PEC 1

ARANTXA GARCIA REDON

2025-03-30

Link al repositorio Github: https://github.com/arared12/Garcia-Redon-Arantxa-PEC1

Explicación de los contenidos del repositorio : pongo los contenidos requeridos por el enunciado de la PEC seguidos del nombre del archivo en el repositorio.

• el informe -> Garcia_Redon_Arantxa_PEC1.pdf • el objeto de clase SummarizedExperiment que contenga los datos y los metadatos en formato binario (.Rda) -> cachexia_se.Rda • el código R para la exploración de los datos debidamente comentado (el control de versiones del mismo debe realizarse con Git) -> Garcia_Redon_Arantxa_PEC1.Rmd • los datos en formato texto -> datos.txt • los metadatos acompañados de una breve descripción en un archivo markdown. -> metadatos_experimento.md

En primer lugar, vamos a cargar los paquetes necesarios para resolver la PEC y también se importarán los datos para el análisis. En mi caso, he decidido trabajar con el conjunto de datos human_cachexia. He seleccionado el conjunto de datos porque es sencillo, pero versátil en cuanto a los análisis que se puede hacer con él. Se trata de datos numéricos de una serie metabolitos de varios pacientes.

Hay un grupo control (sano) y un grupo caquéxico. En los metadatos se indica que las muestras no son pareadas, es decir, que no se trata de datos de una misma persona que en un momento era normal y despues desarrolla la enfermedad, por lo que nos da una guía de los tests estadísticos que podemos emplear y los que no. Nos dice además que todos los datos son numéricos y que no hay datos nulos.

```
library(SummarizedExperiment)
# Se cargan los datos y los metadatos para trabajar con ellos
datos <-read.csv("human_cachexia.csv")
metadatos <-readLines("description.md")
# Se visualizan los datos.
# De nuevo he empleado
print(metadatos)
head(datos)
```

Este conjunto de datos tiene la siguiente estructura:

- Las filas representan los diferentes pacientes del estudio. Algunos son caquéxicos y otros no lo son.
- En las columnas tenemos los diferentes datos que se recogen de cada paciente. La primera columna se trata de el identificador del paciente, la segunda su condicion de salud que puede ser normal o caquéxico y el resto de las columnas son diferentes metabolitos que se han recogido de cada uno de los pacientes.

1. Creación de un objeto SummarizedExperiment y estudio de las diferencias con la clase ExpressionSet

Para la creación del objeto SummarizedExperiment (de ahora en adelante, SE), he accedido a la documentación oficial de Bioconductor (https://www.bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html#subsetting).

En primer lugar, mirando el apartado de la anatomía de un objeto SE, se puede observar la necesidad de trasponer los datos dado que en un objeto SE las diferentes muestras(pacientes) están en las columnas y en

las filas se deben recoger los features, que en este caso son los metabolitos. Además se admite que se tenga más de un ensayo para unas mismas samples y features y unos metadatos, que se almacenen por separado pero ligados al conjunto de datos principal.

En el código de a continuación se va a crear el objeto SE tras preprocesar los datos para poder llevar a cabo su creación.

```
# Creo una nueva matriz que es la traspesta de datos, en la cual elimino
# las dos primeras columnas para quedarme solo con los datos numéricos
datos_num<-t(datos[,-c(1,2)])
# El  coldatos contendrá los valores de la segunda columna de datos.
# Es decir el dato de control o enfermos guardado en un vector
coldatos<-datos[,2]
# Convertimos coldatos en un dataframe que contiene el ID del paciente y su condicion
coldatos<-data.frame(SampleName=datos[,1], Muscle.loss=datos[,2])
#Creamos el objeto SE.
se<-SummarizedExperiment(assays=list(counts=datos_num), colData=coldatos, metadata=metadatos)
```

El objeto SE contiene los siguientes componentes:

- `assays = list(counts = datos_num)`: esto contendrá los datos numéricos que hacen referencia a las cantidades de metabolitos de la matriz transpuesta datos_num.
- `colData = coldatos`: contiene los metadatos asociados a cada columna (es decir, a las muestras). En este caso, se incluye la información del dataframe coldatos, que contiene:
  - SampleName (nombre de la muestra)
  - Muscle.loss (información sobre la pérdida muscular)
- `metadata = metadatos`: Este aargumento nos permite introducir los metadatos en formato texto que hemos importado del archivo description.

