

# PEC 1: Análisis de Datos Ómicos

Daniel Camacho Montaña

2024-11-01

## Contents

<b>1. Introducción</b>	<b>1</b>
<b>2. Lectura de los datos</b>	<b>2</b>
2.1 Preprocesado de los datos . . . . .	4
2.2 Guardado de datos. Creación del repositorio de GitHub. . . . .	5
<b>3. Análisis de los datos</b>	<b>6</b>
3.1 Estudio de la varianza entre réplicas . . . . .	6
3.2 Visualización de la distribución . . . . .	9
3.3 Análisis de los Componentes Principales (PCA) . . . . .	12
3.4 Análisis ANOVA de los fosfopéptidos . . . . .	18
<b>4. Conclusiones</b>	<b>21</b>

## 1. Introducción

El presente análisis se basa en un conjunto de datos de fosfoproteómica obtenido a partir de un experimento que investigó dos subtipos tumorales diferentes utilizando modelos de xenoinjertos derivados de pacientes (PDX). Las muestras fueron enriquecidas con fosfopéptidos y luego analizadas mediante espectrometría de masas acoplada a cromatografía líquida (LC-MS) en condiciones de duplicados técnicos, lo que permitió medir las abundancias normalizadas de señales MS de aproximadamente 1,400 fosfopéptidos.

El objetivo principal de este análisis es identificar fosfopéptidos que puedan diferenciar entre los dos grupos tumorales mediante el uso de métodos estadísticos y visualizaciones gráficas. Los datos proporcionados están organizados en un archivo de Excel, denominado “*TIO2+PTYR-human-MSS+MSIvsPD.XLSX*”.

Los dos grupos tumorales se definen como:

- **Grupo MSS:** Incluye las muestras M1, M5 y T49.
- **Grupo PD:** Incluye las muestras M42, M43 y M64.

Antes de comenzar a trabajar, será necesario crear un repositorio en Git para guardar todos los datos generados durante el análisis. Los repositorios de GitHub ofrecen múltiples beneficios para el desarrollo de software,

como el control de versiones, que permite realizar un seguimiento de los cambios y colaborar de manera eficiente. Además, su interfaz intuitiva y herramientas de gestión de proyectos mejoran la comunicación entre los equipos.

En GitHub, podemos crear directamente el repositorio, al que llamaremos “Camacho-Montan-o-Daniel-PEC1”. Este repositorio estará vinculado a nuestra carpeta local, de manera que, al actualizar los datos, podremos sincronizar los cambios con el repositorio de forma inmediata.

Para crear el repositorio, primero debemos inicializarlo (`git init`) y configurar nuestros datos en GitHub (nombre de usuario y correo electrónico)..

```
system("git init") # Inicializa un nuevo repositorio de Git
system("git config --global user.name 'Daniel Camacho'")
system("git config --global user.email 'dcamacmon@uoc.edu'")
```

En caso de no haber problemas, se generarán dichas configuraciones en el repositorio. A continuación, tendremos que vincular el repositorio local con el repositorio remoto. Con el siguiente comando primero asignaremos la URL del repositorio remoto a la variable `repo_url`, y es la que se utilizará para vincular el repositorio local con el repositorio remoto.

Además, con `system()` ejecutaremos una instrucción de git en el sistema operativo desde R. Paste creará una cadena que combinará el comando de `git remote add origin` con la URL del repositorio remoto, lo que permitirá enviar cambios al repositorio de GitHub.

```
repo_url <- "https://github.com/dcamacmon/Camacho-Monta-o-Daniel-PEC1"
system(paste("git remote add origin", repo_url))
```

```
## [1] 3
```

Con la respuesta afirmativa, podemos confirmar que el directorio remoto se ha vinculado correctamente al directorio local, y podremos comenzar a trabajar.

## 2. Lectura de los datos

Para iniciar con el estudio de los datos, primero debemos cargar la información del dataset de metabolómica desde github. El archivo, al tratarse de XLSM, usaremos el paquete `readxl` y cargaremos la información directamente desde el repositorio remoto.

Como estamos importando los datos desde un directorio web, primero crearemos un archivo temporal con la extensión `.xlsx` (llamado `temp_file`), luego descargaremos el archivo desde la URL (`download.file()`) y finalmente cargaremos los datos en nuestro data frame con la función `read_excel()`.

En este caso, como la dirección URL del repositorio original es muy extensa, tendremos que dividirla y posteriormente pegarla con `paste0()`.

```
library(readxl)

url <- paste0("https://github.com/nutrimetabolomics/metaboData/raw/main/",
             "Datasets/2018-Phosphoproteomics/TI02%2BPTYR-human-MSS%2BMSIvsPD.XLSX")

temp_file <- tempfile(fileext = ".xslm") #Creación del archivo temporal
download.file(url, temp_file, mode = "wb") #Descarga del archivo desde el repositorio
fosfo_data <- read_excel(temp_file) #Cargamos el documento
```

Con el archivo ya cargado, podemos ver una dsitribución generalizada de los datos con `summary()`.

```
summary(fosfo_data) #Resumen estadístico básico
```

```
## SequenceModifications Accession Description Score
## Length:1438 Length:1438 Length:1438 Min. : 19.51
## Class :character Class :character Class :character 1st Qu.: 38.96
## Mode :character Mode :character Mode :character Median : 47.48
## Mean : 51.30
## 3rd Qu.: 60.06
## Max. :132.19
## 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
## CLASS PHOSPHO
## Length:1438 Length:1438
## Class :character Class :character
## Mode :character Mode :character
##
##
##
```

En un análisis preeliminar, vemos que hay 18 columnas de datos:

- La primera columna contiene el tipo de modificaciones en la secuencia peptídica, así como la ubicación de la modificación (identificación de la modificación).
- La tercera columna describe la secuencia analizada.
- La cuarta columna presenta el Score, que evalúa la confiabilidad de la identificación de una secuencia peptídica.
- Las siguientes 12 columnas contienen los datos obtenidos mediante LC-MS para los grupos tumorales y las dos réplicas técnicas.
- Las dos últimas columnas contienen dos variables, CLASS y PHOSPHO, posiblemente de tipo categóricas

Dado que las dos últimas variables son de tipo categóricas, las convertiremos a tipo *factor* para facilitar posibles análisis futuros.

Verificaremos estas dos conversiones con `str()`, que ahora mostrará las dos variables con los niveles correspondientes.

```
fosfo_data$CLASS<-(as.factor(fosfo_data$CLASS))
str(fosfo_data$CLASS)
```

```
## Factor w/ 2 levels "C","H": 2 2 2 2 2 2 2 2 2 ...
```

```
fosfo_data$PHOSPHO<-(as.factor(fosfo_data$PHOSPHO))
str(fosfo_data$PHOSPHO)
```

```
## Factor w/ 2 levels "S/T","Y": 2 2 2 2 2 2 2 2 2 ...
```

Tal y como se sospechaba del análisis estadístico simple, las dos variables pueden ser leídas como categóricas de dos niveles cada una.

## 2.1 Preprocesado de los datos

Una vez cargados los datos, crearemos un objeto de clase `SummarizedExperiment` (una extensión de `ExpressionSet`), que permite el manejo y almacenamiento de datos de fosfoproteómica. Este formato es adecuado porque organiza los datos de abundancia junto con los metadatos relevantes de las filas (entradas). Gracias a este formato, podemos asociar metadatos a las filas y columnas de manera más clara, y también permite manejar datos con réplicas, lo cual es necesario para nuestro análisis.

Para crear el `SummarizedExperiment`, primero necesitamos cargar el paquete `BiocManager` para instalar y cargar el paquete `SummarizedExperiment`.

```
if (!requireNamespace("SummarizedExperiment", quietly = TRUE)) {
  BiocManager::install("SummarizedExperiment")
}
library(SummarizedExperiment)
```

Una vez tenemos el paquete cargado, debemos extraer los datos de “abundancia”, es decir, las lecturas de cada entrada, que se encuentran entre la columna 5 y la 16 (incluidas ambas).

