JAVIER MARTÍNEZ DELGADO

PEC1-ÓMICAS

1.-Seleccionar un dataset de metabolómica

El dataset seleccionado es metaboData/Datasets/2018-Phosphoproteomics at main · nutrimetabolomics/metaboData · GitHub.

```
# Creamos un archivo temporal en el que guardar los datos y los
url = "https://github.com/nutrimetabolomics/metaboData/raw/main
temporal = tempfile(fileext = ".xlsx")
download.file(url, destfile = temporal, mode = "wb")
data = read_excel(temporal)

# Comprobamos que los datos se han cargado bien
head(data)
```

Lo que he hecho, es descargar los datos de github directamente a un archivo temporal por lo que no estoy cargando un archivo que esté en mi ordenador y el código se puede utilizar desde cualquier dispositivo sin que de error. Una vez guardado en el archivo temporal, lo cargamos y lo visualizamos para comprobar que todo está bien

2.-Crear un contenedor SummarizedExperiment que contenga los datos y metadatos

Lo primero que tenemos que hacer es crear la matriz con los valores de expresión por lo que extraeremos esas columnas determinadas del dataframe y guardaremos también el nombre de las columnas, las cuales procesaremos para obtener metadatos

```
# Creamos la matriz con los valores de expresión
matriz_expresion = as.matrix(data[,5:16])

columnas_expresion = colnames(matriz_expresion)

# Sacamos como metadatos si es MSS o PD
vector_vacio = character(length(columnas_expresion))
for (i in seq_along(columnas_expresion)) {
  nombre_columna = columnas_expresion[i]
  parte = strsplit(nombre_columna, "_")[[1]]
  condicion = parte[length(parte)]
  vector_vacio[i] = condicion
}

# Juntamos el nombre de las columnas con los datos de si es MSS o PD para crear los metadatos de la muestra
metadatos_muestra = data.frame(
  id_muestra = columnas_expresion,
  condicion = vector_vacio
)
```

Con el resto de columnas, crearemos más metadatos, pero de las variables en este caso en lugar de metadatos de las muestras

```
# Seleccionamos las demás variables que no son de la muestra para crear los metadatos de las variables
metadatos_variables = data[,c("SequenceModifications","Accession","Description","Score","CLASS","PHOSPHO")]
```

Finalmente crearemos el contenedor SummarizedExperiment

```r
# Creamos el objeto SummarizedExperiment
se = SummarizedExperiment(
  assays = list(counts = matriz_expresion),
  rowData = metadatos_variables,
  colData = metadatos_muestra
)
```

```
> se
class: SummarizedExperiment
dim: 1438 12
metadata(0):
assays(1): counts
rownames: NULL
rowData names(6): SequenceModifications Accession ... CLASS PHOSPHO
colnames(12): M1_1_MSS M1_2_MSS ... M64_1_PD M64_2_PD
colData names(2): id_muestra condicion
>
```

Las dimensiones del contenedor son 1438x12 al igual que el dataframe que habíamos cargado. En assays podemos comprobar que tenemos la matriz de expresión con los valores de expresión para cada muestra. Los valores de rowdata proporcionan información adicional sobre cada muestra y coldata nos dice que tenemos dos columnas de metadatos para las muestras: el id y si es MSS o PD.