En la PEC nos pide que comparemos SE con ExpressionSet. Ambas son clases, o estructuras de datos de Bioconductor que son frecuentemente utilizadas para representar y manejar datos experimentales de datos ómicos. Ambas se usan mucho en estudios de expresión génica.

Para comprender mejor la clase ExpressionSet, de nuevo se ha consultado la página oficial de bioconductor (https://www.bioconductor.org/packages/devel/bioc/vignettes/Biobase/inst/doc/ExpressionSetIntroduction.pdf). La principal diferencia entre los dos paquetes parece ser la flexibilidad en cuanto a los tipos de datos que pueden almacenarse en cada una de las estructuras de datos. Mirando diferentes repositorios en github y stackoverflow se puede ver que ExpressionSet se utiliza para estudios de expresión génica con microarrays de forma casi exclusiva. Sin embargo, SummarizedExperiment es más flexible y permite trabajar con datos de RNAseq y otros datos ómicos.

Otra diferencia es como se estucturan los datos:

- `SummarizedExperiment` se estructura en:
  - assays: que es una matriz o dataframe con los datoss experimentales (por ejemplo los datos de expresión o de metabolómica como en el dataset que estamos usando)
  - `coldata`: un dataframe que contiene los metadatos de las muestras.
  - `rowdata`: un dataframe con los metadatos sobre las filas, como los genes
  - `metadata`: un apartado para metadatos generales del experimento, donde en este caso hemos almacenado la descripción.
- `ExpressionSet` tiene tres componentes principales:
  - `exprs`: una matriz de datos de expresión con los genes en filas y las muestras en columnas.
  - `phenodata`: un objeto que contiene metadatos sobre las muestras, como por ejemplo las condiciones del experimento.
  - `featureData`que contiene información sobre los genes del experimento.

En general, podemos decir que las diferencias en su estructura hacen que `ExpressionSet`es una estructura de datos más rígida que puede resultar muy útil en experimentos de expresión génica con microarrays y esta optimzado para este tipo de análisis, mientras que `SummarizedExperiment`tiene una estructura más flexible que nos permite almacenar más datos y es útil para otros tipos de datos ómicos, no solo de microarrays.

Exportando los diferentes resultados

```
library(rmarkdown)
```

```
## Warning: package 'rmarkdown' was built under R version 4.4.2
```

```
#Exportamos los archivos que pide la PEC
# Exportamos los datos de summarizedExperiment en formato .Rda
save(se, file = "cachexia_se.Rda")
# Exportamos los datos numéricos como archivo de texto
write.table(datos, file = "datos.txt", row.names = FALSE, col.names = FALSE)
#Exportamos los metadatos como Rmarkdown
# Añado aquí la descripción
descripcion <- "### Estos son los metadatos que venían con el conjunto de datos de
caquexia. En el análisis se va a crear un summarized experiment. Con los datos
se va a hacer un análisis de datos sencillo. He recogido tanto los datos de
las columnas, como los metadatos del archivo de metadatos incial en este Markdown"
# Genero el texto que se exportará en el archivo Markdown
markdown_content <- c(
  descripcion,
  "\n## Metadatos del Experimento\n",  # Título metadatos experimentales
  "### Descripción de las columnas\n", # Subtitulo
  paste("**Muestra**: Identificador para cada paciente."),
  paste("**Pérdida Muscular (Muscle.loss)**: Condición del paciente: `control` o `cachexia`."),
  "\n",
  "### Metadatos del objeto SummarizedExperiment:\n",
  paste("```r\n", (coldatos), "\n```"), # Salto de linea e imprimo coldatos
  paste("\n", "###Metadatos del archivo metadatos", "\n"),
  paste("\n", (metadatos), "\n")
  )

# Escribir el contenido en un archivo Markdown
writeLines(markdown_content, con = "metadatos_experimento.md")
```