Del mismo modo, creamos un dataframe para los metadatos, los cuales ya conocemos por el análisis preliminar (información sobre las filas y columnas). Recordemos que tenemos 3 grupos para cada clase tumoral y tenemos 2 réplicas para cada grupo.

```
abundances <- fosfo_data[, 5:16] #Contenemos los datos de abundancias
row_data <- fosfo_data[, c("SequenceModifications", "Accession",
                           "Description", "Score", "CLASS", "PHOSPHO")] #Contenemos los metadatos de la
col_metadata <- DataFrame( #Contenemos los metadatos de las columnas
  SampleID = colnames(abundances),
  Group = c(rep("MSS", 6), rep("PD", 6))
)
```

Como no disponemos de una columna en `row_data` con identificadores, crearemos una nueva columna en los metadatos de las filas denominada `peptide_id`.

```
row_data$peptide_id <- rownames(abundances)
```

Una vez hemos preprocesado los datos para que tengan el formato adecuado, podemos crear el Summarized-Experiment.

```
se <- SummarizedExperiment(  
  assays = list(counts = as.matrix(abundances)), # Datos de abundancia como matriz  
  rowData = row_data, # Metadatos para las filas  
  colData = col_metadata # Metadatos para las columnas  
)
```

Como resumen, los metadatos de las filas (rowData) describirán las características de las entidades medidas, como secuencias, modificaciones o puntuaciones de identificación. Los metadatos de las columnas (colData) contienen información sobre las muestras o grupos tumorales. Los datos de ensayo (assay) son las mediciones experimentales (obtenidos por LC-SM) que se almacenan en una matriz para análisis posteriores.

## 2.2 Guardado de datos. Creación del repositorio de GitHub.

Ahora que tenemos el SummarizedExperiment (SE), lo exportaremos en formato .RDA para incluirlo en nuestro repositorio de GitHub. Además, extraeremos los datos de expresión (assay data), los metadatos de las muestras (colData) y los metadatos de las variables (rowData) en formato de texto.

Como ya disponemos de toda la información de SE, simplemente debemos usar la función save() y especificar el nombre del archivo que crearemos.

```
save(se, file = "Fosfodatos_se.Rda") #Guardamos el SE en formato .Rda
```

Para los datos de ensayo y metadatos, primero debemos seleccionar que datos queremos del SE para cada archivo. Para los datos de assay, los guardaremos en counts, para posteriormente crear el archivo basándonos en esta información. Además, especificaremos la separación por tabuladores (sep = "\t"), el nombre de las filas (row.names = TRUE) y columnas (col.names = TRUE). Es importante especificar que los datos de texto no se rodeen de comillas, por lo que especificaremos que quote=FALSE.

```
counts<-assay(se,"counts") #Obtenemos los datos de assay data  
write.table(counts, file = "datos_assay.txt", sep = "\t",  
            quote = FALSE, row.names = TRUE, col.names = TRUE) #Obtenemos el .txt
```

Seguiremos el mismo procedimiento para los metadatos.

```
sample_metadata <- colData(se) #Obtenemos los metadatos de las muestras  
write.table(sample_metadata, file = "metadatos_muestras.txt", sep = "\t",  
            quote = FALSE, row.names = TRUE, col.names = TRUE) #Se guardan en un .txt  
  
gene_metadata <- rowData(se) #Obtenemos los metadatos de las variables  
write.table(gene_metadata, file = "metadatos_genes.txt", sep = "\t",  
            quote = FALSE, row.names = TRUE, col.names = TRUE) #Se guardan en un .txt
```

A veces es conveniente mantener los metadatos por separado. Por ello, crearemos un archivo .md que contenga los metadatos del dataset. Esta información se introducirá en markdown\_content, que posteriormente se almacenará en el archivo metadatos\_dataset.md.

```
writeLines(markdown_content, "metadatos_dataset.md")
```

Toda esta información, junto con el objeto SE, los metadatos y los demás archivos generados, deben almacenarse en un repositorio de GitHub. Con el repositorio local previamente creado, podemos cargar los archivos guardados. Primero, agregamos los archivos al directorio Git (`git add`). Si los archivos ya están guardados, realizamos un commit para “cargar” las modificaciones generadas (`git commit`) y, finalmente, un push para subir los cambios al repositorio remoto (`git push origin main`).

```
# Agregar todos los archivos
system("git add .")
```

```
## [1] 0
```

```
# Hacer commit de los cambios
system("git commit -m 'Añadir_archivos'")
```

```
## [1] 0
```

```
# Hacer push para subir los cambios a GitHub
system("git push origin main") # Asegúrate de que la rama sea correcta
```

```
## [1] 0
```

El mensaje de retorno `[1] 0` indica que los comandos se ejecutaron correctamente, lo que significa que todos los cambios en el directorio de trabajo han sido añadidos al área de preparación (staging area) de Git.

## 3. Análisis de los datos

### 3.1 Estudio de la varianza entre réplicas

Con el objeto `SummarizedExperiment` (SE) podemos visualizar los datos para comprender su distribución y variabilidad. Existen dos grupos principales (los tipos tumorales), y dentro de cada grupo se clasifican tres muestras.

Dado que cada muestra tiene dos réplicas, analizaremos si la varianza entre ambas es significativa. Esto es importante, ya que una diferencia en la varianza podría deberse a errores técnicos o a la recolección de datos. Para este análisis, utilizaremos el paquete `car`, comenzando con un estudio en la muestra M1 (réplicas M1\_1 y M1\_2).

```
library(car)
abundancias_M1 <- abundances[, c("M1_1_MSS", "M1_2_MSS")]
```

Crearemos un `data.frame` con los datos de abundancia de M1, incorporando ambas réplicas y añadiendo un factor de dos niveles para diferenciarlas. A continuación, visualizaremos las primeras seis observaciones (`head()`) y la estructura de los datos (`str()`).

```
# Crear un dataframe con los nombres de las columnas adecuados
datos_M1 <- data.frame(
  Abundancia = c(abundancias_M1[["M1_1_MSS"]], abundancias_M1[["M1_2_MSS"]]),
  Replica = factor(rep(c("Replica 1", "Replica 2"), each = nrow(abundancias_M1)))
)
head(datos_M1)
```

```
##      Abundancia  Replica
## 1      24.29438 Replica 1
## 2       0.00000 Replica 1
## 3     3412.60332 Replica 1
## 4  220431.17880 Replica 1
## 5   18254.77813 Replica 1
## 6  644513.31840 Replica 1
```

```
str(datos_M1)
```

```
## 'data.frame': 2876 obs. of 2 variables:
## $ Abundancia: num 24.3 0 3412.6 220431.2 18254.8 ...
## $ Replica : Factor w/ 2 levels "Replica 1","Replica 2": 1 1 1 1 1 1 1 1 1 1 ...
```

Realizamos el test de Levene para verificar la homogeneidad de varianzas entre ambas réplicas.

```
resultados_levene <- leveneTest(Abundancia ~ Replica, data = datos_M1)
print(resultados_levene)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##      Df F value Pr(>F)
## group 1 0.2676 0.605
##      2874
```

Vemos que el test de homogeneidad de varianzas acepta la hipótesis nula, por lo que podemos asumir que no hay una variabilidad significativa entre ambas lecturas.

Para obtener el resultado de test de Levene para todas las réplicas, primero crearemos un nuevo dataframe vacío que contendrá el resultado del test, tanto el p-valor como el identificador de la muestra.

No obstante, para realizar una comparación a tal escala, primero deberíamos escalar los datos, de manera que sus varianzas sean comparables. Para ello, escalaremos utilizando el método de Z-score, por lo que los datos serán transformados, restando la media de cada variable y dividiendo por su desviación estándar.

```
abundance_data <- assay(se, "counts")
abundance_data <- scale(abundance_data)
summary(abundance_data)
```

```
##      M1_1_MSS      M1_2_MSS      M5_1_MSS      M5_2_MSS
## Min.      :-0.2483 Min.      :-0.16981 Min.      :-0.2378 Min.      :-0.2327
## 1st Qu.   :-0.2422 1st Qu.   :-0.16612 1st Qu.   :-0.2352 1st Qu.   :-0.2298
## Median   :-0.2152 Median   :-0.15171 Median   :-0.2166 Median   :-0.2093
## Mean      : 0.0000 Mean      : 0.00000 Mean      : 0.0000 Mean      : 0.0000
## 3rd Qu.   :-0.1215 3rd Qu.   :-0.09401 3rd Qu.   :-0.1215 3rd Qu.   :-0.1167
## Max.      :17.8157 Max.      :29.29723 Max.      :15.2140 Max.      :17.2675
##      T49_1_MSS      T49_2_MSS      M42_1_PD      M42_2_PD
## Min.      :-0.2048 Min.      :-0.2221 Min.      :-0.2063 Min.      :-0.2047
## 1st Qu.   :-0.2013 1st Qu.   :-0.2180 1st Qu.   :-0.2035 1st Qu.   :-0.2021
## Median   :-0.1838 Median   :-0.2000 Median   :-0.1868 Median   :-0.1860
## Mean      : 0.0000 Mean      : 0.0000 Mean      : 0.0000 Mean      : 0.0000
## 3rd Qu.   :-0.1206 3rd Qu.   :-0.1313 3rd Qu.   :-0.1106 3rd Qu.   :-0.1114
## Max.      :18.3765 Max.      :13.8173 Max.      :25.3861 Max.      :25.9133
```