3.-Exploración del dataset

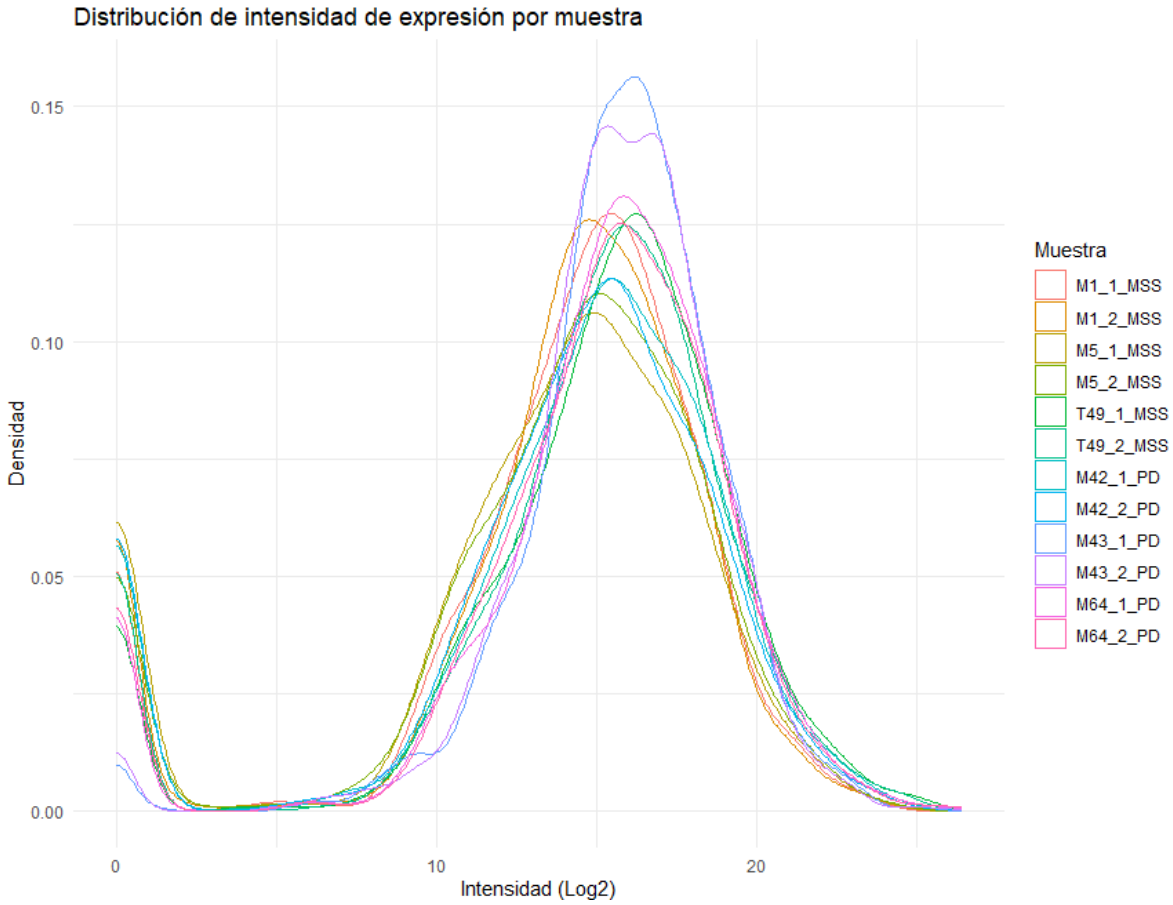
Lo primero que se suele hacer en un análisis exploratorio cuando se tienen variables cuantitativas es un resumen estadístico de las variables

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

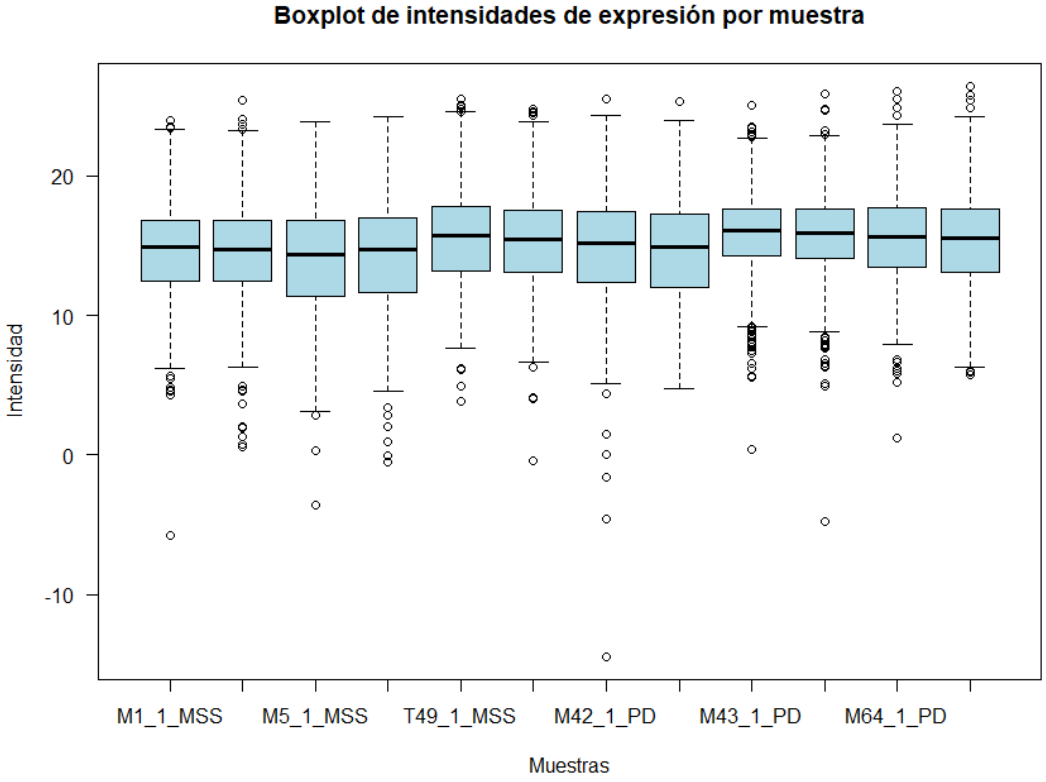
Llama mucho la atención el rango tan grande de las variables ya que el mínimo es 0 y el máximo es de decenas de millones.