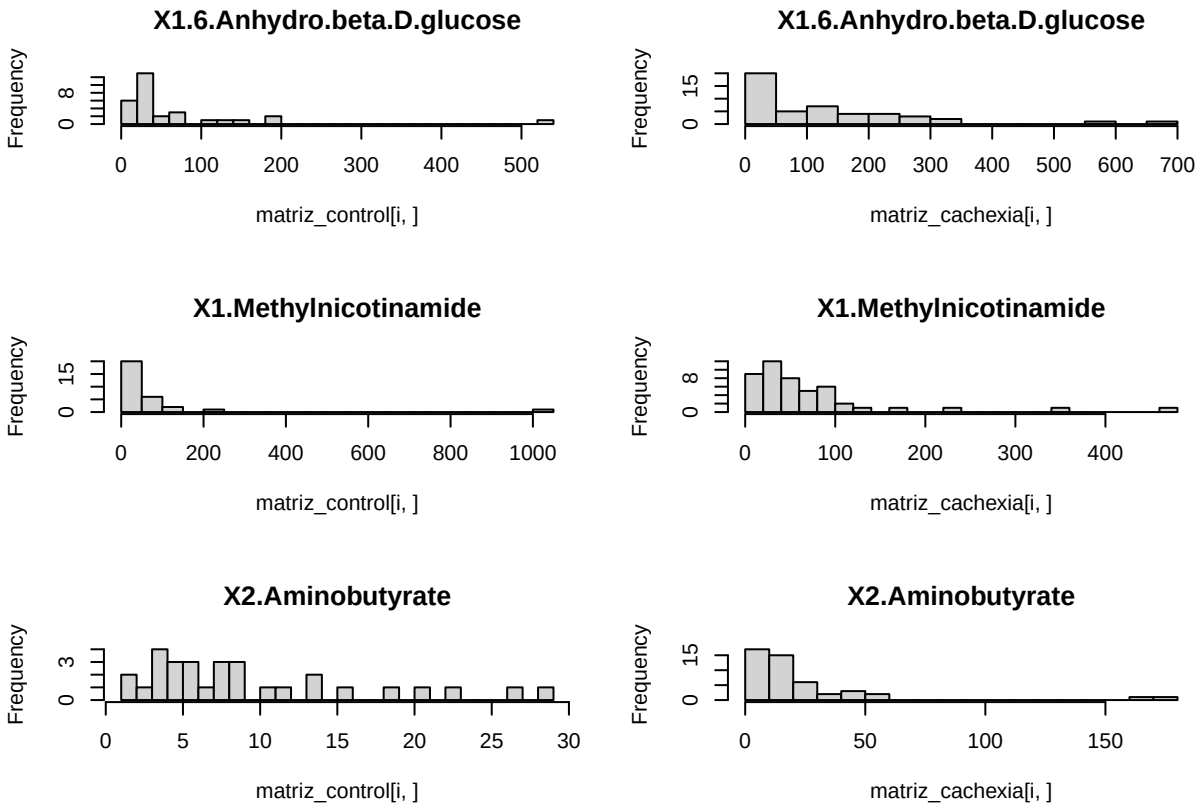
2. Análisis exploratorio de los datos del dataset cachexia.

Para poder comparar los datos entre el grupo de caquexia y el grupo control, voy a crear dos matrices distintas con los datos de cada uno de los grupos.

Después voy a comenzar con el análisis exploratorio de alguno de los datos. En primer lugar, utilizo la función `par()` para representar histogramas con los niveles de los 3 primeros metabolitos de las matrices control y caquexia. Como todavía no se ha explorado en profundidad los datos, no obtenemos grandes diferencias. Podría ser más interesante repetir esto más tarde representando los metabolitos que se expresan de forma más diferente entre los dos grupos. Sin embargo, se incluye dado que es algo que se ha hecho en las actividades previas del reto y en ocasiones puede resultar útil para detectar tendencias en los datos.

```
# Filtro el dataset y separo en grupo control y caquexia
matriz_control<-assays(se[,se$Muscle.loss=="control"])$counts
matriz_cachexia<-assays(se[,se$Muscle.loss=="cachexic"])$counts
# Representacion de los 3 primeros metabolitos en control y casquexia
```

```
par(mfrow=c(3,2))
for(i in 1:3){
  hist(matriz_control[i,], main=rownames(matriz_control)[i], breaks = 20)
  hist(matriz_cachexia[i,], main=rownames(matriz_cachexia)[i], breaks = 20)
}
```



De estos histogramas se puede observar a priori que los niveles de los metabolitos se distribuyen de manera ligeramente diferente en los dos grupos. En el caso del aminobutirato, no se ve tan claro porque la escala de los dos ejes es muy diferente, pero en general, parece verse que los niveles de los metabolitos están ligeramente aumentados en el grupo de caquexia. Sin embargo, no podemos hacer afirmaciones en base solo a esta representación por lo que posteriormente se hará un análisis más detallado para ver si los metabolitos se encuentran en mayores niveles en el grupo de caquéxicos o no.

