DE_Cyt_vs_Nuc

Núria Rivera Brugués

2023-10-02

```
counts <- read.csv("/home/nrb/Documentos/counts/counts2_reduced_ANNA.csv", row.names = 1, header=T, sep=

info <- read.delim("~/Documentos/counts_RSEM/targets.csv", sep=",")
colnames(counts)<-info[,14]
rownames(info)<-info$Sample
barcode=factor(info$Sample)
subsample=factor(info$Subgroup)
group=factor(info$Group)
time=factor(info$Time)
```

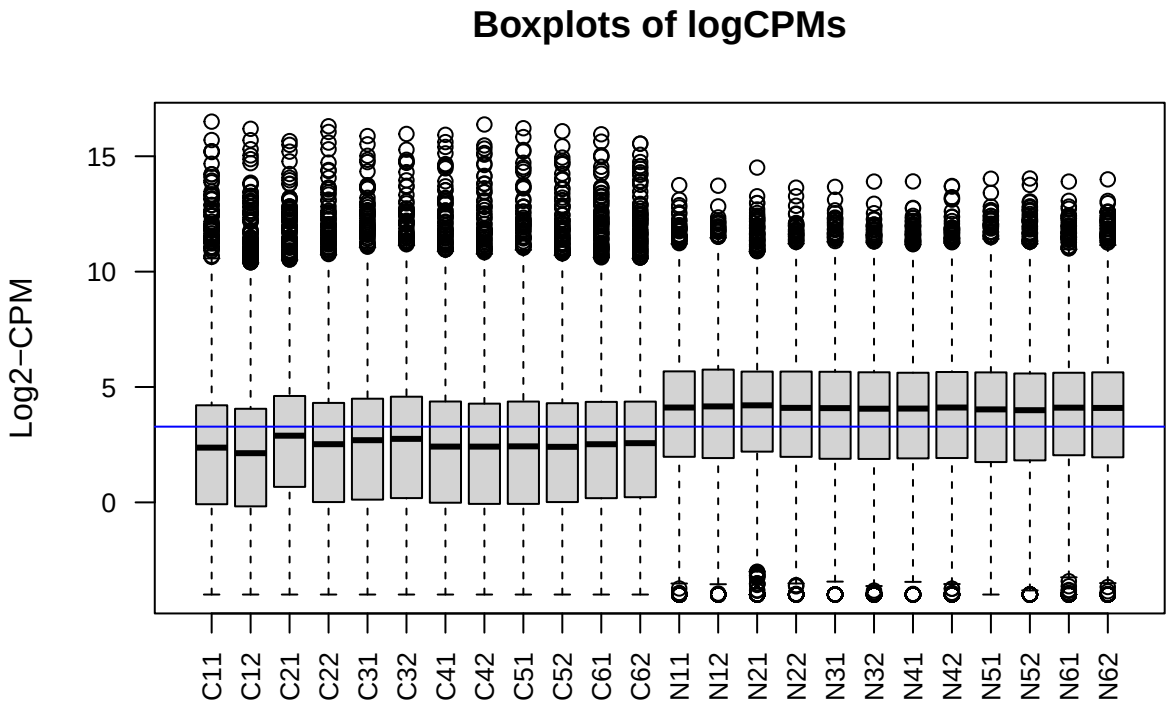
```
library(edgeR)
counts.CPM <- cpm(counts)
head(counts.CPM)
```

```
##                              C11         C12          C21          C22
## ENSMUSG00000000001.5   94.4627841  63.1378088 9.644922e+01  89.28485014
## ENSMUSG00000000003.16   0.0000000   0.0000000 0.000000e+00   0.00000000
## ENSMUSG00000000028.16   0.5111120   0.5101252 9.030175e-01   0.63061695
## ENSMUSG00000000031.19   0.1393942   0.2354424 1.458721e+00   0.05255141
## ENSMUSG00000000037.18   0.0000000   0.0000000 3.473144e-02   0.00000000
## ENSMUSG00000000049.12 950.4823958 945.6937180 1.044097e+03 649.90332061
##                              C31          C32         C41         C42
## ENSMUSG00000000001.5    79.7410918  98.85565291 111.1020926  97.0662279
## ENSMUSG00000000003.16    0.0000000   0.00000000   0.0000000   0.0000000
## ENSMUSG00000000028.16    0.9043615   1.34210545   0.9668057   0.8518140
## ENSMUSG00000000031.19    0.2359204   0.11503761   0.2417014   0.1216877
## ENSMUSG00000000037.18    0.0000000   0.03834587   0.0000000   0.0000000
## ENSMUSG00000000049.12 1348.7568689 909.64072856 931.9604465 914.1180573
##                              C51          C52         C61          C62
## ENSMUSG00000000001.5  124.7032634  105.8056272 110.6704241  108.9363080
## ENSMUSG00000000003.16   0.0000000    0.0000000   0.0000000    0.0819378
## ENSMUSG00000000028.16   0.8551506    0.4729498   0.7939055    0.7374402
## ENSMUSG00000000031.19   0.3346241    0.3040392   0.1190858    0.1638756
## ENSMUSG00000000037.18   0.0000000    0.0000000   0.0000000    0.0000000
## ENSMUSG00000000049.12 865.7841958 1140.7887048 891.6352458 1021.4776113
##                              N11         N12          N21          N22
## ENSMUSG00000000001.5   49.82996223  50.3436243  37.3733453  37.55031700
## ENSMUSG00000000003.16   0.00000000   0.0000000   0.0000000   0.00000000
## ENSMUSG00000000028.16   2.55385980   2.8410623   1.8487754   1.97633247
## ENSMUSG00000000031.19   0.02969604   0.2045565   0.2340222   0.04940831
## ENSMUSG00000000037.18   0.14848022   0.1818280   0.3276311   0.14822494
```

```
## ENSMUSG00000000049.12 363.62806170 354.1554644 331.5860544 308.92546976
##                                N31         N32          N41          N42
## ENSMUSG00000000001.5    37.9161167  46.2473822  36.06398226  44.57865053
## ENSMUSG00000000003.16    0.0000000   0.0000000   0.00000000   0.00000000
## ENSMUSG00000000028.16    1.8472757   2.3595603   2.23046316   3.04678915
## ENSMUSG00000000031.19    0.1878585   0.0314608   0.05793411   0.04650876
## ENSMUSG00000000037.18    0.1565488   0.2516864   0.14483527   0.16278067
## ENSMUSG00000000049.12 288.3315593 403.8623436 427.00334337 382.48807714
##                                N51         N52          N61          N62
## ENSMUSG00000000001.5    37.64270685  36.6120638  37.63667270  43.61372848
## ENSMUSG00000000003.16    0.00000000   0.0000000   0.00000000   0.00000000
## ENSMUSG00000000028.16    1.94034571   2.3245755   3.36843729   1.95208679
## ENSMUSG00000000031.19    0.02587128   0.0000000   0.08982499   0.07705606
## ENSMUSG00000000037.18    0.07761383   0.1010685   0.15719374   0.20548282
## ENSMUSG00000000049.12 241.97404616 266.2902279 428.51013636 352.58283323
```

```
y=DGEList(counts=counts)
isexpr <- rowSums(cpm(y) > 1) >= 3
y=y[isexpr,keep.lib.size=FALSE]
```

```
logcounts <- cpm(y,log=TRUE)
boxplot(logcounts, ylab="Log2-CPM",las=2, xlab="", cex.axis=0.8, main="Boxplots of logCPMs")
abline(h=median(logcounts), col="blue")
```




```
y=calcNormFactors(y)
y$samples
```

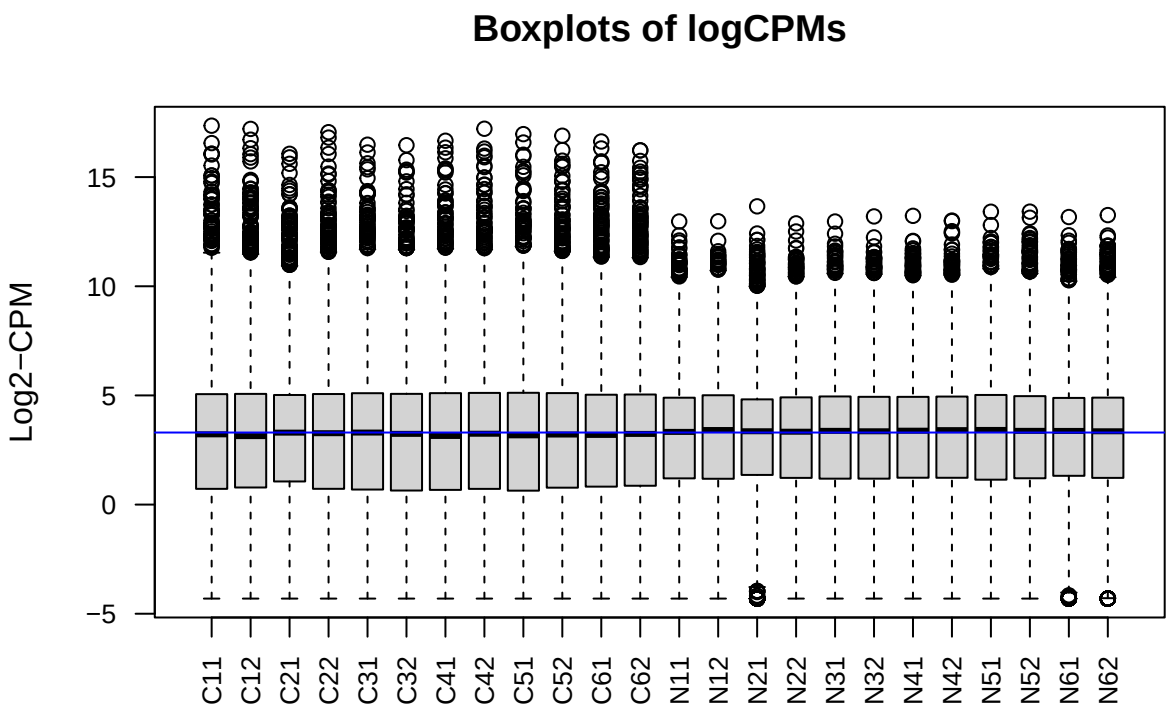
```
##     group lib.size norm.factors
## C11     1 21511269    0.5528799
## C12     1 25465491    0.4939034
```

```
## C21     1 28765354    0.7523256
## C22     1 19017184    0.5919619
## C31     1 25417658    0.6551185
## C32     1 26060576    0.7106147
## C41     1 24809152    0.6006614
## C42     1 24641025    0.5592276
## C51     1 26880823    0.5934393
## C52     1 29583817    0.5700261
## C61     1 25176053    0.6247366
## C62     1 24393748    0.6253469
## N11     1 33614117    1.7200796
## N12     1 43918939    1.6778225
## N21     1 42631575    1.8009266
## N22     1 40400919    1.6917930
## N31     1 31884720    1.6342908
## N32     1 31730598    1.6280071
## N41     1 34461381    1.6056080
## N42     1 42926568    1.6274034
## N51     1 38593150    1.5288973
## N52     1 39513412    1.5365525
## N61     1 44448306    1.6615218
## N62     1 38865391    1.6675397
```

```
dim(y)
```

```
## [1] 15432    24
```

```
logcounts_norm <- cpm(y,log=TRUE)
boxplot(logcounts_norm, ylab="Log2-CPM",las=2, xlab="", cex.axis=0.8, main="Boxplots of logCPMs")
abline(h=median(logcounts_norm), col="blue")
```