##	M43_1_PD	M43_2_PD	M64_1_PD	M64_2_PD
##	Min. : -0.2210	Min. : -0.16948	Min. : -0.16909	Min. : -0.1458
##	1st Qu.: -0.2086	1st Qu.: -0.16131	1st Qu.: -0.16512	1st Qu.: -0.1432
##	Median : -0.1780	Median : -0.14133	Median : -0.15032	Median : -0.1315
##	Mean : 0.0000	Mean : 0.00000	Mean : 0.00000	Mean : 0.0000
##	3rd Qu.: -0.0909	3rd Qu.: -0.07411	3rd Qu.: -0.09368	3rd Qu.: -0.0838
##	Max. : 21.9741	Max. : 29.62607	Max. : 25.60790	Max. : 26.5920

Con los datos transformados, procederemos a crear el dataframe para contener los resultados del test de Levene para todas las muestras. Como conocemos los nombres de dichas muestras, los contendremos en un nuevo objeto.

```
# Obtenemos los nombres de las columnas para las réplicas
muestras <- c("M1", "M5", "T49", "M42", "M43", "M64")

resultados_levene <- data.frame(Muestra = character(),
                                p_valor = numeric(),
                                stringsAsFactors = FALSE)
```

Luego iteramos a través de cada muestra, donde los sufijos “\_MSS” o “\_PD” indican el grupo, y “\_1” o “\_2” indican la réplica. Ejecutamos un bucle que verifica la existencia de columnas para las réplicas MSS o PD, evalúa la varianza entre las dos réplicas con el test de Levene y almacena los resultados en el data.frame.

```
resultados_levene <- data.frame(Muestra = character(),
                                p_valor = numeric(),
                                stringsAsFactors = FALSE) # Inicializamos el dataframe para almacenar

# Iteramos sobre las muestras
for (muestra in muestras) {
  # Definimos los nombres de las columnas de las réplicas de MSS y PD
  col1 <- paste0(muestra, "_1_MSS")
  col2 <- paste0(muestra, "_2_MSS")
  col3 <- paste0(muestra, "_1_PD")
  col4 <- paste0(muestra, "_2_PD")

  # Inicializamos la variable abundancias
  abundancias <- NULL

  # Verificamos si ambas columnas MSS existen, guardamos los datos
  if (col1 %in% colnames(abundance_data) & col2 %in% colnames(abundance_data)) {
    abundancias <- abundance_data[, c(col1, col2)]
  }
  # Verificamos si ambas columnas PD existen, guardamos los datos
  else if (col3 %in% colnames(abundance_data) & col4 %in% colnames(abundance_data)) {
    abundancias <- abundance_data[, c(col3, col4)]
  }

  # Si se encontraron abundancias, se realiza el test de Levene
  if (!is.null(abundancias)) {
    datos <- data.frame(
      Abundancia = c(abundancias[, 1], abundancias[, 2]),
      Replica = factor(rep(c("Replica 1", "Replica 2"), each = nrow(abundancias)))
    )
```



```

resultado_levene <- leveneTest(Abundancia ~ Replica, data = datos)

# Guardar el p-valor en la tabla de resultados
resultados_levene <- rbind(resultados_levene,
                           data.frame(Muestra = muestra,
                                       p_valor = resultado_levene$`Pr(>F)`[1]))
}
}

```

Finalmente, mostramos el resultado de los test de Levene.

```
print(resultados_levene)
```

```

##  Muestra    p_valor
## 1      M1 0.04475678
## 2      M5 0.87330361
## 3     T49 0.65103490
## 4     M42 0.97244938
## 5     M43 0.20803981
## 6     M64 0.55992030

```

Vemos entonces, que hay diferencias significativas en la variabilidad entre algunas de las réplicas de las muestras de los dos grupos. Una vez escalados, vemos que hay variabilidad en la primera muestra, mientras que en el resto no presenta dicha variabilidad. Esto podría indicar que hay una variabilidad técnica, debido a que el equipo de LC-MS es muy sensible, y cualquier cambio en las condiciones puede afectar la abundancia medida.

Esto implica, que pese a que podemos promediar las réplicas de la mayoría de muestras, decidiremos mantenerlas independientes para los futuros análisis, ya que en los estudios de LC-MS la variabilidad puede mostrar diferencias biológicas subyacentes en las muestras.

### 3.2 Visualización de la distribución

A continuación, mostraremos la distribución de las muestras de la matriz de abundancia. Al tratar con 12 columnas de datos al mismo tiempo, numéricamente es difícil determinar algún patrón en ellos. Para evaluarlo más gráficamente, realizaremos un boxplot con estos mismos datos.

Primero recuperaremos la matriz de abundancias (`abundance_data`) del SE, y la convertiremos en una matriz de formato largo (usando la función `melt()` del paquete `reshape2`) y modificamos los títulos para facilitar su lectura. Esta conversión es importante para poder trabajar con futuro paquetes como `ggplot2` y `dplyr`, ya que mejora la legibilidad y simplifica el procesamiento de datos.

```
library(reshape2)
```

```

abundance_long <- melt(abundance_data)
colnames(abundance_long) <- c("Fosfo péptido", "Muestra", "Abundancia")

```

Después, dividiremos los dos grupos tumorales (MSS y PD) y se identificarán correspondientemente.

```

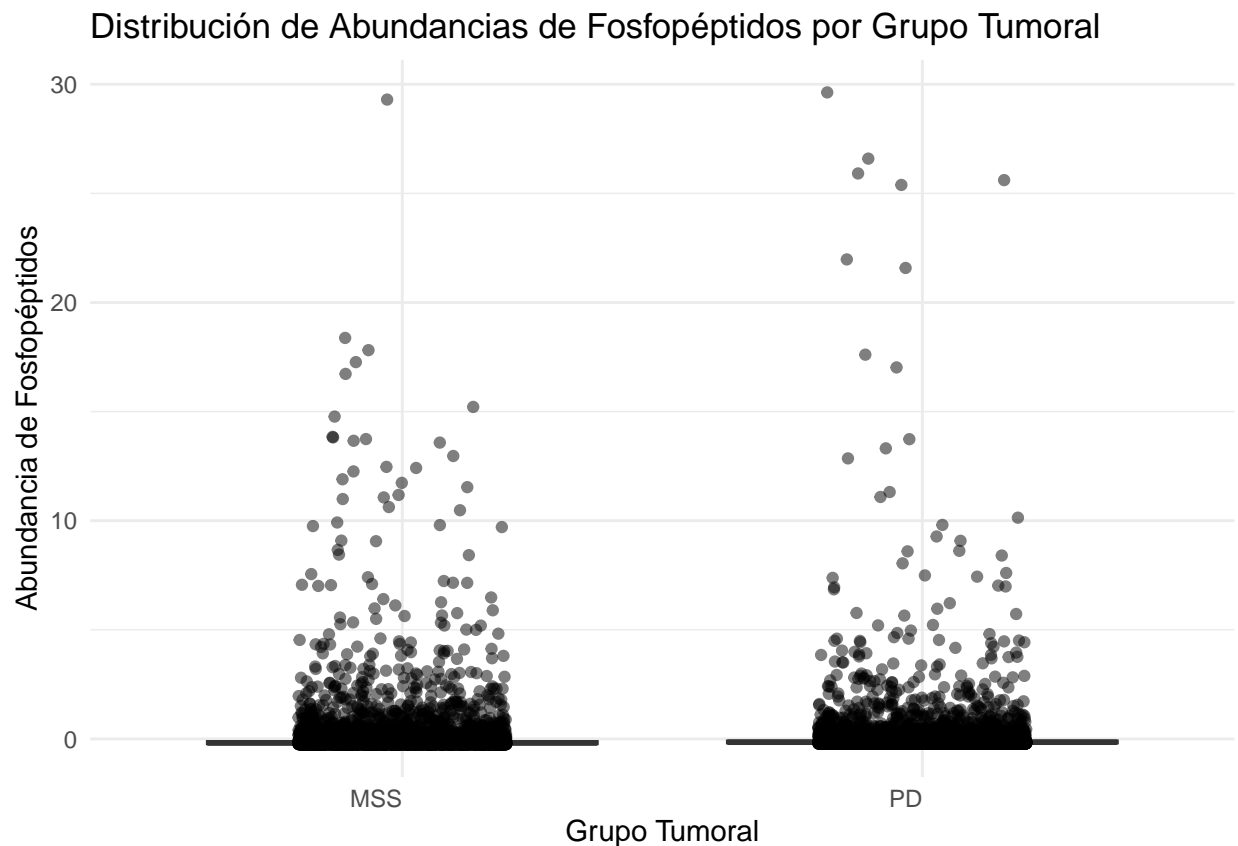
grupos <- as.data.frame(colData(se)$Group)
colnames(grupos) <- "Group"
abundance_long$Group <- rep(grupos$Group, each = nrow(abundance_data))