En este punto, vamos a comenzar a dirigir el análisis exploratorio de los datos a encontrar diferencias entre el grupo control y el enfermo. Para estudiar la diferencia entre las medias de un grupo y el otro, vamos a comprobar mediante el test de Shapiro Wilk la normalidad de los datos. En los histogramas, podemos ver que seguramente no sigan una distribución normal, pero lo comprobaremos matemáticamente. Como se ve más adelante, los datos no son normales, por lo que se empleará el test de Wilcoxon bilateral para comprobar si las diferencias entre las medias de los dos grupos son significativamente diferentes.

```
# Test de normalidad de Shapiro-Wilk
normalidad = function(x) {
  # Realiza el test de Shapiro-Wilk
  return(shapiro.test(x)$p.value)
}
# Aplica el test de normalidad a cada fila de la matriz de datos
normality_results <- apply(datos_num, 1, normalidad)
```

```r
# Identificamos los metabolitos normales (p-valor > 0.05)
normal_metabolites <- rownames(datos_num)[normality_results >= 0.05]
length(normal_metabolites)
```

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

```r
# Identificamos los metabolitos no normales (p-valor < 0.05)
no_normal <- rownames(datos_num)[normality_results < 0.05]
length(no_normal)
```

```
## [1] 63
```

```r
# Función que realiza el test de Wilcoxon para comparar los grupos control y caquexia
wilcoxon_test = function(x) {
  resultado = wilcox.test(x[1:47], x[48:77])
  # La funcion devuelve el estadístico W y el p-valor
  return(c(resultado$statistic, resultado$p.value))
}

# Aplicamos la funcion a datos_num
ans <- apply(datos_num, 1, wilcoxon_test)
ws <- ans[1,]  # Estadístico W
pvalues <- ans[2,]  # p-valor del test de Wilcoxon

# Identificar metabolitos con p-valores < 0.05 (media significativamente diferente)
sign_mol <- rownames(matriz_cachexia)[pvalues < 0.05]
print("Metabolitos significativos:")
```

```
## [1] "Metabolitos significativos:"
```

```r
length(sign_mol)
```

```
## [1] 55
```

Se ha obtenido el número de las moléculas que están diferencialmente presentes en los pacientes normales y con caquexia. Para saber todavía mejor que variables son las más significativamente diferentes, voy a crear grupos según diferentes umbrales de p-valor. Se obtiene que hay 8 compuestpos cuyo p-valor es inferior a 0.0001, se imprimen estos metabolitos.

```r
for (i in c(0.05,0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001)){
  print(paste("Compuestos con p-valores inferiores a", i, length(which(pvalues < i))))
}
```

```
## [1] "Compuestos con p-valores inferiores a 0.05 55"
## [1] "Compuestos con p-valores inferiores a 0.01 44"
## [1] "Compuestos con p-valores inferiores a 0.001 27"
## [1] "Compuestos con p-valores inferiores a 1e-04 8"
## [1] "Compuestos con p-valores inferiores a 1e-05 0"
## [1] "Compuestos con p-valores inferiores a 1e-06 0"
## [1] "Compuestos con p-valores inferiores a 1e-07 0"
```

```r
muy_distintos <- c(rownames(matriz_cachexia)[pvalues < 0.0001])
print("Metabolitos con p-valor inferior a 0.0001:")
```

```
## [1] "Metabolitos con p-valor inferior a 0.0001:"
```

```r
print(muy_distintos)
```

```
## [1] "X3.Hydroxyisovalerate" "Adipate"               "Betaine"
```

```
## [4] "Glucose"               "Leucine"               "N.N.Dimethylglycine"
## [7] "Quinolinate"           "Valine"
```

Ahora, se va a calcular la matriz de correlación entre los diferentes metabolitos para cada uno de los dos grupos, de manera que podremos ver de manera aproximada las relaciones entre los diferentes metabolitos. Valores cercanos a 1 en la matriz de correlación nos indicarían una fuerte relación entre las variables. Por otra parte, los valores pueden ser positivos (indicando que ambas variables aumentan juntas) o negativos, indicando que cuando uno de los valores aumenta, el otro disminuye.