A continuación, queremos ver la distribución de estos datos por lo que haremos una gráfica, pero antes, tendremos que aplicar el logaritmo a los datos ya que, al hacer la gráfica sin el logaritmo, como los datos están muy concentrados cerca del cero, no se puede apreciar nada

```r
# Histogramas
library(ggplot2)
log_expr_data = log2(expr_data + 1)
log_expr_data_long = reshape2::melt(log_expr_data, variable.name = "Muestra", value.name = "Intensidad_Log")

ggplot(log_expr_data_long, aes(x = Intensidad_Log, color = Muestra)) +
  geom_density() +
  labs(title = "Distribución de intensidad de expresión por muestra",
       x = "Intensidad (Log2)", y = "Densidad") +
  theme_minimal()
```

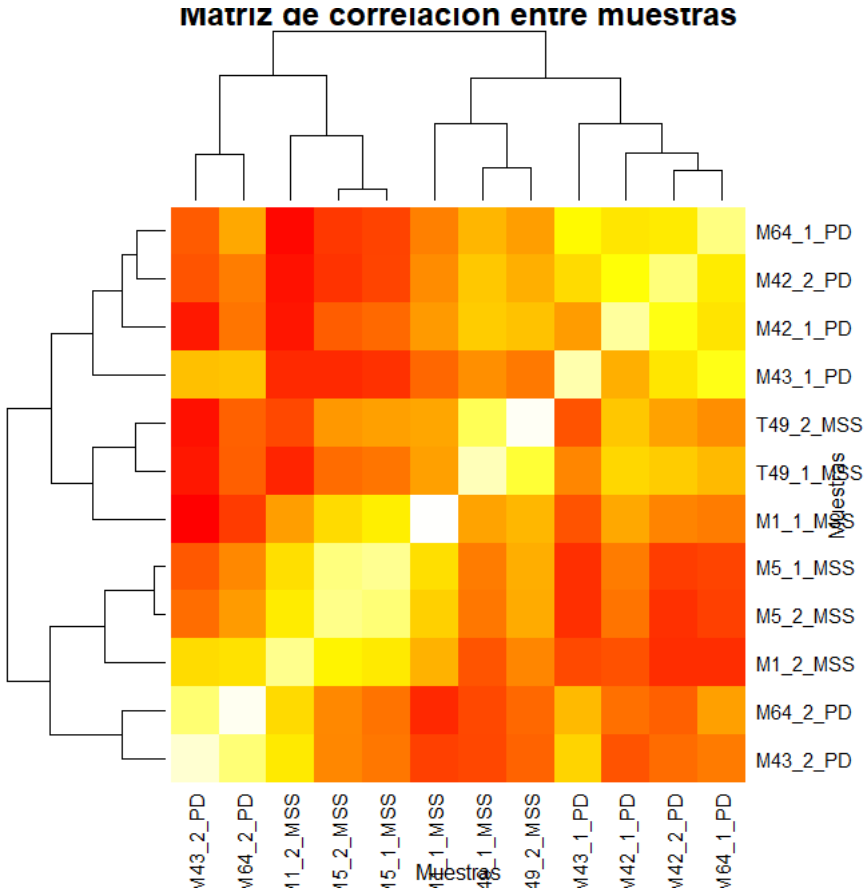


Podemos ver que las muestras se diferencian más entre sí a una intensidad muy baja cercana al cero o a una intensidad determinada formando un pico