```

Finalmente, crearemos un gráfico ggplot con los dos grupos, por lo que necesitaremos el paquete `ggplot2`. Generaremos un gráfico boxplor con dispersión de puntos para visualizar la dsitribución de las abundancias de fosfopéptidos por grupo tumoral en el dataframe transformado (en formato largo). Para evitar que los outliers saturen el gráfico, los eliminaremos del gráfico.

```
library(ggplot2)
```

```
ggplot(abundance_long, aes(x = Group, y = as.numeric(Abundancia), fill = Group)) +
  geom_boxplot(outlier.shape = NA) + # Oculta los outliers si hay muchos puntos
  geom_jitter(width = 0.2, alpha = 0.5, color = "black") + # Añade puntos de dispersión
  labs(title = "Distribución de Abundancias de Fosfopéptidos por Grupo Tumoral",
       x = "Grupo Tumoral",
       y = "Abundancia de Fosfopéptidos") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, hjust = 1), # Rotar etiquetas en el eje x
        legend.position = "none")
```



Vemos que, de entre los dos grupos tumorales, parece ser que el grupo PD tiene una abundancia superior (de hecho, es bastante similar, pero parece que el PD tiene ciertos valores mucho más altos). Debido a que hemos especificado cierto grado de transparencia, el hecho de que la base del gráfico se vea tan oscura implica una alta densidad de datos en esa zona.

Para poder ver los datos estadísticos descriptivos de ambos grupos, usaremos el paquete `dplyr`, creando un dataframe con la media, la mediana, la desviación estándar y el ratio intercuartílico.

```
library(dplyr)
```

```
abundance_stats <- abundance_long %>%  
  group_by(Group) %>%  
  summarise(  
    Media = mean(Abundancia, na.rm = TRUE),  
    Mediana = median(Abundancia, na.rm = TRUE),  
    SD = sd(Abundancia, na.rm = TRUE),  
    IQR = IQR(Abundancia, na.rm = TRUE)  
  )  
print(abundance_stats)
```

```
## # A tibble: 2 x 5  
##   Group      Media Mediana      SD      IQR  
##   <chr>    <dbl>   <dbl> <dbl>   <dbl>  
## 1 MSS      2.17e-18 -0.184  1.00  0.106  
## 2 PD       5.73e-18 -0.147  1.00  0.0928
```

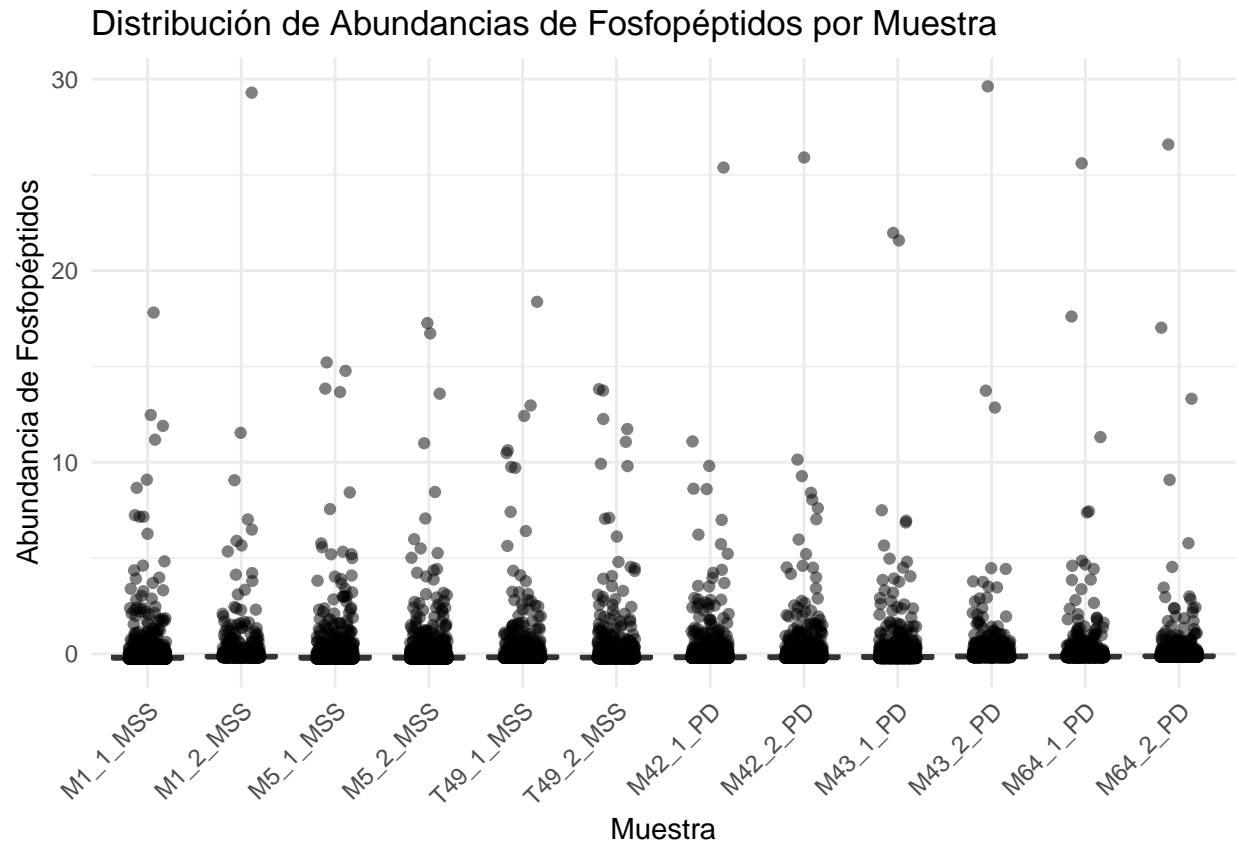
Como ya veíamos en el gráfico original de los grupos tumorales, la media y la mediana de PD es superior a la del grupo MSS, pero con unos valores muy cercanos a 0 en ambos casos. Esto es normal debido a que el Z-Score transforma los datos para que tengan una media de 0 y una desviación estándar de 1 (como vemos en la tabla).

La mediana y SD son similares en ambos grupos, por lo que podemos afirmar que después de la transformación no hay una gran diferencia en la variabilidad de las abundancias entre los dos grupos. El IQR (la diferencia entre el Q1 y el Q3) también es muy similar entre los dos grupos, lo que indica que la variabilidad dentro de los dos grupos es comparable.

Además, el IQR superior indica que la mitad central de los datos es más dispersa (hay más diferencia entre el Q1 y el Q3).

Como ya tenemos los datos de la muestra en `abundance_long`, solo debemos crear el boxplot en base a esos datos, con las muestras (MSS y PD) en base a su abundancia. Además, como tenemos 12 muestras, inclinamos las etiquetas del eje X para facilitar su lectura.

```
ggplot(abundance_long, aes(x = Muestra, y = Abundancia)) +  
  geom_boxplot(aes(fill = Group), outlier.shape = NA) + # Oculta los outliers  
  geom_jitter(width = 0.2, alpha = 0.5, color = "black") + # Añade puntos  
  labs(title = "Distribución de Abundancias de Fosfopéptidos por Muestra",  
        x = "Muestra",  
        y = "Abundancia de Fosfopéptidos") +  
  theme_minimal() +  
  theme(axis.text.x = element_text(angle = 45, hjust = 1), # Rotar etiquetas en el eje x  
        legend.position = "none") # Eliminar la leyenda
```



Como sabemos, las muestras M1, M5 y T49 son del grupo MSS, mientras que las muestras M42, M43 y M64 son del grupo PD.

En el gráfico, vemos que las muestras de MSS tienen valores inferiores. En cambio, en PD, vemos que todas las muestras parecen tener valores superiores a MSS, pero no muchos, lo que podría explicar la media ligeramente superior de PD.

### 3.3 Análisis de los Componentes Principales (PCA)

Para profundizar con el análisis de los datos, realizaremos un análisis de los componentes principales (PCA), ya que nos pueden proporcionar datos relevantes, así como permitirnos identificar outliers que desvían el patrón general de los datos. Un paso previo a realizar un análisis de PCA es normalizar los datos. En este caso, obviaremos este proceso, ya que como se explica en la introducción, estas lecturas ya han sido normalizadas previamente.