En este caso, el resultado de la matriz no se va a incluir en el informe, pero en general se observa que la correlación a penas existe en la mayoría de los casos. Si bien si que hay algunos metabolitos con una correlación de 0,8 que sería una correlación fuerte y positiva, en términos generales, no podemos decir que haya muchas variables fuertemente correlacionadas entre sí. Lo que he podido observar es que algunos de los metabolitos analizados son aminoácidos. Los diferentes aminóacidos presentan valores relativamente altos en la matriz de correlación (valores de entre 0.6 y 0.8), lo cual tiene sentido porque son componentes fundamentales del músculo y tiene sentido que encontremos una correlación positiva y relativamente fuerte.

```
correlacion <- cor(t(assays(se)$counts))
```

A continuación se pretende identificar si en el grupo de caquexia hay metabolitos que se encuentran en niveles más bajos que en el grupo de control. Inicialmente, se había planteado lo contrario, si hay algunos que estén presentes en mayores cantidades, pero como tan solo hay dos metabolitos cuya media es inferior en el grupo de caquéxicos que en el grupo control, he decidido hacer el análisis contrario. En primer lugar, se hace una comparación de las medias de cada metabolito entre los dos grupos. Como solo se obtienen 2 metabolitos, se hace un test t de una cola para comprobar si estos metabolitos se encuentran en cantidades significativamente menores en el grupo.

```
media_caq <- rowMeans(matriz_cachexia)
media_cnt <-rowMeans(matriz_control)
under <- rownames(matriz_cachexia)[media_caq<media_cnt]
print(paste("La media del metabolito", under, "parece ser menor en el grupo control que
            en el de caquexia."))
```

```
## [1] "La media del metabolito X1.Methylnicotinamide parece ser menor en el grupo control que\n
## [2] "La media del metabolito Pantothenate parece ser menor en el grupo control que\n            en el
```

```
# Metabolitos a analizar
metabolitos <- c("X1.Methylnicotinamide", "Pantothenate")
# Vector vacio de longitud 2
pvalores <- numeric(length(metabolitos))

# Iteramos sobbre los dos metabolitos de interes
for (i in 1:length(metabolitos)) {
  metabolito <- metabolitos[i]
  # Datos del metabolito
  control_data <- assays(se[, se$Muscle.loss == "control"])$counts[metabolito, ]
  cachexia_data <- assays(se[, se$Muscle.loss == "cachexic"])$counts[metabolito, ]
  # One-tailed t-test
  wil_res <- wilcox.test(control_data, cachexia_data, alternative = "greater")
  pvalores[i] <- wil_res$p.value
}
```

```
## Warning in wilcox.test.default(control_data, cachexia_data, alternative =
## "greater"): cannot compute exact p-value with ties
## Warning in wilcox.test.default(control_data, cachexia_data, alternative =
## "greater"): cannot compute exact p-value with ties
```

```r
names(pvalores) <- metabolitos
print("Los p-valores son:")
```

```
## [1] "Los p-valores son:"
```

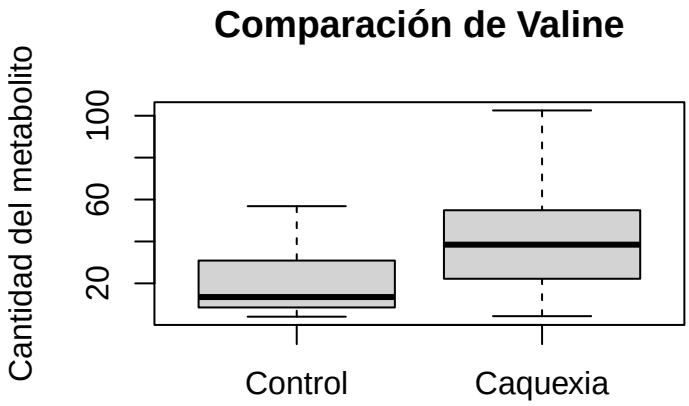
```r
pvalores
```

```
## X1.Methylnicotinamide          Pantothenate
##             0.9832380             0.9673865
```

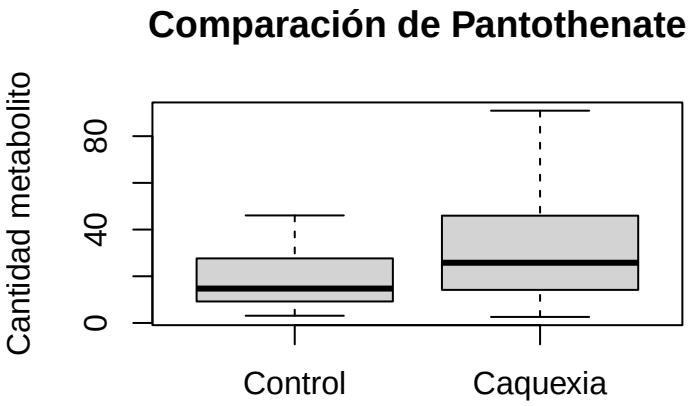
Dado que los p-valores obtenidos son elevados, no contamos con evidencia estadística suficiente para concluir que los metabolitos se encuentren en menor cantidad en el grupo de caquexia en comparación con el grupo control.

