clara separación entre los grupos MSS y PD, lo que indica que estos dos grupos presentan diferencias significativas, las cuales no se habían podido apreciar en los análisis anteriores. Además, también se puede ver que mientras las muestras el grupo MSS permanecen agrupadas, las muestras del grupo PD están muy dispersas, lo que indica heterogeneidad en el grupo.

5. Análisis Estadístico

Se usará el paquete *Lima* para llevar a cabo un análisis de expresión diferencial.

Para ello, en primer lugar, se crea la matriz de diseño, en la que a cada muestra se le aplica un 1 y un 0, dependiendo de si está presente o no en los grupos definidos.

```
# Convertir el archivo de targets a dataframe y definir grupos
targets <- as.data.frame(targets)
groups <- as.factor(targets$Phenotype)

# Crear el diseño del modelo
designMat <- model.matrix(~ -1 + groups)
print(designMat)
```

```
##    groupsMSS groupsPD
## 1          1        0
## 2          1        0
## 3          1        0
## 4          1        0
## 5          1        0
## 6          1        0
## 7          0        1
## 8          0        1
## 9          0        1
## 10         0        1
## 11         0        1
## 12         0        1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$groups
## [1] "contr.treatment"
```

Como se ha comentado en la introducción, hay una réplica de cada muestra, por ello, se utilizará la librería duplicateCorrelation con el objetivo de manejar adecuadamente estas réplicas.

```
# Calcular la correlación entre réplicas técnicas
dupcor <- duplicateCorrelation(abundancias, designMat, block=targets$Individual)
print(dupcor$consensus.correlation)
```

```
## [1] 0.8770936
```

```
# Crear la matriz de contraste para comparar PD y MSS
contMat <- makeContrasts(mainEff = groupsPD - groupsMSS, levels = designMat)
print(contMat)
```

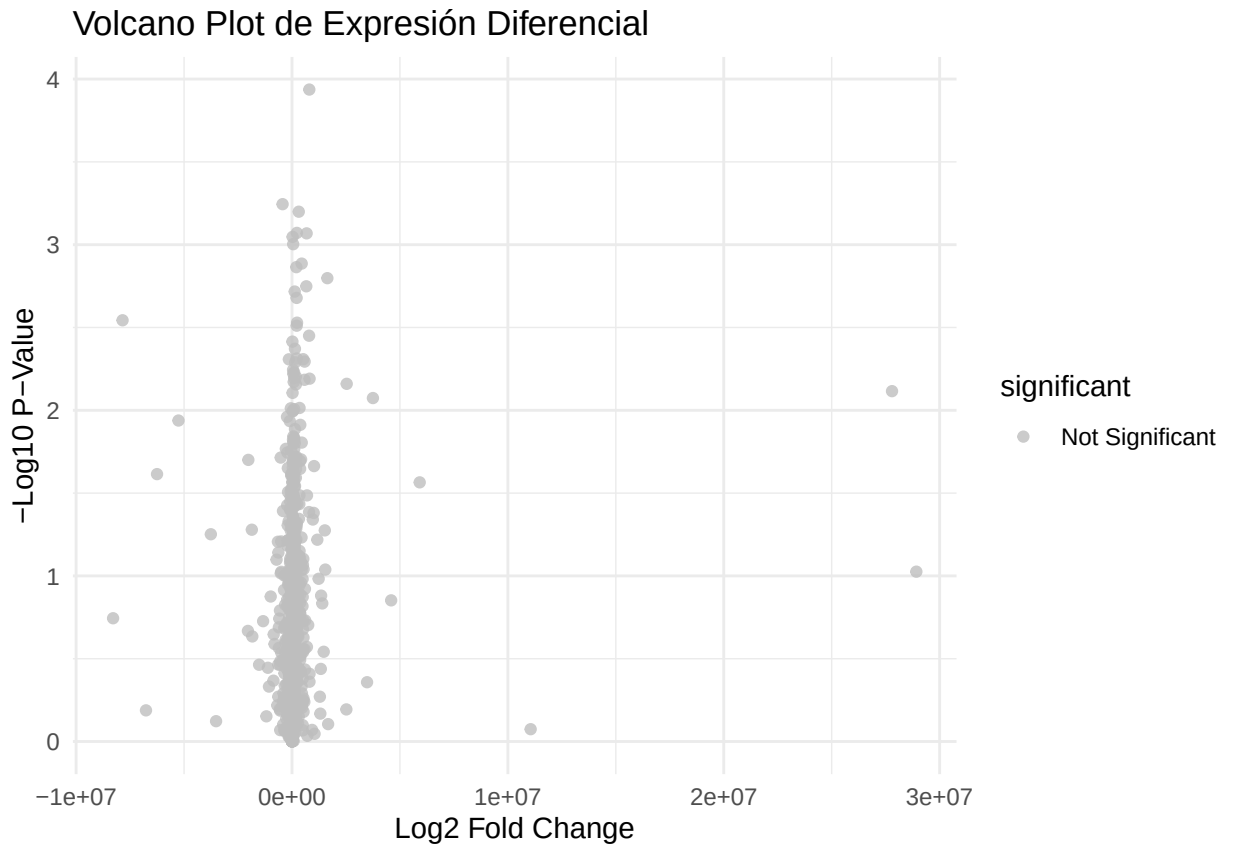
```
##            Contrasts
## Levels      mainEff
##   groupsMSS      -1
##   groupsPD        1
```

```
# Ajuste del modelo con lmFit usando la correlación de réplicas
fit <- lmFit(abundancias, designMat, block = targets$Individual, correlation = dupcor$consensus)
fit2 <- contrasts.fit(fit, contMat)
fit2 <- eBayes(fit2)
```

```
## Warning: Zero sample variances detected, have been offset away from zero
```

```
# Obtener los resultados en formato de dataframe
results_df <- data.frame(
  logFC = fit2$coef[, "mainEff"],
  P.Value = fit2$p.value[, "mainEff"],
  adj.P.Val = p.adjust(fit2$p.value[, "mainEff"], method = "BH")
)

# Añadir una columna de significancia para resaltar los puntos en el Volcano Plot
results_df <- results_df %>%
  mutate(significant = ifelse(adj.P.Val < 0.05 & abs(logFC) > 1, "Significant", "Not Significant"))

# Crear el Volcano Plot personalizado con ggplot2
ggplot(results_df, aes(x = logFC, y = -log10(P.Value), color = significant)) +
```

```
  geom_point(alpha = 0.8) +
  scale_color_manual(values = c("Significant" = "red", "Not Significant" = "gray")) +
  labs(title = "Volcano Plot de Expresión Diferencial",
       x = "Log2 Fold Change",
       y = "-Log10 P-Value") +
  theme_minimal()
```