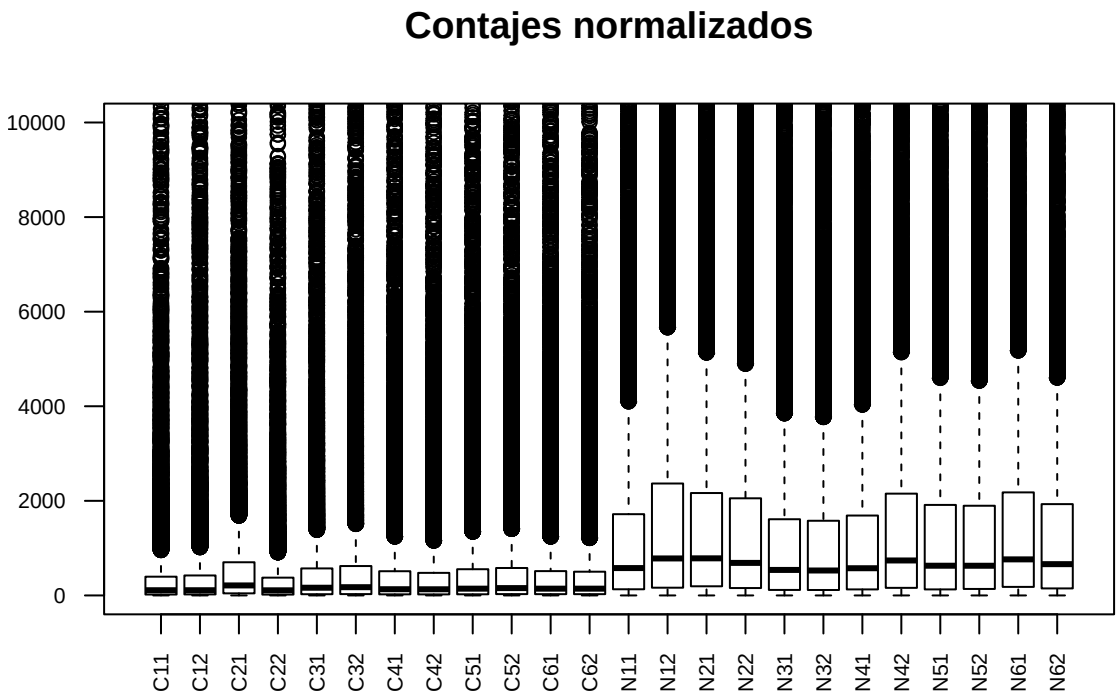


Exploración de los datos

Una vez descartados los genes poco expresados y con los recuentos almacenados en un objeto DGEList, podemos‘proceder a realizar algunos gráficos exploratorios para determinar si los datos aparentan buena calidad y/o si presentan algun problema.

Distribución de los contajes

```
boxplot(y$counts, col = y$samples$cols, las = 2, cex.axis = 0.7,
    main = "Contajes normalizados", ylim = c(0, 10000))
```



Análisis de similaridad entre las muestras

Distancia entre muestras

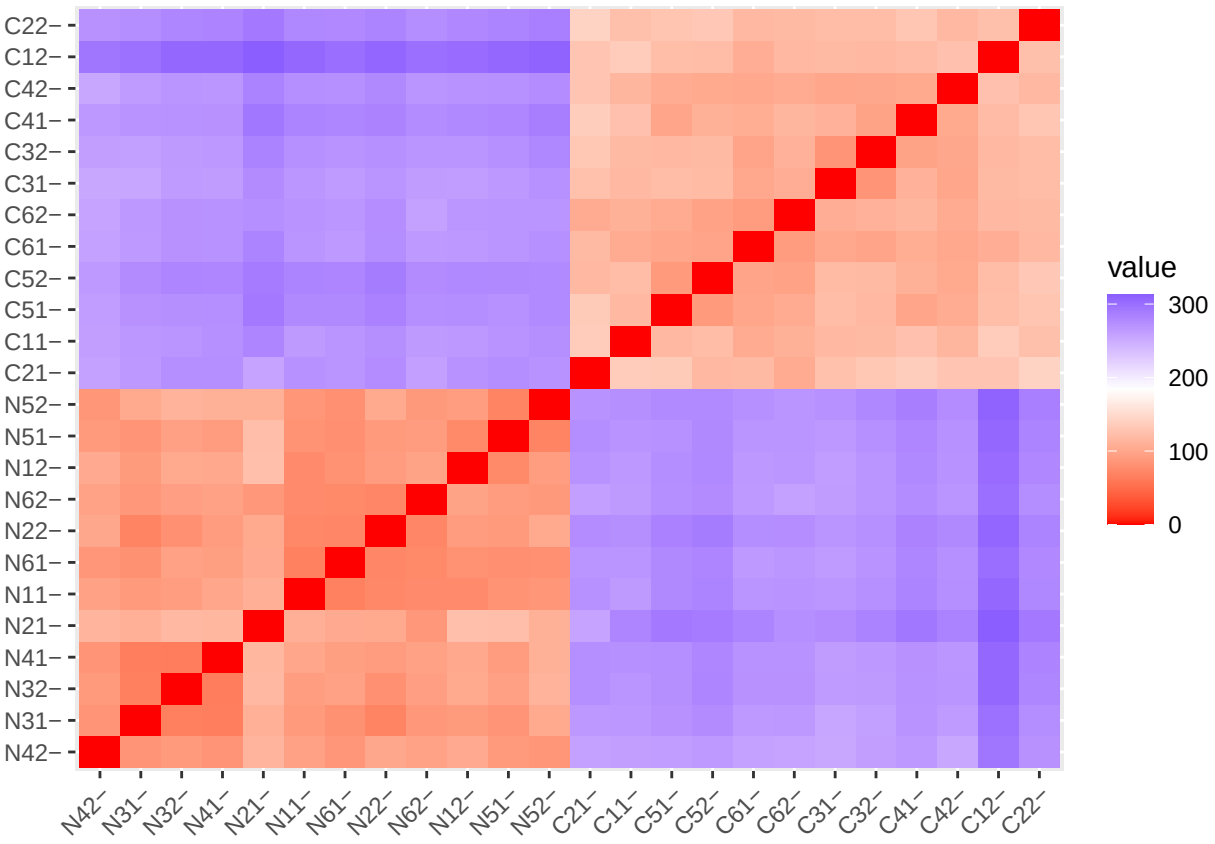
La función dist permite calcular una matriz de distancias que contiene las comparaciones dos a dos entre todas las muestras. Por defecto se utiliza una distancia euclídea.

```
log2count_norm <- cpm(y, log = TRUE)
sampleDists <- dist(t(log2count_norm))
round(sampleDists, 1)
```

```
##       C11   C12   C21   C22   C31   C32   C41   C42   C51   C52   C61   C62
## C12 135.4
## C21 135.6 128.8
## C22 123.4 124.0 142.0
## C31 116.4 118.2 124.5 120.3
```

```
## C32 118.0 116.9 131.1 120.7  83.7
## C41 124.8 119.1 136.1 129.8 109.9  96.5
## C42 114.2 124.9 128.6 117.5  99.7 100.7 102.6
## C51 117.2 121.7 134.3 128.1 120.9 115.9  98.7 105.4
## C52 119.6 120.2 117.2 131.7 118.8 118.4 109.3 102.9  88.1
## C61 103.8 107.2 117.8 116.2 101.4  97.4 105.5 101.5  99.3  97.3
## C62 109.5 116.8 103.8 117.6 107.1 110.4 114.1 103.9 104.0  95.8  90.7
## N11 264.3 305.0 271.7 279.8 267.3 272.8 282.2 274.3 278.1 282.3 268.7 269.7
## N12 265.6 301.1 270.6 279.6 262.3 268.5 279.0 271.0 275.0 278.8 266.3 267.8
## N21 281.5 312.5 257.4 291.1 276.6 283.9 292.9 283.9 290.7 289.3 282.2 274.1
## N22 273.4 306.4 275.5 281.9 268.1 272.8 283.5 278.1 284.2 288.8 275.1 274.5
## N31 266.7 297.3 266.2 275.1 254.4 259.9 269.7 263.2 271.5 276.7 265.0 265.4
## N32 268.6 304.8 274.0 280.9 263.3 264.7 270.6 268.8 273.6 281.1 271.3 271.2
## N41 272.4 304.9 273.9 282.6 263.0 266.1 271.3 267.6 274.1 280.4 270.8 271.1
## N42 259.9 293.8 258.6 271.4 253.4 260.7 265.6 253.7 261.6 265.0 258.5 257.2
## N51 269.6 305.2 274.8 281.8 266.2 272.5 280.2 271.9 271.8 278.0 268.5 268.9
## N52 273.4 308.3 271.1 285.8 271.8 279.9 287.1 275.7 277.5 277.1 272.3 268.8
## N61 268.4 300.2 268.4 278.6 263.7 270.3 280.7 272.7 277.9 280.9 265.2 267.4
## N62 264.2 298.9 259.3 275.2 262.6 268.4 276.0 269.1 274.4 276.2 265.2 258.7
##       N11   N12   N21   N22   N31   N32   N41   N42   N51   N52   N61
## C12
## C21
## C22
## C31
## C32
## C41
## C42
## C51
## C52
## C61
## C62
## N11
## N12  75.5
## N21 108.2 122.9
## N22  72.4  90.7 102.7
## N31  88.3  89.2 108.3  69.0
## N32  91.0 102.2 116.4  79.3  65.8
## N41  99.6 101.4 115.6  89.6  63.9  63.5
## N42  95.0 103.1 112.9 100.5  83.1  88.4  83.6
## N51  82.2  74.1 122.2  88.7  83.1  94.2  90.5  88.3
## N52  85.0  91.6 109.7 101.9 102.8 111.5 109.3  84.6  69.5
## N61  66.7  81.0 103.6  70.8  80.9  95.1  92.8  85.4  78.8  80.4
## N62  75.7  96.4  86.3  71.2  86.2  92.6  94.6  95.4  90.5  87.4  74.3
```