Para explorar visualmente las diferencias en las distribuciones, representaremos primero los boxplots de la valina, ya que hemos observado que existe una diferencia significativa en la media entre los grupos control y caquexia. Posteriormente, se presentará el boxplot del pantotenato, en el cual hemos comprobado que las diferencias entre ambos grupos no son significativas. En ambos casos se incluye el argumento outline para eliminar los outliers.

```r
# Boxplot para la valina
boxplot(matriz_control[57,], matriz_cachexia[57,], outline=FALSE,
        names=c("Control", "Caquexia"),
        main=paste("Comparación de", rownames(matriz_control)[57]),
        ylab="Cantidad del metabolito")
```



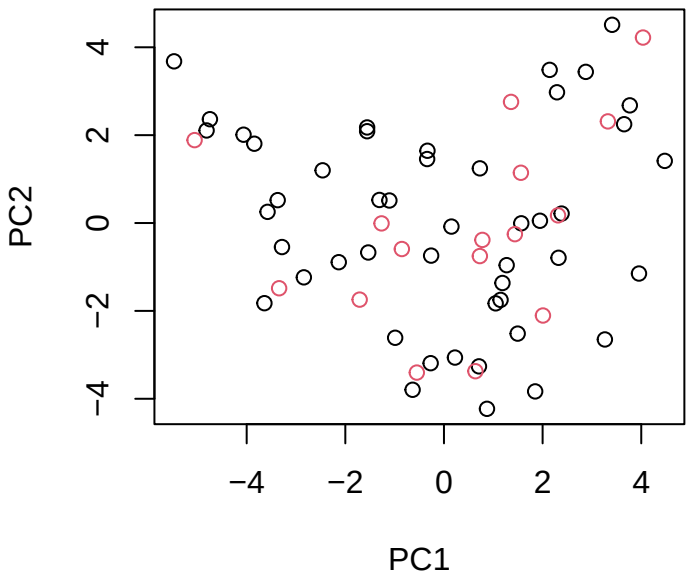

```r
# Boxplot para el pantotenato
boxplot(matriz_control[42,], matriz_cachexia[42,], outline=FALSE,
        names=c("Control", "Caquexia"),
        main=paste("Comparación de", rownames(matriz_control)[42]),
        ylab="Cantidad metabolito")
```



Análisis multivariante

Ahora se va a llevar a cabo un análisis de la componente principal (PCA). El primer paso siempre es normalizar los datos, para que todas las variables tengan media 0 y varianza 1. Esto se hace porque es un análisis sensible a la magnitud de las variables y si por ejemplo hay una variable del orden de 1000 y otra del orden de 3, esto afectaría al análisis.Después con la función prcomp se hace el análisis PCA. Finalmente en pesos se calcula el porcentaje de la varianza explicado por cada componente dividido por la varianza total. El gráfico del PCA generado mostrará las muestras agrupadas según la variable muscle.loss y tendrá en el eje X los valores del primer componente principal y en el eje Y los valores del segundo.

```
norm_data<-t(scale(t(assays(se)$counts)))
PCAS2 <- prcomp(norm_data)
pesos<-PCAS2$sdev^2/sum(PCAS2$sdev^2)*100
par(mfrow=c(1,1))
plot(PCAS2$x[,1:2], col=as.factor(coldatos$Muscle.loss))
```




```
# Comentado porque no aporta nada, no se observa ningun tipo de cluster
## y queda poco espacio
## clust.euclid.average <- hclust(dist(t(norm_data)),method="average")
##plot(clust.euclid.average, hang=-1, col=as.factor(coldades$Muscle.loss))
```

En el gráfico, no podemos observar ningún tipo de agrupación en el gráfico de PCA. Esto podría deberse a que en realidad no tenemos diferencias muy marcadas entre los datos de los metabolitos en los dos grupos, si que hay algun metabolito donde hay una diferencia notable pero, en general no tenemos diferencias muy marcadas, lo que podría ser la causa de que no veamos agrupamientos en este análisis.

Para finalizar se va a representar un vulcano plot.