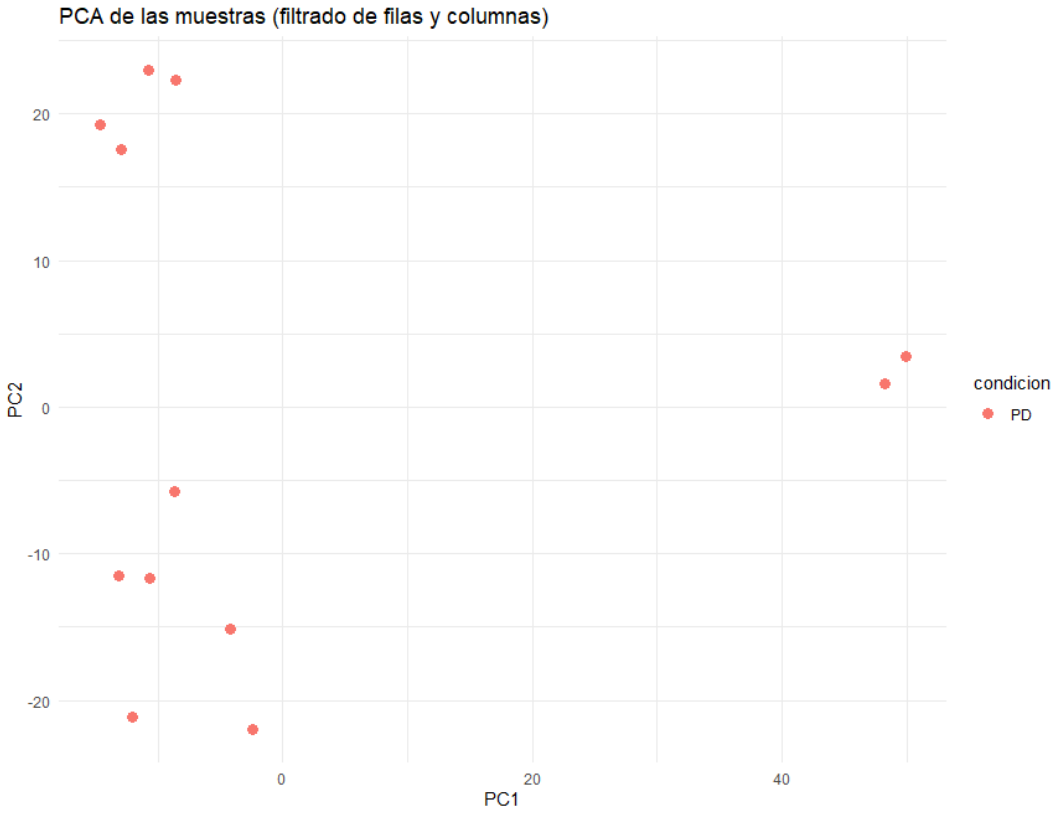
Al hacer un boxplot para ver cada variable y los outliers que tienen (también se ha tenido que usar los datos con logaritmo). Podemos ver que no hay ninguna que destaque sobre otras muestras pero sí que hay algunas con un rango mayor en los outliers

Si hacemos una matriz de correlación entre las variables podemos ver las que están fuertemente relacionadas



Para hacer un PCA, se ha tenido que hacer un threshold de intensidad para eliminar las de menor intensidad ya que el código devuelve un error al hacer el PCA