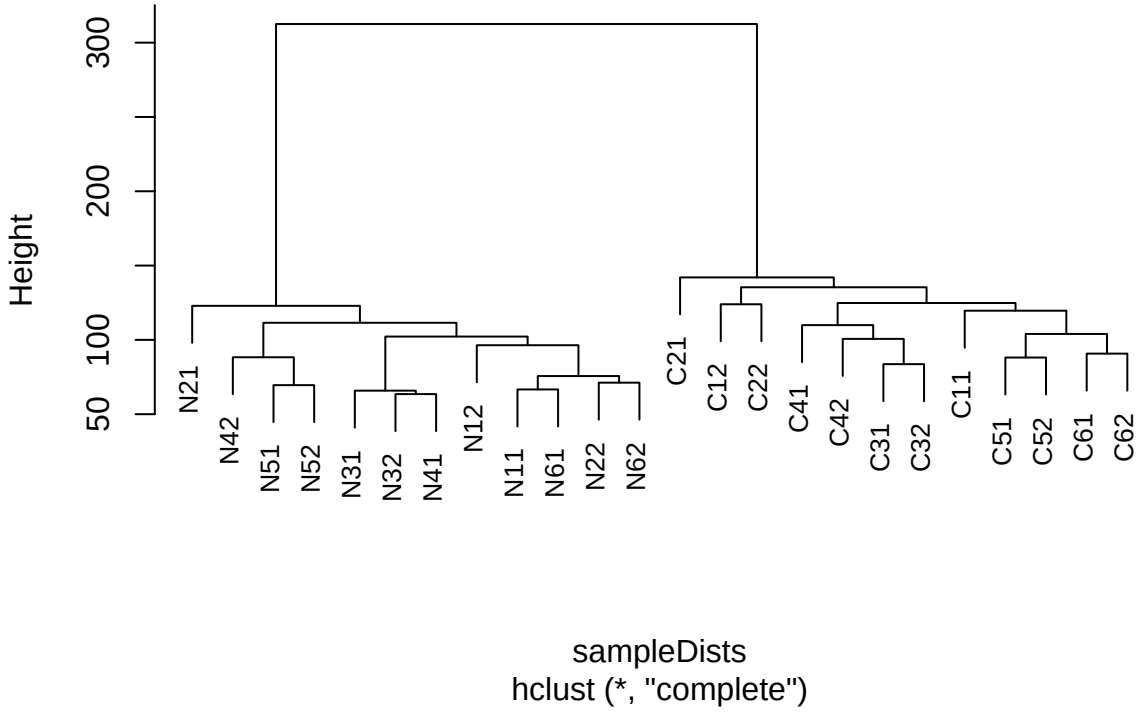
```
par(mfrow = c(1, 1))
fviz_dist(sampleDists)
```



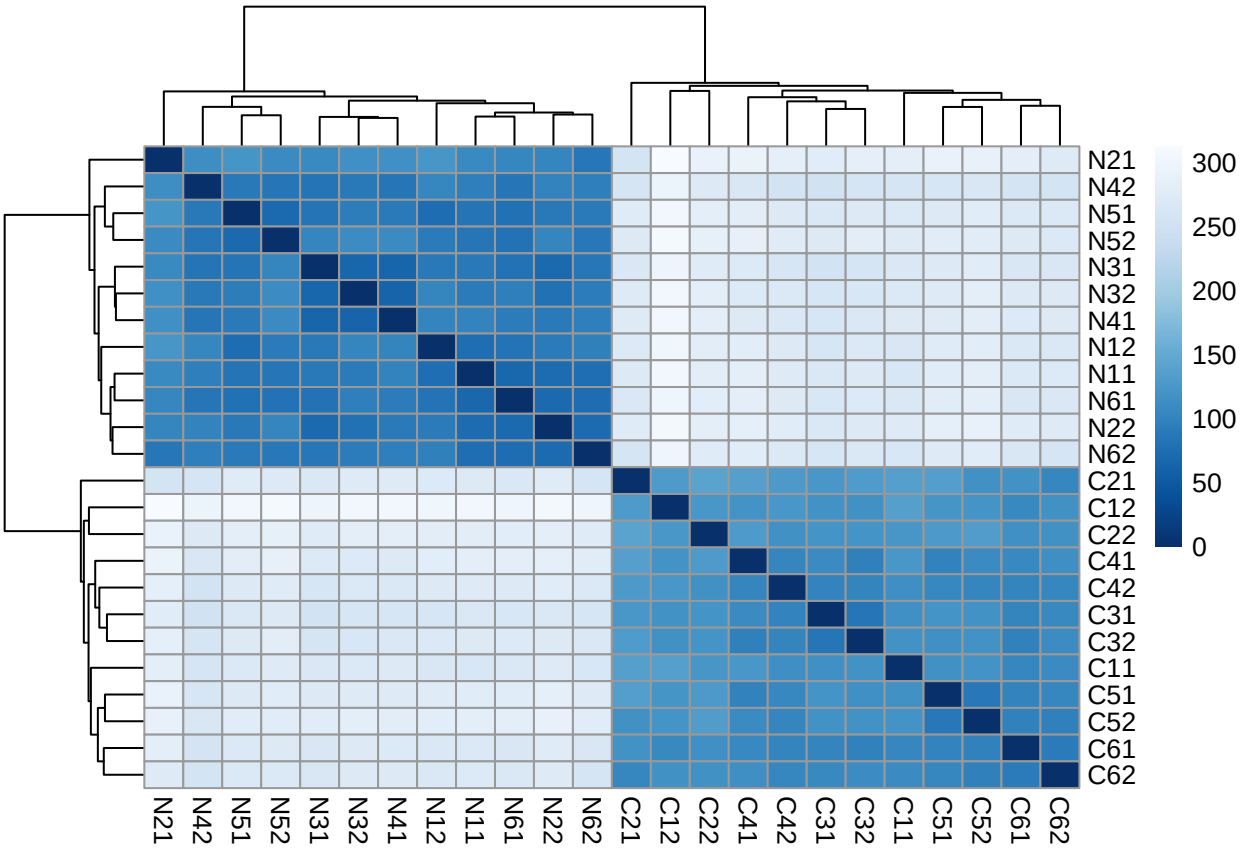
Agrupamiento jerárquico

Un agrupamiento jerárquico proporciona una representación alternativa, también basada en la matriz de distancias.

```
hc <- hclust(sampleDists)
plot(hc, labels = colnames(log2count_norm), main = "Agrupamiento jerárquico de las muestras",
    cex = 0.8) # Dendograma de distancias entre muestras
```



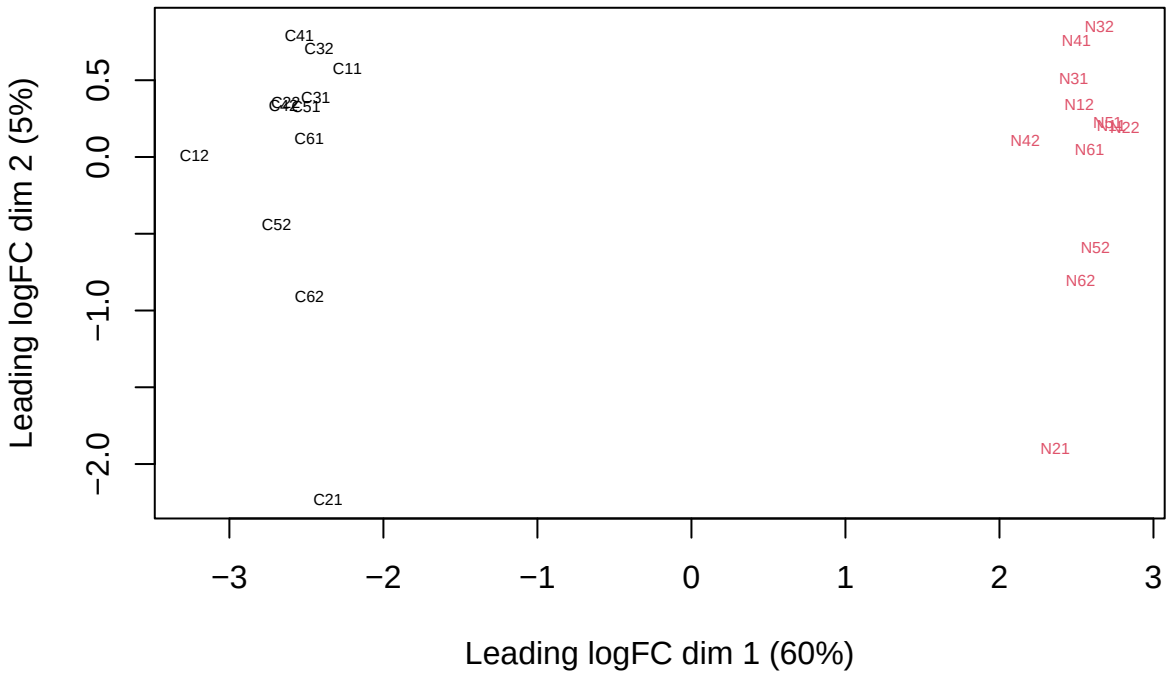

```
library("pheatmap")
library("RColorBrewer")
sampleDistMatrix <- as.matrix( sampleDists )
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```



Análisis de Escalamiento Multidimensional (MDS)

Reducción dimensional