Primero crearemos un nuevo objeto con los datos de las abundancias con varianza diferente a 0 (ya que los valores con varianza 0 son constantes, no aportan información relevante al análisis). Para poder aplicar el mismo procedimiento a todas las filas, emplearemos la función `apply`. Seguidamente, filtraremos aquellos datos que tengan una varianza inferior a 0.01

```
# Extraemos los datos de abundancia y calculamos la varianza de todos los datos de abundancia
colMeans(abundance_data) # Debe devolver valores cercanos a 0
```

```
##      M1_1_MSS      M1_2_MSS      M5_1_MSS      M5_2_MSS      T49_1_MSS
## 1.271487e-17  6.961814e-18  1.394203e-17 -5.229504e-18 -2.072153e-17
##      T49_2_MSS      M42_1_PD      M42_2_PD      M43_1_PD      M43_2_PD
```

```
## 5.445439e-18 1.758850e-18 6.677117e-19 9.952343e-18 1.405934e-17
##      M64_1_PD      M64_2_PD
## 8.302666e-18 1.288376e-18
```

```
variances<-apply(abundance_data, 2, sd)
abundance_data <- abundance_data[, variances > 0.01]
```

A continuación, realizaremos el PCA y crearemos el data frame con los resultados. Además, cabe destacar la necesidad de transponer los datos de los fosfopéptidos para que se encuentren en las columnas (hasta ahora se han encontrado en las filas). El PCA se centrará y escalará los datos, lo que es recomendable para asegurar que todas las variables contribuyan de manera equitativa al análisis. También añadiremos la información de los grupos (MSS o PD), lo que facilitará la lectura de los datos en función de los grupos.

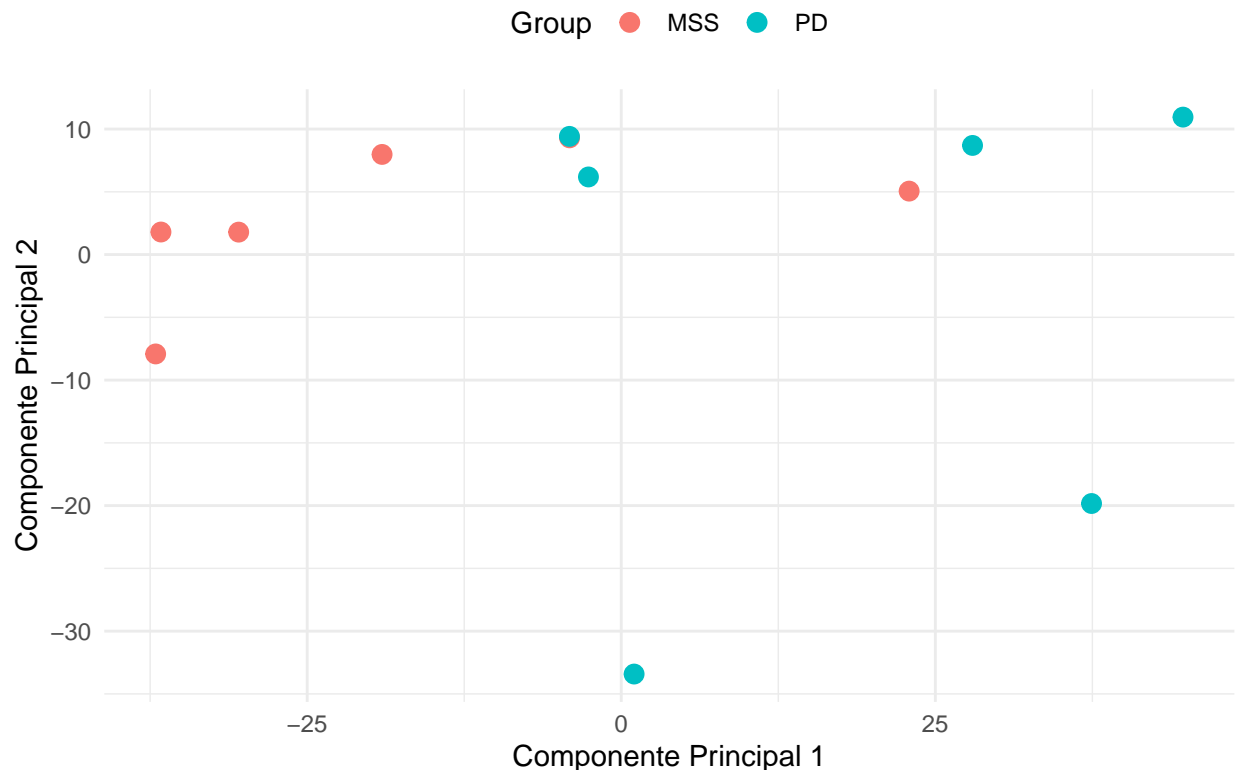
```
pca_result <- prcomp(t(abundance_data), center = TRUE, scale. = TRUE)

pca_data <- as.data.frame(pca_result$x)
# Agregamos información de grupo
pca_data$Group <- colData(se)$Group
```

Finalmente, para analizar visualmente la distribución, realizaremos un gráfico con el paquete ggplot2.

```
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group)) +
  geom_point(size = 3) +
  labs(title = "Análisis de Componentes Principales 1 y 2",
       x = "Componente Principal 1",
       y = "Componente Principal 2") +
  theme_minimal() +
  theme(legend.position = "top")
```

## Análisis de Componentes Principales 1 y 2



Vemos que, en el caso del grupo MSS, se ve mucho más representada por la componente principal 2 (se ven más concentradas arriba), mientras que el grupo PD se ve mucho más representado por la componente principal 1 (se ven más concentradas a la derecha). Aun así, debemos tener en cuenta que cada componente principal representa cierta cantidad de la variabilidad de los datos.

Para determinar que grado de variabilidad se representa con cada PCA, reuniremos los datos en un nuevo dataframe. Como ya calculamos los CP anteriormente, obtendremos primero las varianzas de cada componente (`variances`), así como la proporción de varianza explicada.

```
# Ver la proporción de varianza explicada
variances <- pca_result$sdev^2 # Varianzas de cada componente
explained_variance <- variances / sum(variances)*100 # Proporción de varianza explicada

# Crear un dataframe para visualización
pca_summary <- data.frame(
  Component = paste0("PC", 1:length(explained_variance)),
  Variance = explained_variance
)
```

En el data frame creado, tenemos una columna con el nombre de las PC y con su varianza, pero no tenemos el formato correcto, así que modificaremos el resultado para que se vea en base a 100 y no en notación científica. Además, no tenemos la varianza acumulada (únicamente la varianza de cada CP independientemente), por lo que crearemos la columna `CumulativeVariance`. Recordemos que, las CP se ordenan automáticamente por orden descendente en función de la variabilidad explicada, por lo que las primeras CP representarán gran parte de la variabilidad.

Los datos de CP siempre aparecen en notación científica (pese a que hemos especificado anteriormente que

queremos un porcentaje). Para una mejor lectura de los datos, modificaremos el formato, redondeando a 4 decimales y creamos la varianza acumulada.

```
#Modificamos el formato de los datos de Variance, así como la nueva columna acumulada.
pca_summary$Variance<-format(pca_summary$Variance, nsmall = 4, scientific = FALSE)
pca_summary$Variance <- round(as.numeric(pca_summary$Variance), 4) #Redondeamos a 4 decimales
pca_summary$CumulativeVariance <- cumsum(pca_summary$Variance) #Creamos la varianza acumulada

# Mostrar la proporción de varianza explicada
print(pca_summary)
```

##	Component	Variance	CumulativeVariance
## 1	PC1	55.3523	55.3523
## 2	PC2	13.1338	68.4861
## 3	PC3	11.1844	79.6705
## 4	PC4	8.0892	87.7597
## 5	PC5	6.2449	94.0046
## 6	PC6	2.0335	96.0381
## 7	PC7	1.2689	97.3070
## 8	PC8	0.9991	98.3061
## 9	PC9	0.9420	99.2481
## 10	PC10	0.4833	99.7314
## 11	PC11	0.2686	100.0000
## 12	PC12	0.0000	100.0000

Vemos, entonces, que las dos componentes principales contienen el 68.4861% de la variabilidad total. Esta variabilidad es importante, y usar únicamente 2 CP podría simplificar el análisis. Aún así, usaremos 3 Componentes Principales, que contendrán el 79.6705% de la variabilidad total.