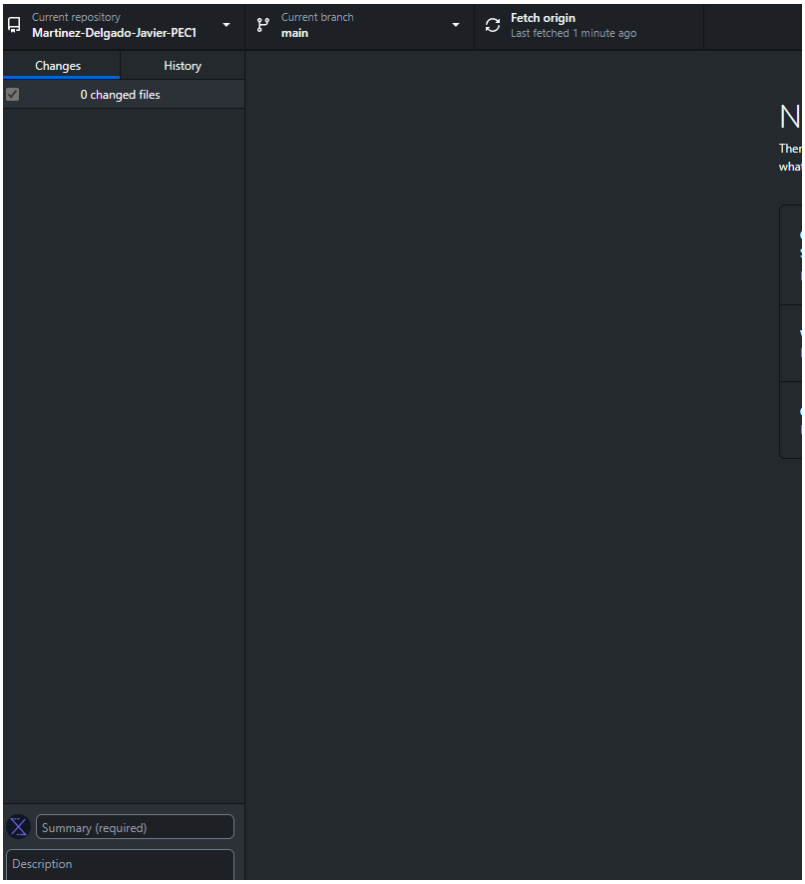
```
# Definimos la tolerancia para detectar una varianza baja
tolerancia = 1e-8
expr_matrix_filtered_rows = expr_matrix[apply(expr_matrix, 1, var) > tolerancia, ]
expr_matrix_filtered = expr_matrix_filtered_rows[, apply(expr_matrix_filtered_rows, 2, var) > tolerancia]

# PCA
pca = prcomp(t(expr_matrix_filtered), scale. = TRUE)
pca_df = as.data.frame(pca$x)
pca_df$condition = colData(se)$condition[colnames(expr_matrix_filtered)]

ggplot(pca_df, aes(x = PC1, y = PC2, color = condicion)) +
  geom_point(size = 3) +
  labs(title = "PCA de las muestras (filtrado de filas y columnas)", x = "PC1", y = "PC2") +
  theme_minimal()
```