```
plotMDS(y, col=as.numeric(group), labels=barcode, cex = 0.5)
```



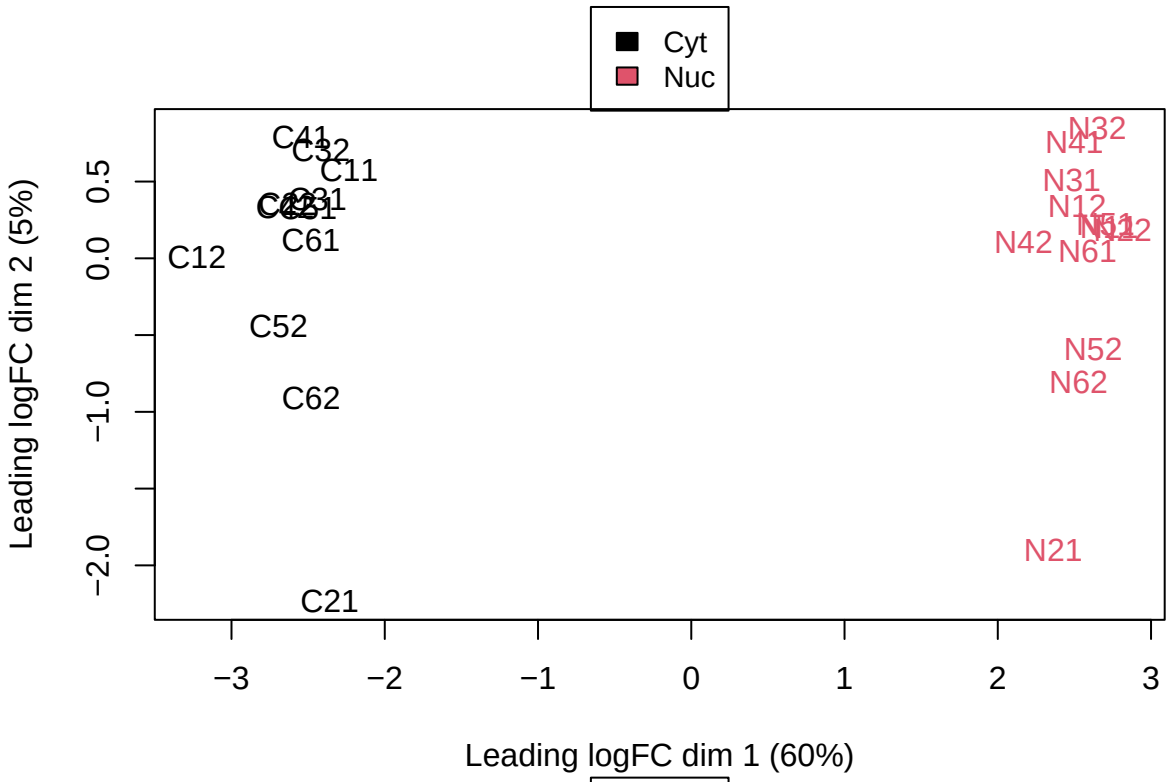