De todos modos, aún conociendo la variabilidad asociada a los CP, desconocemos que fosfopéptidos influyen más en cada CP. En `rotation` de PCA, podemos ver que carga supone sobre las CP (nos centraremos en las 3 primeras, pero por separado). Primero, tendremos que cargar el paquete `tibble`.

```
library(tibble)
```

Después, obtendremos los resultados en `cargas` y crearemos un data frame con los datos de las CP1, CP2 y CP3 al que llamaremos `cargas_df`.

```
cargas <- pca_result$rotation
cargas_df <- as.data.frame(cargas[, c("PC1", "PC2", "PC3")])
cargas_df <- cargas_df %>%
  rownames_to_column(var = "Fosfopeptido")
```

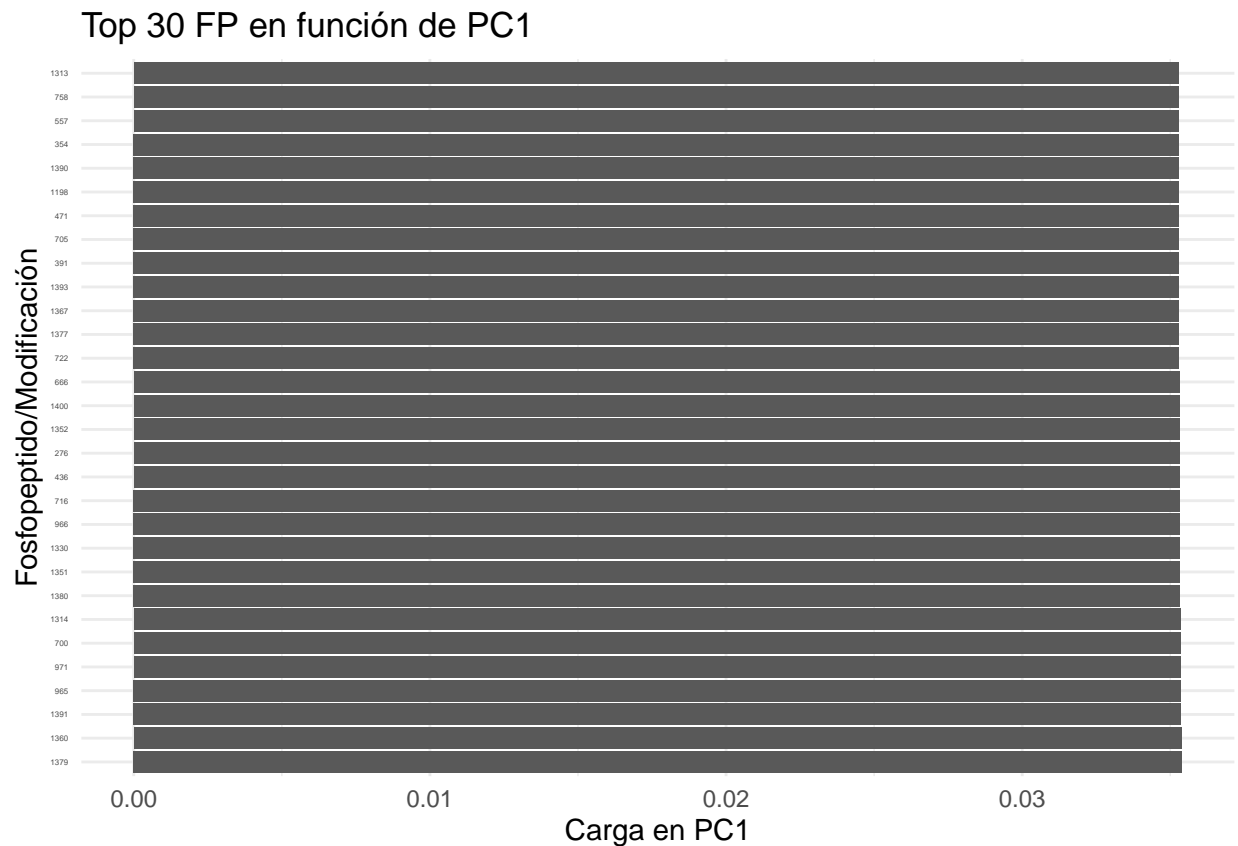
Posteriormente, seguiremos un proceso en que extraeremos los 30 valores con cargas más alta, reordenando de forma descendiente los fosfopéptidos en función de cada CP (en cada CP habrá diferentes fosfopéptidos que afecten con mayor intensidad).

Realizaremos el mismo proceso para las 3 componentes principales: Reordenamos de manera descendiente todos los fosfopéptidos en función de la CP, obtenemos los 30 valores más altos y los guardamos en el data frame. En el dataframe, determinamos que la columna con los fosfopéptidos sean de tipo factor, y estableceremos que los niveles del factor a partir de los valores de carga de los fosfopéptidos.

```
top_cargas_pc1 <- cargas_df %>%
  arrange(desc(PC1)) %>%
  head(30)
top_cargas_pc1$Fosfopeptido <- factor(top_cargas_pc1$Fosfopeptido,
                                       levels = top_cargas_pc1$Fosfopeptido)
```

Finalmente crearemos un gráfico de barras en que veamos como afectan cada uno de esos 30 resultados a la componente principal 1. Tendremos, en orden descendiente, los 30 fosfopéptidos que más influyen en esta CP.

```
ggplot(top_cargas_pc1, aes(x = Fosfopeptido, y = PC1)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top 30 FP en función de PC1",
       x = "Fosfopeptido/Modificación", y = "Carga en PC1") +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 3, angle = 0, hjust = 1)) # Tamaño y ángulo
```



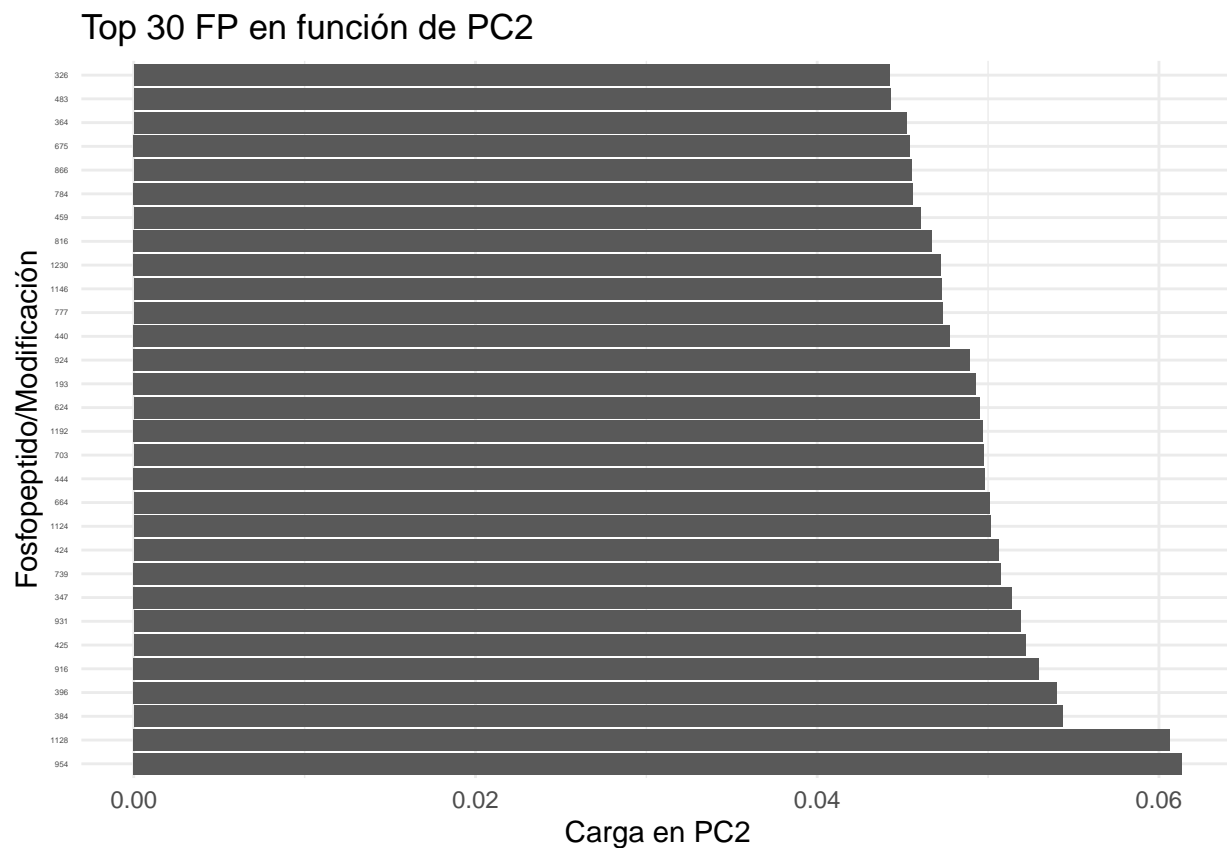
Realizaremos el mismo proceso para CP2:

```
#Reordenamos los datos en función de CP2
top_cargas_pc2 <- cargas_df %>%
  arrange(desc(PC2)) %>%
  head(30)
#Establecemos los niveles y el tipo factor
```



```
top_cargas_pc2$Fosfopeptido <- factor(top_cargas_pc2$Fosfopeptido,
                                       levels = top_cargas_pc2$Fosfopeptido)

#Creamos del gráfico
ggplot(top_cargas_pc2, aes(x = Fosfopeptido, y = PC2)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Top 30 FP en función de PC2",
       x = "Fosfopeptido/Modificación", y = "Carga en PC2") +
  theme_minimal() + # Tamaño y ángulo
  theme(axis.text.y = element_text(size = 3, angle = 0, hjust = 1))
```