Como se puede apreciar en el VolcanoPlot no se observan diferencias significativas entre grupos, pero esto puede deberse a a alta variabilidad, tamaño pequeño de muestra y a las similitudes que presentan ambos grupos. La variabilidad observada en el PCA puede no deberse a diferencias significativas entre los fosfopéptidos.

Exportación de Archivos

```
# Guardar el objeto SummarizedExperiment como .RDS
saveRDS(se, file = "summarize_data/summarized_experiment.rds")

# Guardar el objeto en formato .RData
save(se, file = "summarize_data/summarized_experiment.RData")
```

```
# Guardar los datos en forma de texto
# Exportar la matriz de datos de abundancia
write.csv(as.data.frame(assay(se, "counts")), "datos_texto/abundancias.csv", row.names = TRUE)

# Exportar los metadatos de las muestras
write.csv(as.data.frame(colData(se)), "datos_texto/sample_info.csv", row.names = TRUE)
```

```r
# Exportar los metadatos de los fosfopéptidos
write.csv(as.data.frame(rowData(se)), "datos_texto/row_info.csv", row.names = TRUE)
```

Repositorio de Github

A continuación se muestra los comandos utilizados para crear el repositorio Github:

- En primer lugar, se crea el repositorio en mi cuenta de Github y lo llamamos Manzano-Reche-Gabriel-PEC1

```r
# Iniciar Git en local
git init

# Añadir archivos presentes en el path
system("git add .")

# Realizar un commmit
system('git command -m "Mensaje"')

# Conectar repositorio local con el repositorio de GitHub
system("git remote add origin https://github.com/GabriManz/Manzano-Reche-Gabriel-PEC1")

# Subir los cambios al repositorio de GitHub
system("git push origin main")
```

El repositorio de Github se encuentra en el siguiente enlace:

- https://github.com/GabriManz/Manzano-Reche-Gabriel-PEC1

Este repositorio presenta la siguiente estructura:

```
Manzano-Reche-Gabriel-PEC1/
   data/
       abundancias.csv
       sample_info.csv
       row_info.csv
   summarized_data/
       summarized_experiment.rds
       summarized_experiment.RData
   README.md
   Manzano-Reche-Gabriel-PEC1.Rmd
   Manzano-Reche-Gabriel-PEC1.html
   Manzano-Reche-Gabriel-PEC1.pdf
   Manzano-Reche-Gabriel-PEC1.R
   .gitignore
```