```
pdf(paste("plotMDS.pdf",sep=""))
plotMDS(y, col=as.numeric(group), labels=barcode, cex = 0.5)
dev.off()
```

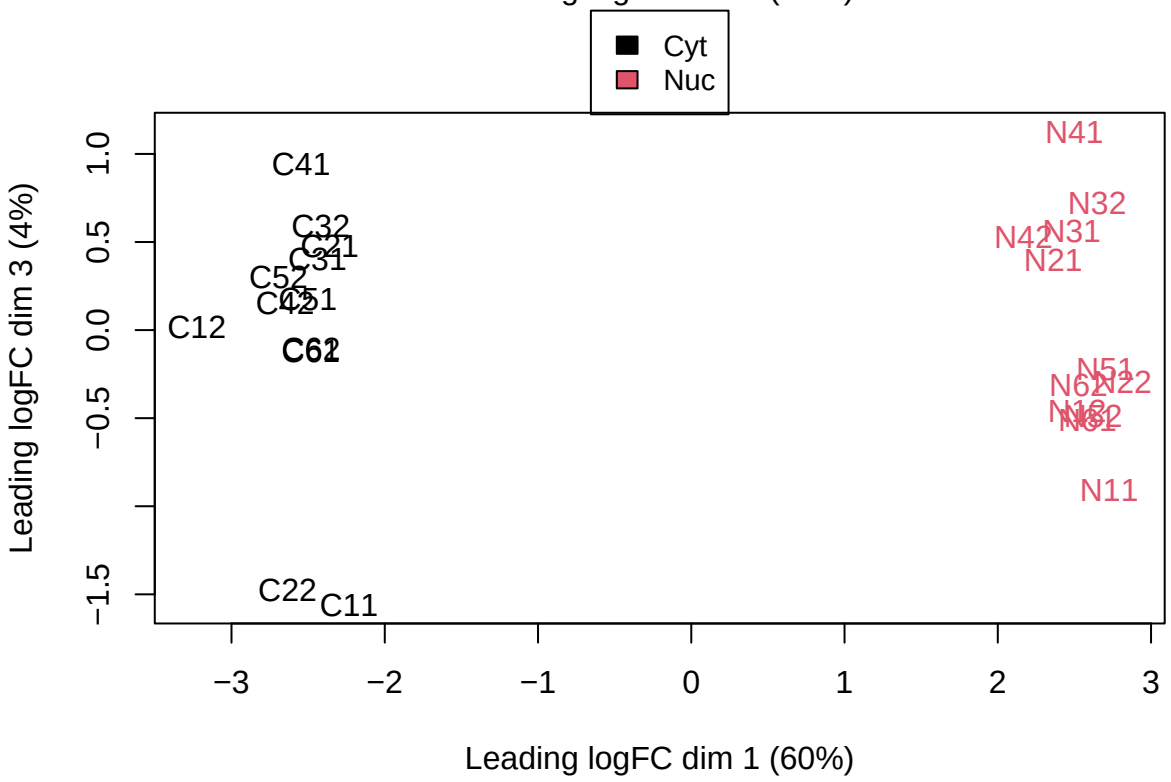
```
## pdf
##   2
```

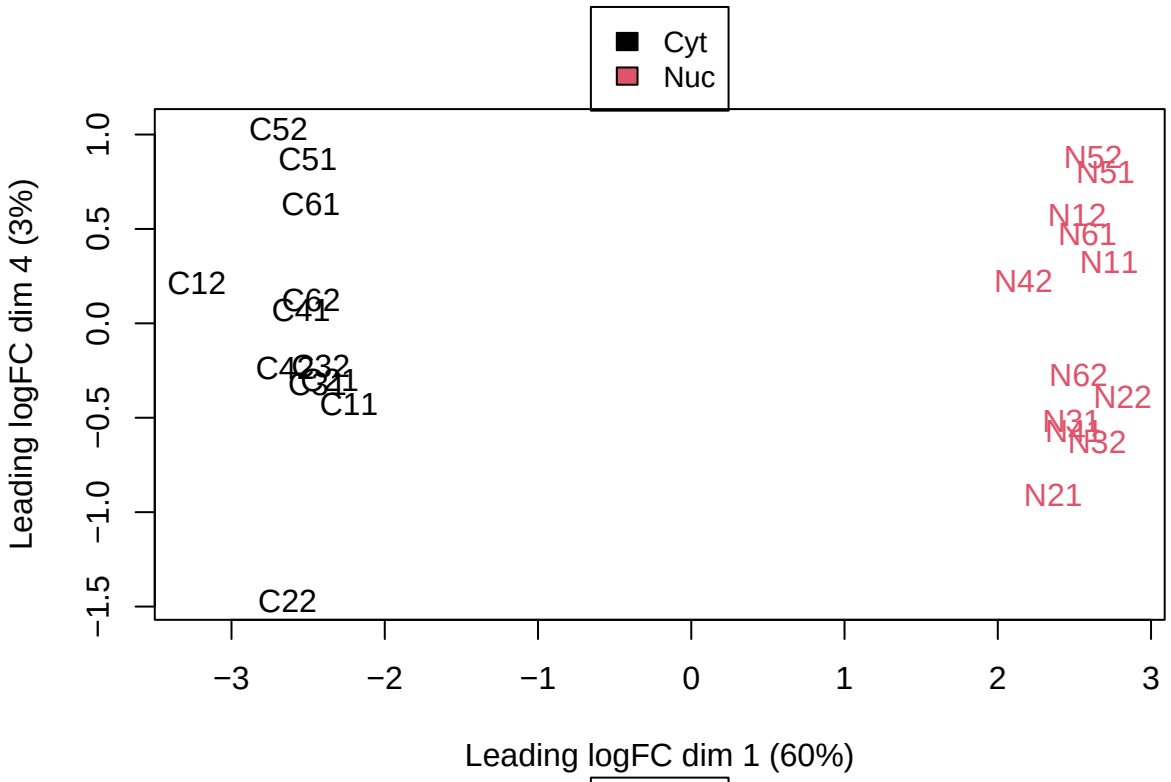
```
combi=combn(unique(c(1:8)), 2)
pdf("plotMDS_01_wo_outliers.pdf")
par(mfrow=c(2,3))
for (i in 1:ncol(combi)){
  plotMDS(y,dim.plot = combi[,i],col=as.numeric(group),pch=16, labels=colnames(y) )
  legend(x = "top",inset = c(0, -0.20 ), legend =levels(unique(group)), cex = 0.9 ,fill= as.numeric(unic
}
dev.off()
```

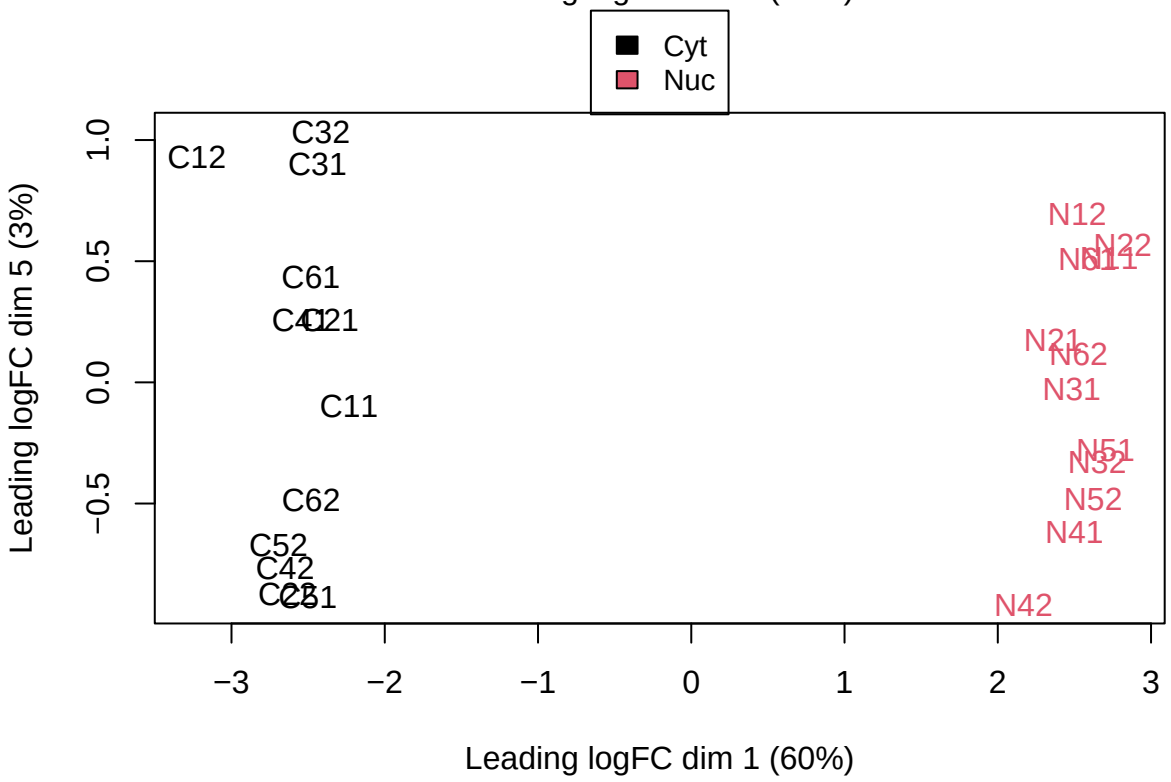
```
## pdf
##   2
```

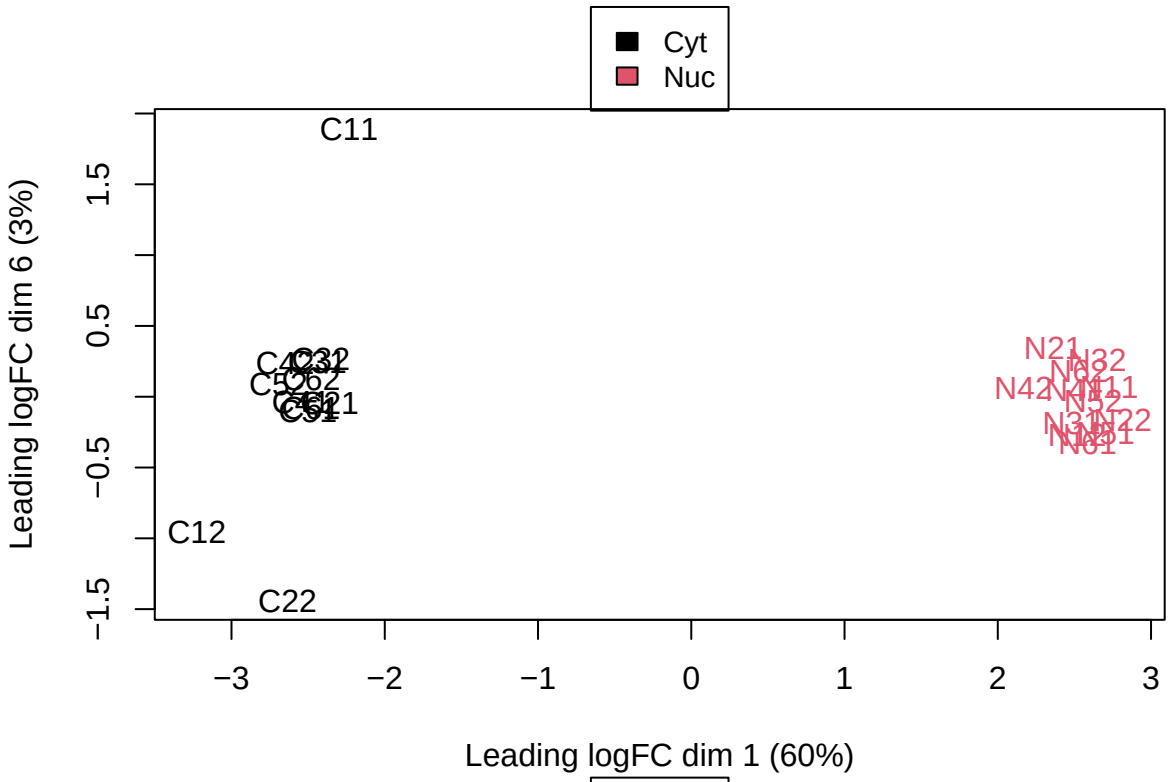
```
for (i in 1:ncol(combi)){
  plotMDS(y,dim.plot = combi[,i],col=as.numeric(group),pch=16, labels=colnames(y) )