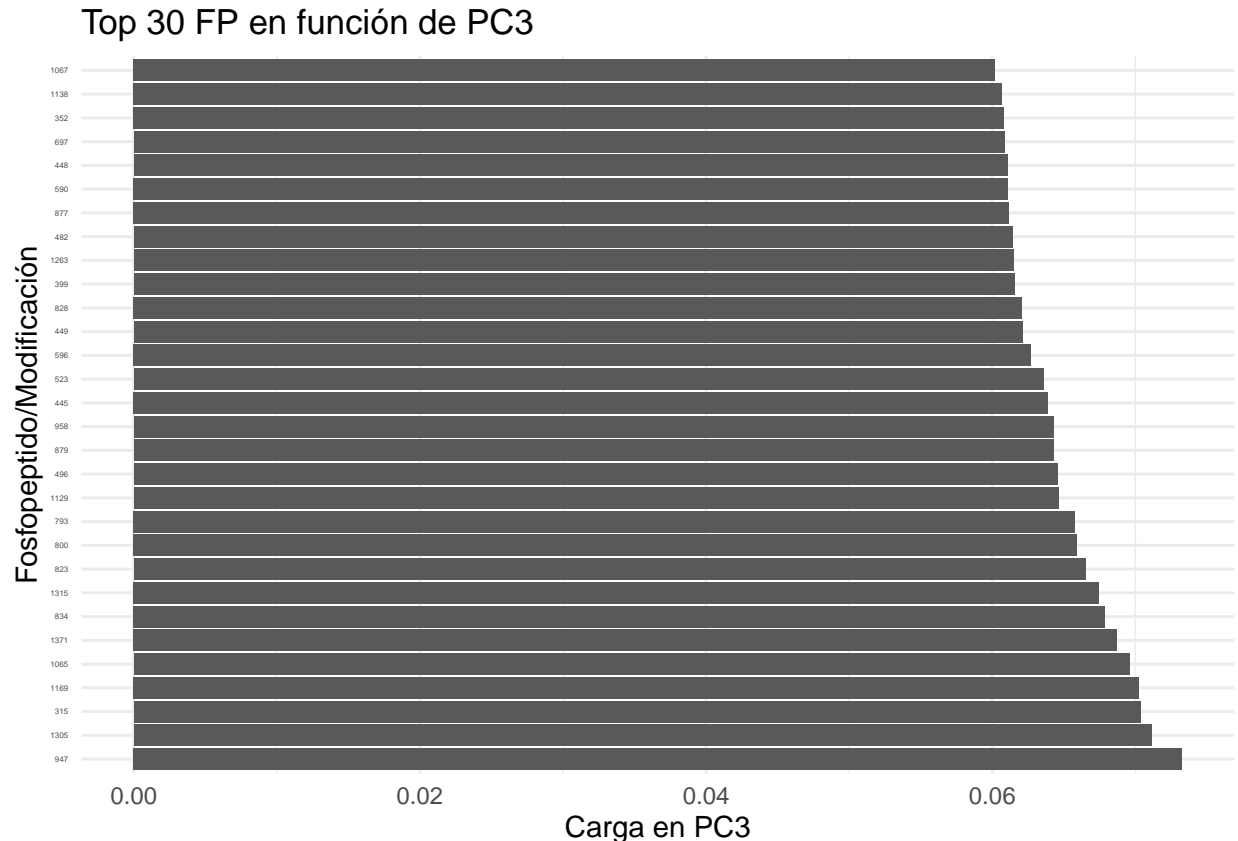
Repetiremos el proceso de reordenado y graficamos para PC3:

```
#Reordenamos los datos en función de CP3
top_cargas_pc3 <- cargas_df %>%
  arrange(desc(PC3)) %>%
  head(30)

#Establecemos los niveles y el tipo factor
top_cargas_pc3$Fosfopeptido <- factor(top_cargas_pc3$Fosfopeptido,
                                       levels = top_cargas_pc3$Fosfopeptido)

#Creación del gráfico
ggplot(top_cargas_pc3, aes(x = Fosfopeptido, y = PC3)) +
  geom_bar(stat = "identity") +
  coord_flip() +
```

```
labs(title = "Top 30 FP en función de PC3",
     x = "Fosfopeptido/Modificación", y = "Carga en PC3") +
theme_minimal() + # Tamaño y ángulo de las etiquetas
theme(axis.text.y = element_text(size = 3, angle = 0, hjust = 1))
```



Como podemos ver, cada fosfopéptido tiene una influencia diferente en cada CP, aunque en la CP1 los 50 primeros parece que todos tienen una carga similar. Tal y como se predecía, los fosfopéptidos no tiene la misma carga para todos los CP, por lo que visualmente no podemos determinar que FP es realmente significativo para diferenciar entre los dos tipos tumorales.

### 3.4 Análisis ANOVA de los fosfopéptidos

Para continuar con el análisis, realizaremos un ANOVA de un factor para ver si hay diferencias significativas en las abundancias de fosfopéptidos entre los grupos. Como los datos están estructurados con las variables (fosfopéptidos) en las filas y los grupos (MSS o PD) en las columnas, tendremos que transponer los datos de las abundancias. Para poder determinar a que grupo tumorales pertenecen las muestras, necesitaremos los metadatos contenidos en colData.

```
grupos <- colData(se)$Group

#Para tener las lecturas en horizontal y los fosfopéptidos en vertical
abundancias_df <- as.data.frame(t(abundance_data))
abundancias_df$grupo <- grupos
```

Además, debemos tener en cuenta que no podemos aplicar un ANOVA a cada fosfopéptido independientemente, por lo que usaremos `apply` para repetir el proceso en todas las columnas (ahora la variable fosfopéptido) y guardaremos el resultado (el p-valor) en `anova_results`.

```
# Función para aplicar ANOVA en cada fosfopéptido
anova_results <- apply(abundancias_df[, -ncol(abundancias_df)], 2, function(y) {
  aov_result <- aov(y ~ abundancias_df$grupo)
  summary(aov_result)[[1]]$Pr(>F)[1] # Extrae el valor p
})
```

Finalmente, crearemos un data frame con los resultados, con los resultados del test ANOVA para los fosfopéptidos. Además, realizaremos un ajuste de los p-valor obtenidos con el test ANOVA mediante el enfoque de Benjamini-Hochberg, ya que, de esta manera, controlamos la cantidad de falsos positivos.

```
anova_results <- data.frame(Fosfopeptido = names(anova_results),
                           p_value = anova_results)

anova_results$p_adjusted <- p.adjust(anova_results$p_value, method = "fdr")
```

A continuación, para poder filtrar los fosfopéptidos, primero tendremos que seleccionar los índices que tienen un p-valor ajustado inferior a 0.05 (el valor de significancia). Con los índices ya guardados, podremos buscar aquellos péptidos contenidos en el SE que correspondan a estos índices. Para poder leer estos datos correctamente, también cargaremos los metadatos asociados a estos péptidos.

```
#Seleccionamos los índices que tienen un p-valor inferior a 0.05
signif_peptides <- which(anova_results$p_adjusted < 0.05)

#Obtenemos los datos y metadatos a partir de los índices seleccionados
signif_abundances <- assay(se, "counts")[signif_peptides, ]
signif_row_data <- rowData(se)[signif_peptides, ]

#Mostramos los 6 primeros resultados
print(head(signif_row_data$SequenceModifications))
```

```
## [1] "PYQYPALTPEQK[4] Phospho"      "AYTNFDAER[2] Phospho"
## [3] "LSLEGDHSTPPSAYGSVK[14] Phospho" "FAGDKGYLTK[7] Phospho"
## [5] "HKAPGSADYGFAPAAGR[9] Phospho" "NSHTDNVSYEHSFNK[9] Phospho"
```

Después del test ANOVA vemos que, de los 1438 fosfopéptidos analizados, solo 104 muestran diferencias significativas (teniendo en cuenta el p\_valor ajustado) entre los grupos MSS y PD. Estos fosfopéptidos podrían ser útiles para distinguir los dos subtipos tumorales, lo que sugiere que existen patrones de fosforilación asociados a las diferencias moleculares entre los grupos.