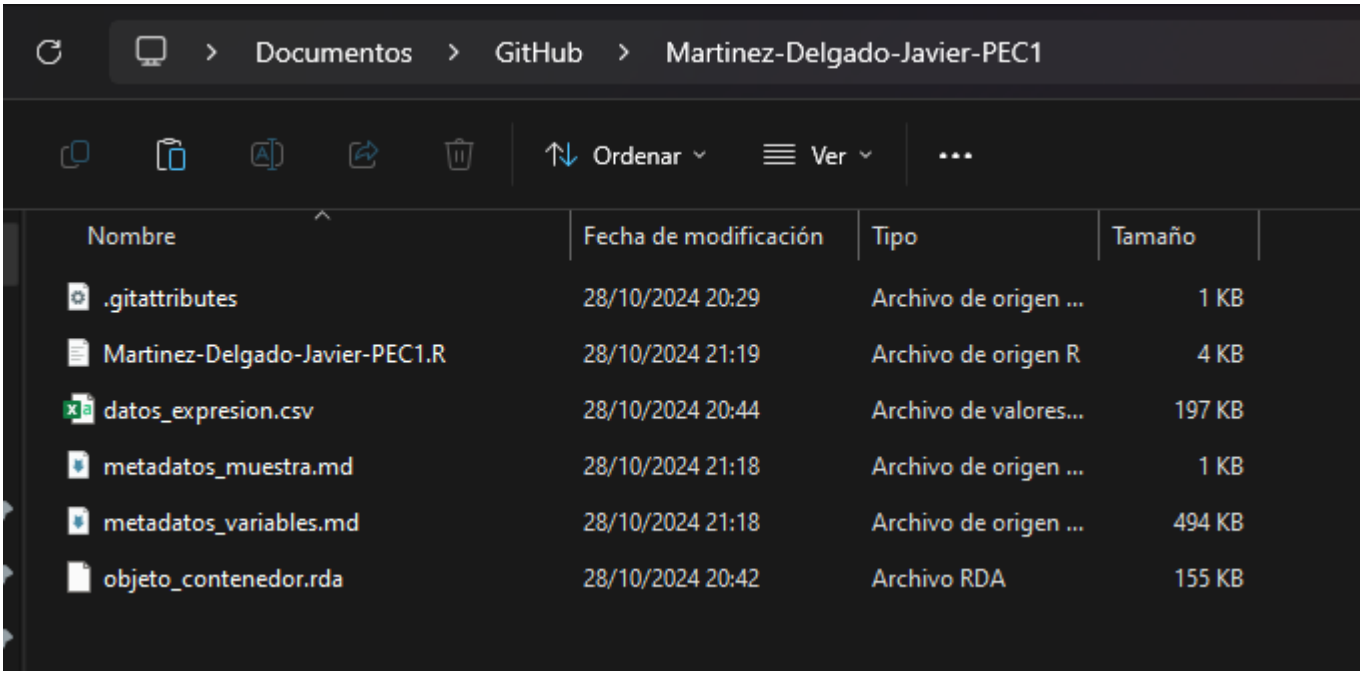


Como se puede ver a simple vista en el PCA, hay 3 grupos principales y Sólo quedan muestras del tipo PD, por lo que se han eliminado todas las muestras de tipo MSS

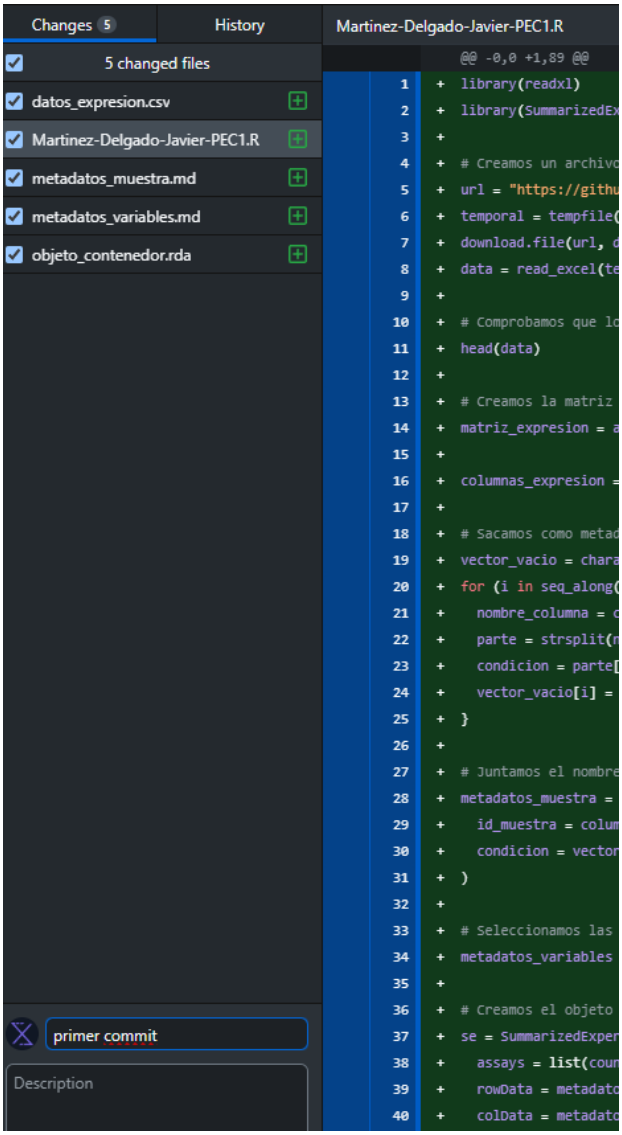
4.-Reposición de los datos en GitHub

En primer lugar, creamos el repositorio dentro de nuestra cuenta (en mi caso lo hago en github desktop ya que me es más fácil al tenerlo ya configurado y usarlo a menudo)





Metemos los ficheros en la carpeta del repositorio



Y hacemos un commit de los archivos

https://github.com/xshaikz/Martinez-Delgado-Javier-PEC1