  legend(x = "top",inset = c(0, -0.20 ), legend =levels(unique(group)), cex = 0.9 ,fill= as.numeric(unic
}
```

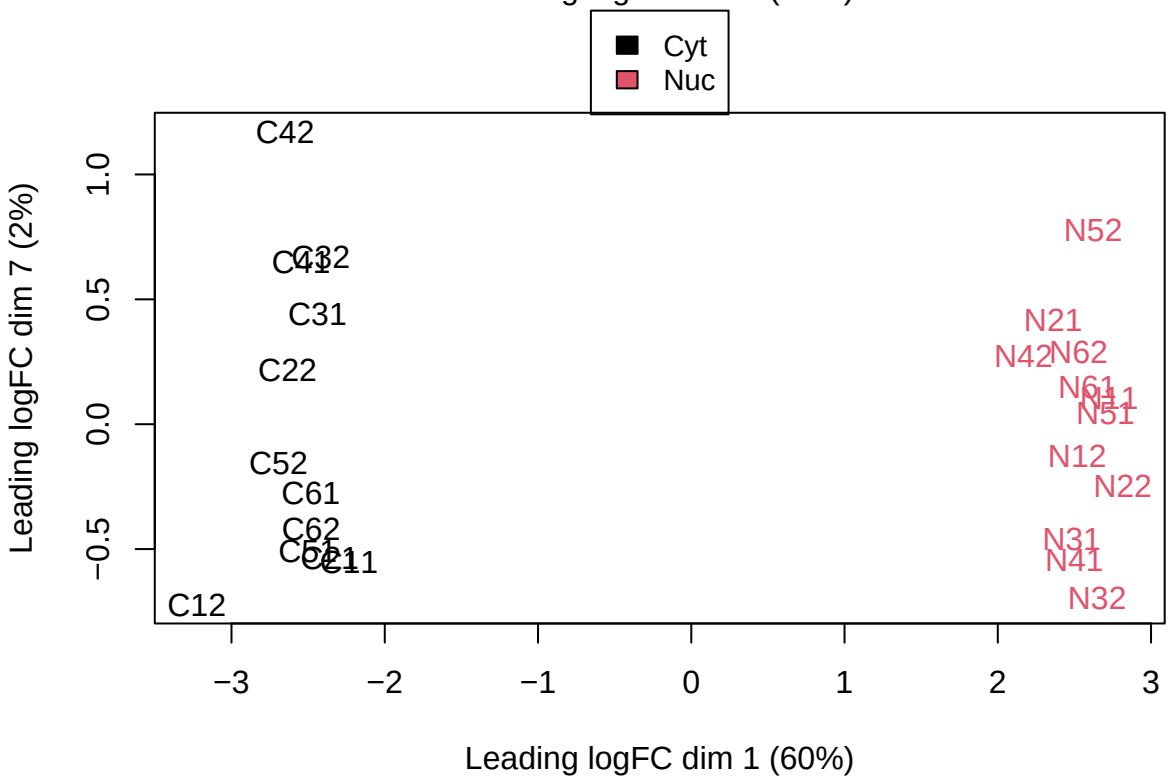

Cyt
Nuc
Leading logFC dim 2 (5%)
Leading logFC dim 1 (60%)
C41
C32
C11
C12
C61
C52
C62
C21
N32
N41
N31
N12
N42
N61
N52
N62
N21
0.5
0.0
−1.0
−2.0
−3
−2
−1
0
1
2
3

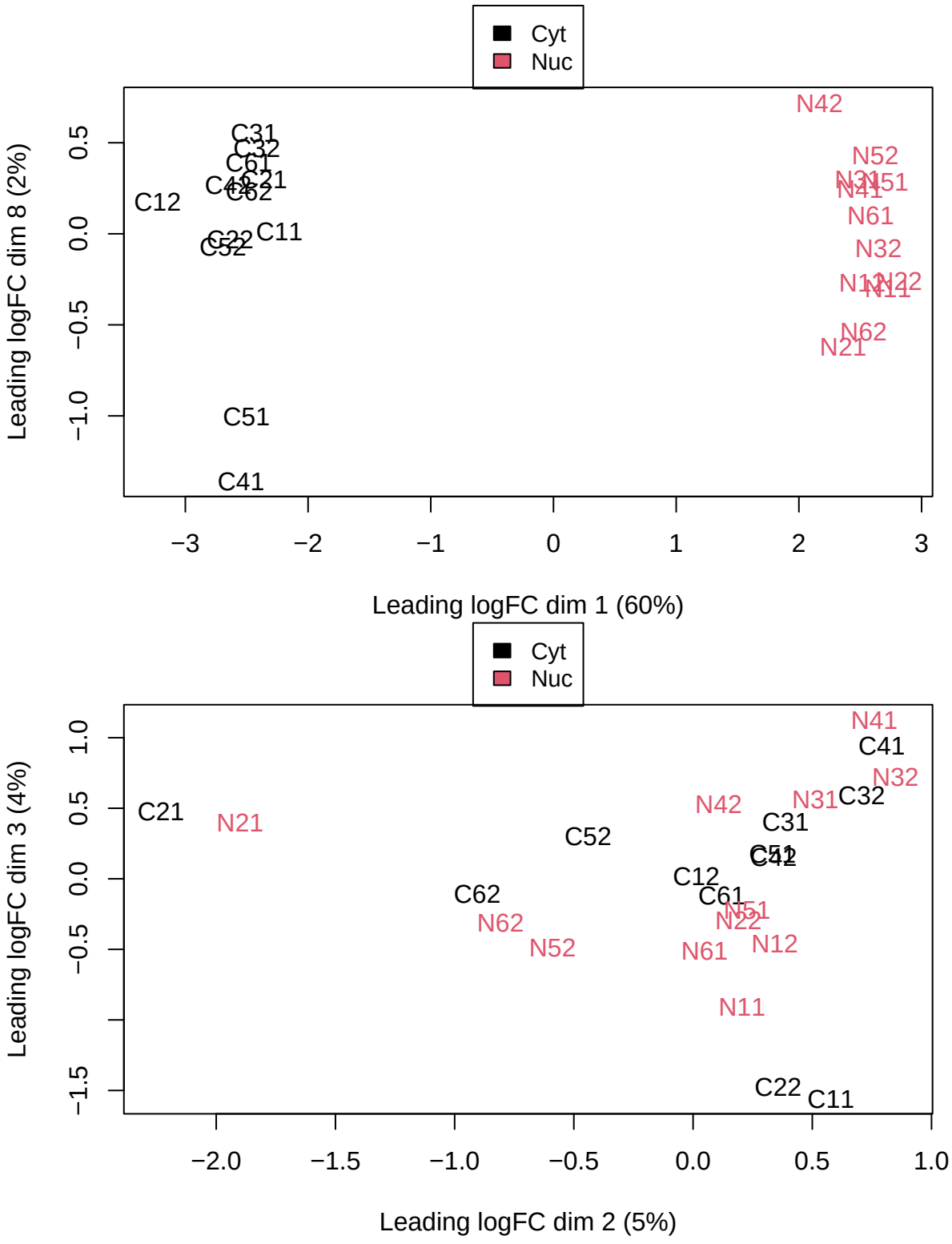

Cyt
Nuc
Leading logFC dim 3 (4%)
Leading logFC dim 1 (60%)
N41
C41
N32
C32
N42
N31
C31
N21
C52
C12
N51
N22
N11
C22
C11
1.0
0.5
0.0
−0.5
−1.5
−3
−2
−1
0
1
2
3

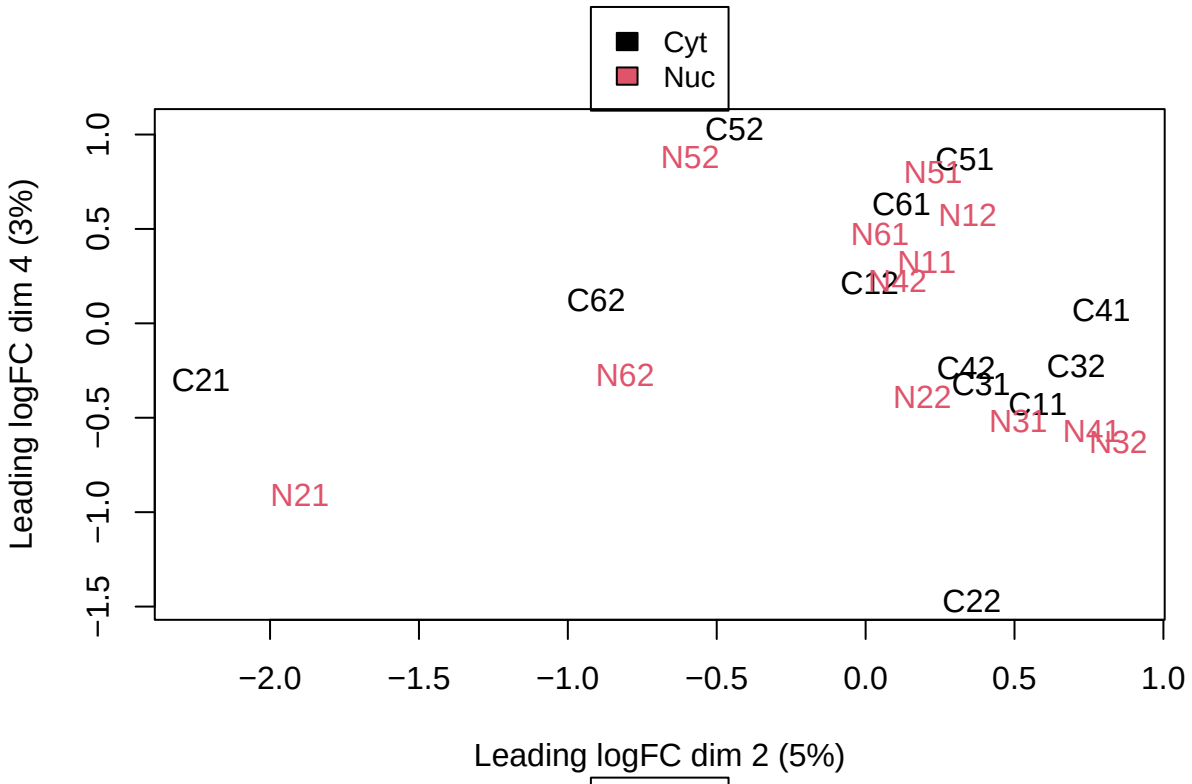

Cyt
Nuc
Leading logFC dim 4 (3%)
Leading logFC dim 1 (60%)
C52
C51
C61
C12
C62
C41
C42
C32
C31
C11
C22
N52
N51
N12
N61
N11
N42
N62
N22
N31
N41
N32
N21
1.0
0.5
0.0
−0.5
−1.0
−1.5