Para poder determinar patrones diferenciales, podemos estudiar los datos proporcionados por los fosfopéptidos que han resultado significativos. Por lo tanto, primero tendremos que crear un nuevo objeto con solo estos datos. Como conocemos los índices, crearemos un dataframe llamado `signif_abundances`, el cual escalaremos usando el método de Z-score.

```
# Abundancias de fosfopéptidos significativos
signif_abundances <- assay(se, "counts")[signif_peptides, ]
save(se, file = "Fosfodatos_se_significativos.Rda") #Guardamos el SE en formato .Rda
signif_abundances <- (scale(signif_abundances))
```

Como posteriormente requeriremos del paquete de ggplot para visualizar el análisis y su distribución, convertiremos los datos a formato largo usando melt(). Para finalizar el preprocesado de datos, añadiremos la información del grupo.

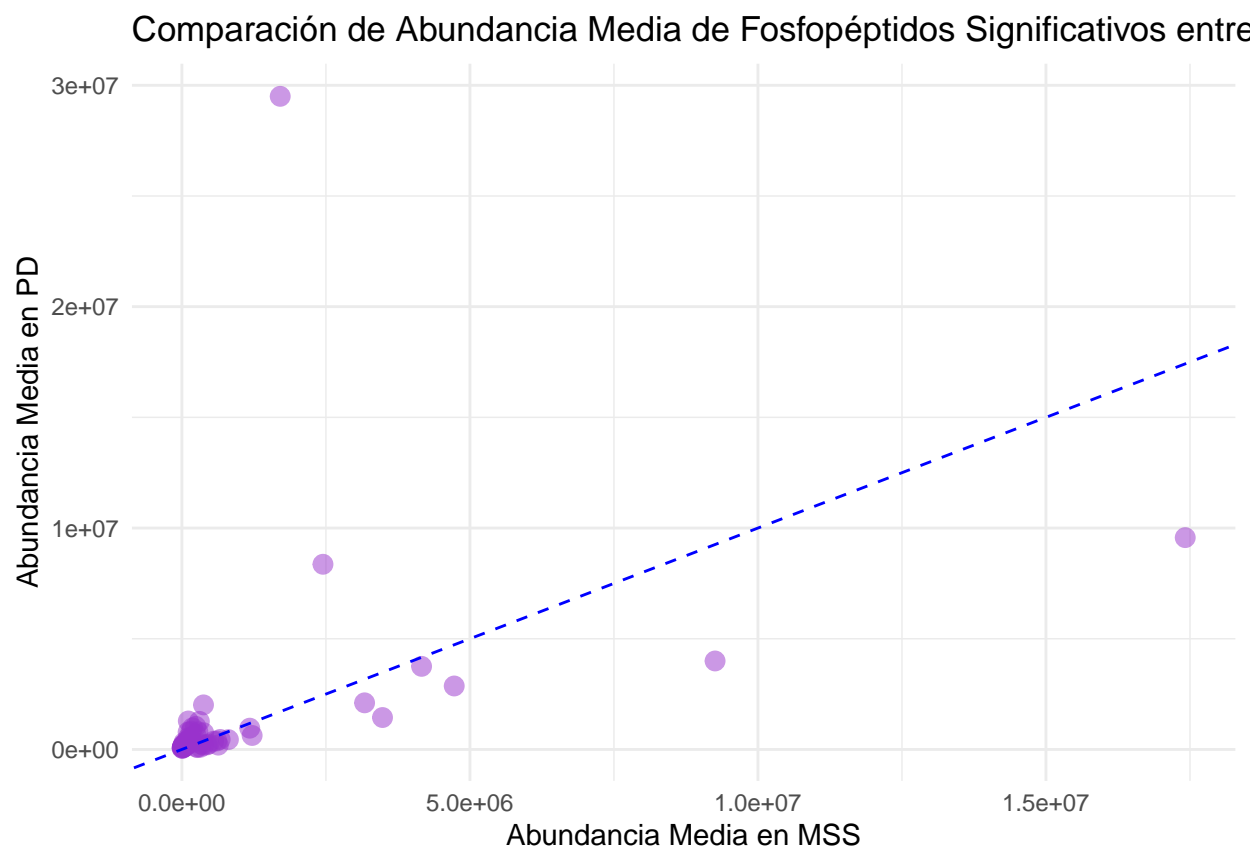
```
signif_abundances_long <- melt(signif_abundances)
colnames(signif_abundances_long) <- c("peptide_id", "Sample", "Abundance")
signif_abundances_long$Group <- colData(se)$Group[signif_abundances_long$Sample]
```

Una vez tenemos los datos filtrados y preprocesados, crearemos un data frame que contenga los metadatos de ambos grupos y calcule las medias de abundancia para cada fosfopéptido en los grupos tumorales.

```
group_means <- data.frame(
  MSS_mean = rowMeans(signif_abundances[, colData(se)$Group == "MSS"]),
  PD_mean = rowMeans(signif_abundances[, colData(se)$Group == "PD"])
)
```

Visualizaremos los datos que hemos obtenido, en función de PD y MSS y veremos si, en alguno de los péptidos filtrados, se visualizan patrones claros de diferenciación. Para poder determinar un umbral que “divida” las medias de ambos grupos, añadiremos una línea (geom\_abline) que represente la media en relación 1:1 (slope=1) y que pase por las coordenadas de origen.

```
ggplot(group_means, aes(x = MSS_mean, y = PD_mean)) +
  geom_point(color = "darkorchid", size = 3, alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "blue") + # Línea de identidad
  theme_minimal() +
  labs(
    title = "Comparación de Abundancia Media de Fosfopéptidos Significativos entre MSS y PD",
    x = "Abundancia Media en MSS",
    y = "Abundancia Media en PD"
  )
```



Gracias al ggplot podemos ver que, como mínimo, tenemos 2 fosfopéptido que se ve más representado por uno de los grupos tumorales, mientras que la mayoría parecen no reflejar ningún patrón claro respecto a la media de abundancia entre la media de cada grupo.

Estas diferencias son bastante claras entre los grupos MSS y PD, por lo que podemos afirmar que (sabiendo que las diferencias son significativas gracias al test ANOVA), tienen diferencias claras entre los grupos MSS y PD, y podría implicar que están asociados con una característica distintiva de cada grupo tumoral.

## 4. Conclusiones

Con base en el análisis que hemos realizado y la visualización de las abundancias para fosfopéptidos entre los grupos tumorales MSS y PD, podemos extraer las siguientes conclusiones.

El ANOVA de un factor ha identificado 104 fosfopéptidos significativos entre los dos grupos tumorales, lo que sugiere que hay una diferencia en los patrones de fosforilación entre estos dos grupos. Estos fosfopéptidos podrían estar asociados con características moleculares claves que distinguen los dos subtipos tumorales.

El análisis visual de las abundancias medias de los fosfopéptidos resultantes en el gráfico de dispersión sugiere que existen algunos fosfopéptidos que tienen abundancias significativamente mayores en uno de los grupos. Los puntos alejados de la línea de identidad (que representa igual abundancia entre los dos grupos) indican que estos fosfopéptidos tienen diferencias marcadas en su abundancia entre los subtipos tumorales.

Los 104 fosfopéptidos significativos podrían ser biomarcadores potenciales que diferencian los subtipos tumorales MSS y PD, y podrían estar involucrados en vías de señalización específicas. Si estos biomarcadores son validados en más estudios, podrían tener aplicaciones en diagnóstico, pronóstico o tratamiento para pacientes con estos tipos de tumores.

Además, la posibilidad de estudiar la fosforilación en sitios específicos (en Y o S/T) podría revelar patrones aún más claros de diferenciación entre los grupos MSS y PD. Los metadatos sobre las modificaciones en los residuos de fosforilación proporcionan información adicional que podría permitir identificar patrones específicos más claros.

Finalmente, para asegurar la validez de los resultados, se podrían realizar análisis de validación cruzada o validar los hallazgos utilizando un conjunto de datos independiente. Esto permitiría confirmar que los fosfopéptidos identificados son consistentes y reproducibles en otros conjuntos de datos o experimentos.

Toda la documentación se ha recopilado en el siguiente repositorio de github:

*<https://github.com/dcamacmon/Camacho-Monta-o-Daniel-PEC1>*