−3
−2
−1
0
1
2
3

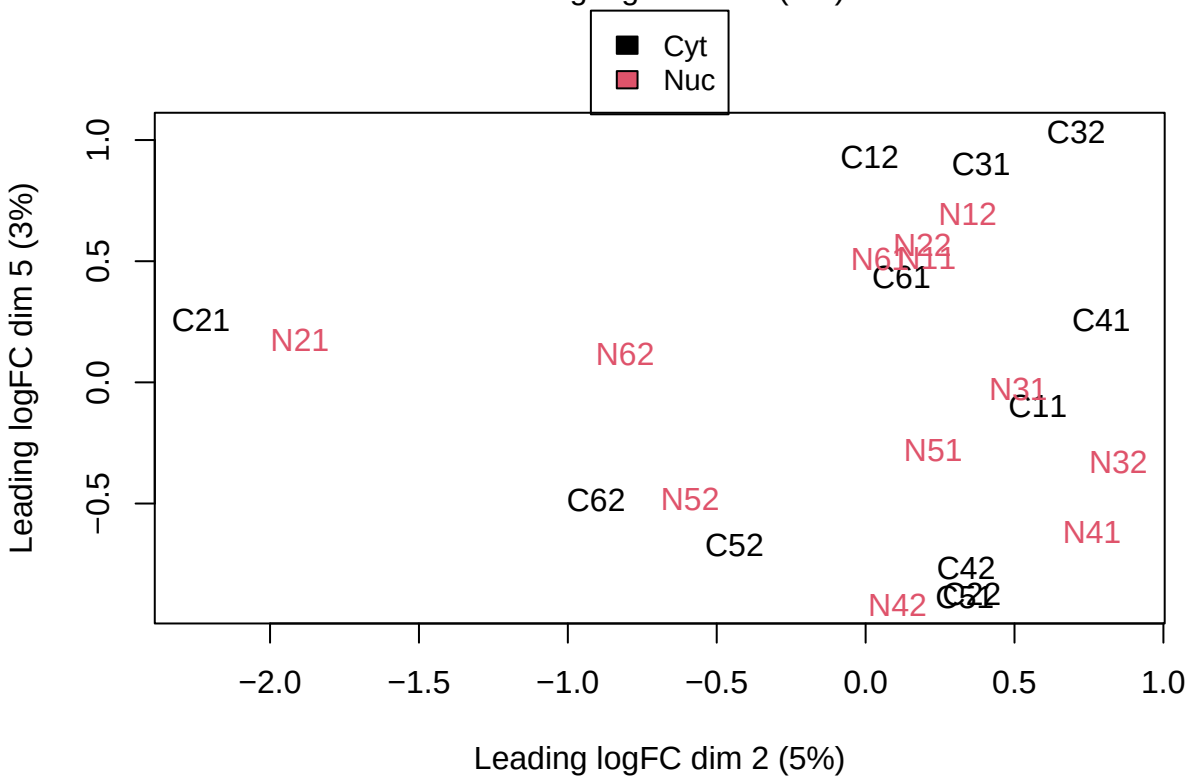

Cyt
Nuc
Leading logFC dim 5 (3%)
Leading logFC dim 1 (60%)
C32
C12
C31
C61
C41
C11
C62
C52
C42
C31
N12
N22
N51
N21
N62
N31
N51
N32
N52
N41
N42
1.0
0.5
0.0
−0.5
−3
−2
−1
0
1
2
3

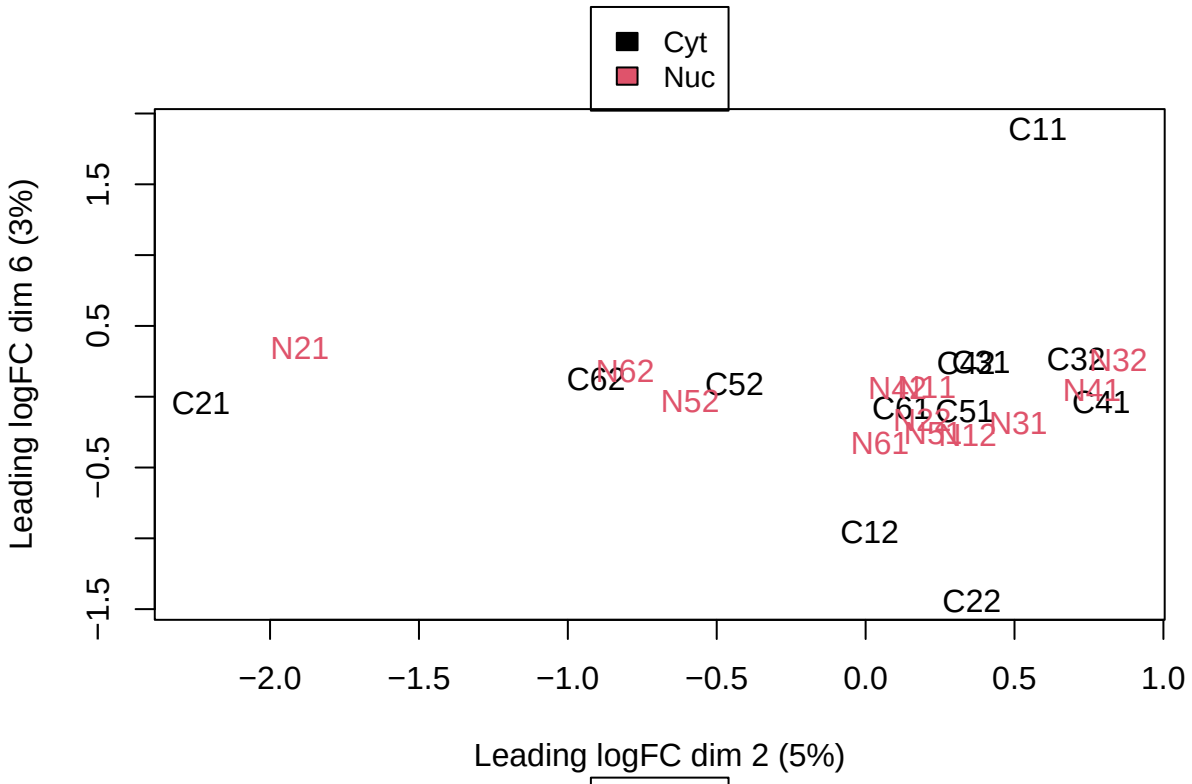

Cyt
Nuc
C11
C12
C22
N21
N42
N52
Leading logFC dim 6 (3%)
Leading logFC dim 1 (60%)
−3
−2
−1
0
1
2
3
1.5
0.5
−0.5
−1.5

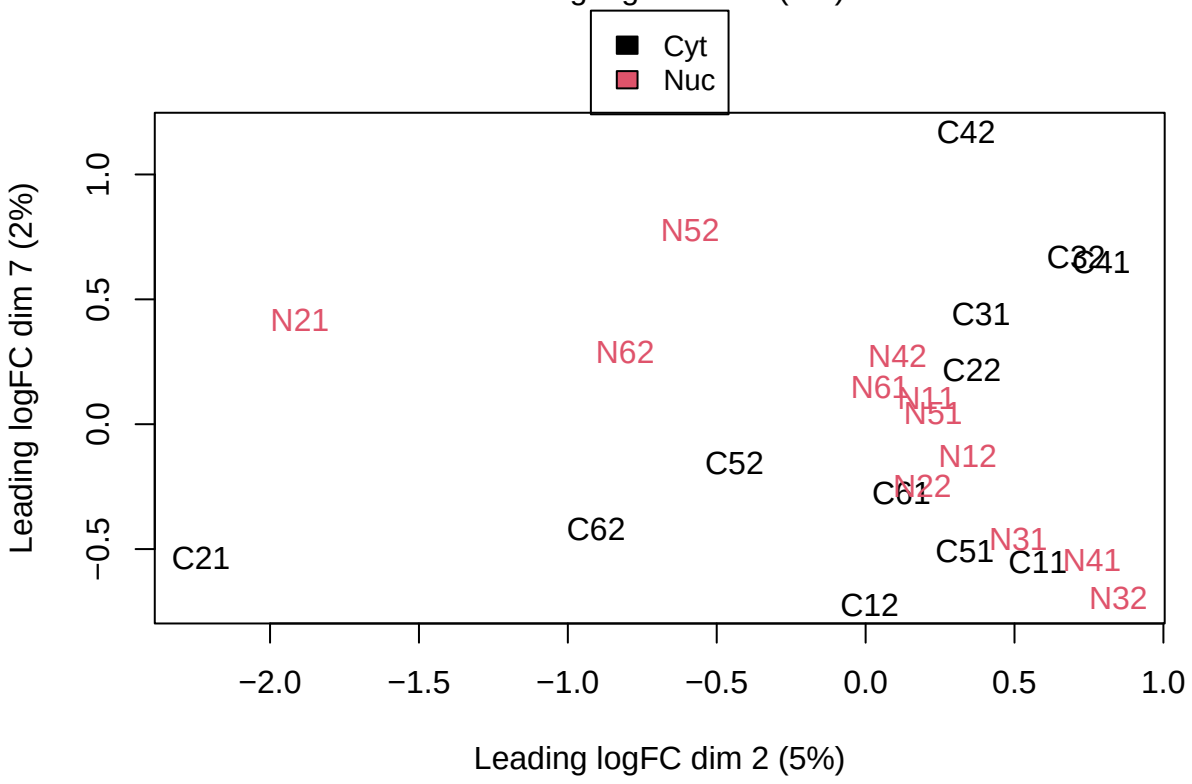

Cyt
Nuc
C42
C31
C22
C52
C61
C62
C12
N52
N21
N42
N62
N61
N51
N12
N22
N31
N41
N32
Leading logFC dim 7 (2%)
Leading logFC dim 1 (60%)
−3
−2
−1
0
1
2
3
1.0
0.5
0.0
−0.5

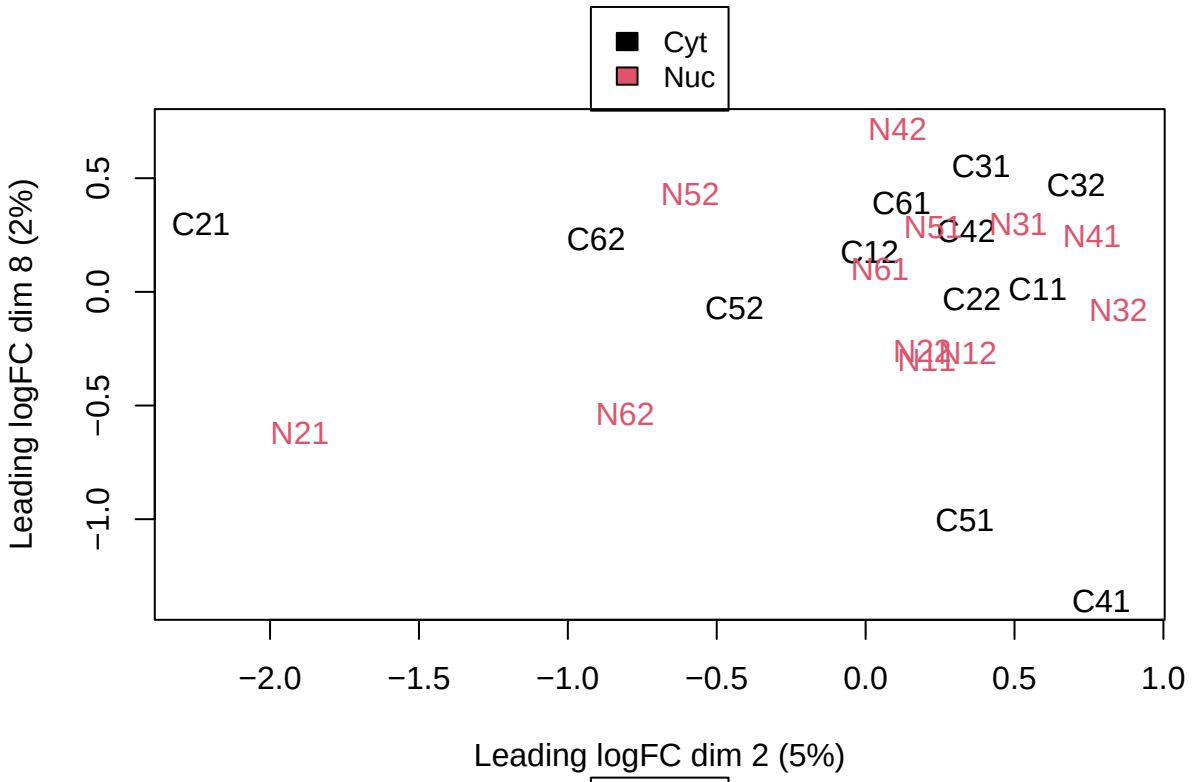

Cyt
Nuc
Leading logFC dim 8 (2%)
Leading logFC dim 1 (60%)
N42
C31
C32
C61
C21
C12
C62
C11
C22
C52
N52
N31
N51
N41
N61
N32
N12
N22
N11
N62
N21
C51
C41
Cyt
Nuc
Leading logFC dim 3 (4%)
Leading logFC dim 2 (5%)
N41
C41
N32
C21
N21
N42
N31
C32
C31
C52
C51
C42
C12
C62
C61
N62
N51
N22
N52
N12
N61
N11
C22
C11

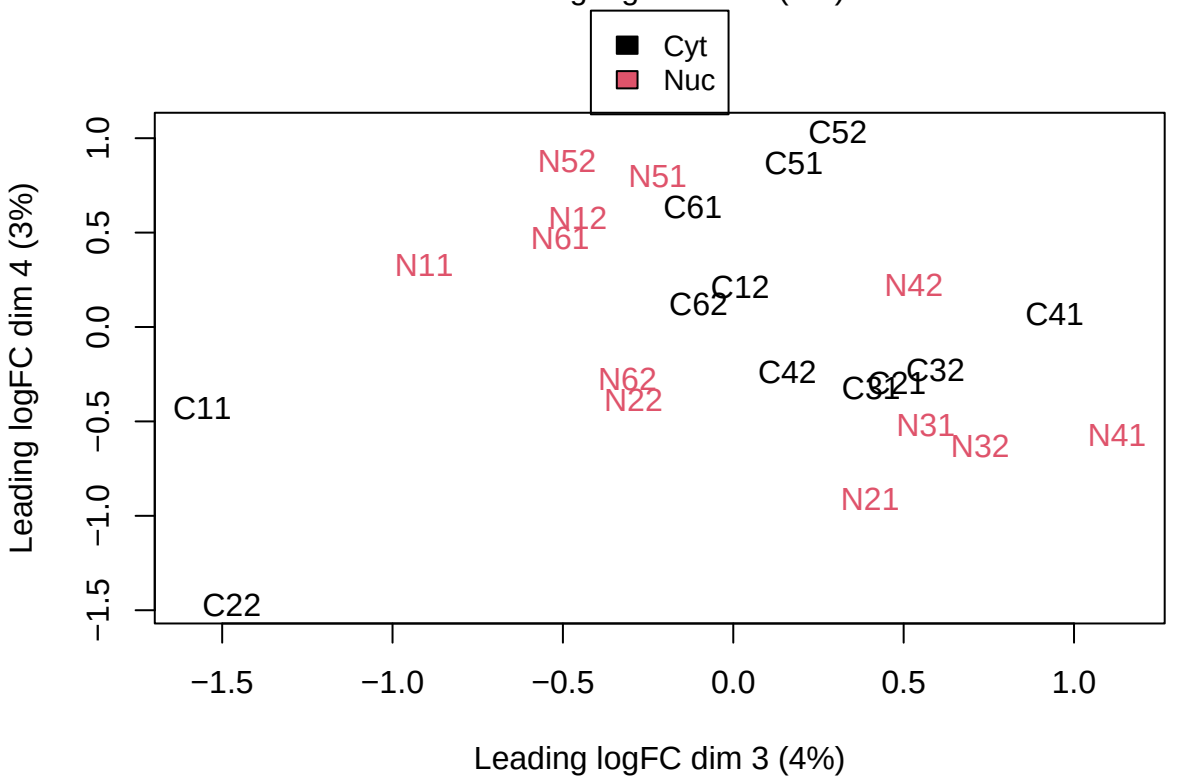

Cyt
Nuc
Leading logFC dim 4 (3%)
Leading logFC dim 2 (5%)
C52
N52
C51
N51
C61
N12
N61
N11
C12
N42
C62
C41
C21
N62
C42
C32
C31
N22
C11
N31
N41
N32
N21
C22
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

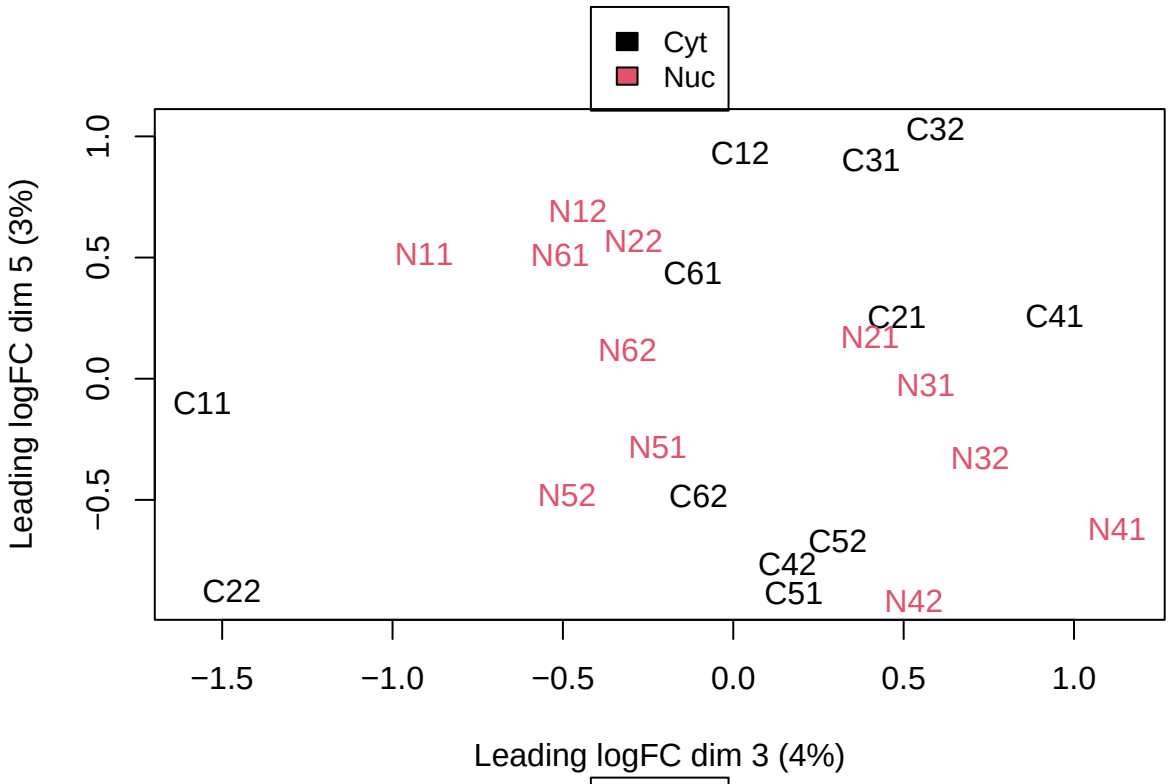

Cyt
Nuc
Leading logFC dim 5 (3%)
Leading logFC dim 2 (5%)
C32
C12
C31
N12
N22
N61
N11
C61
C21
C41
N21
N62
N31
C11
N51
N32
C62
N52
N41
C52
C42
N42
C22
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

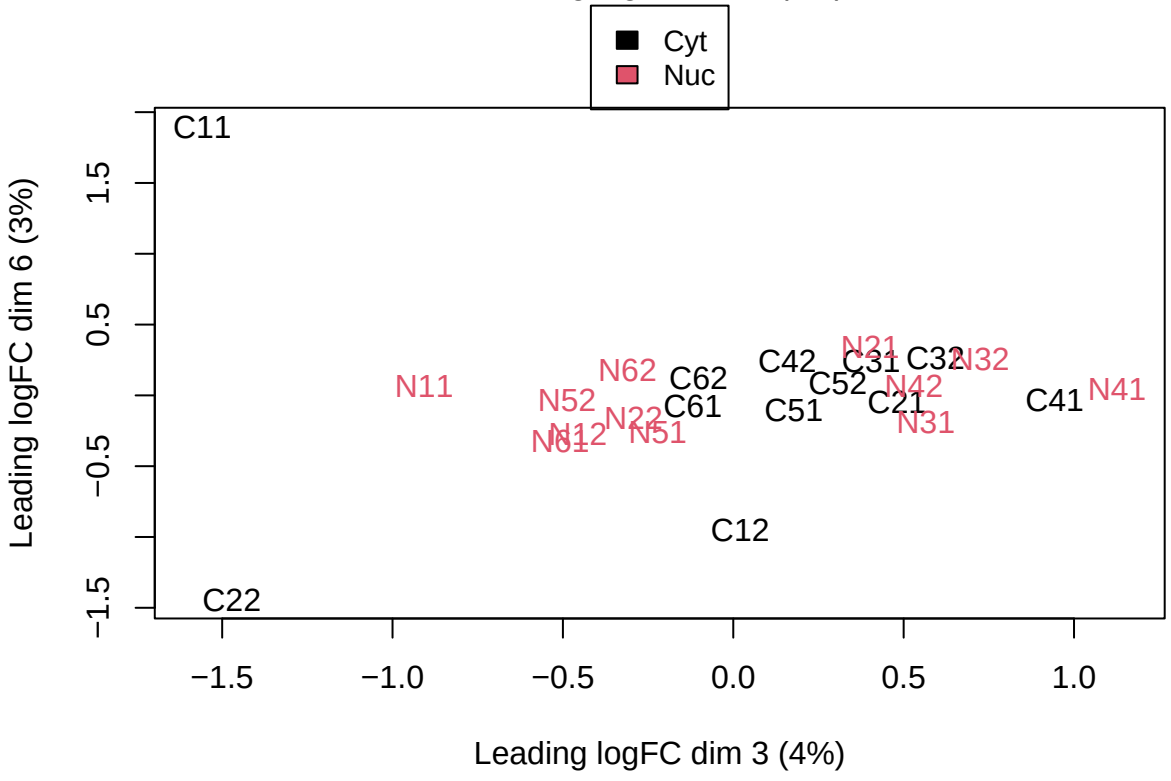

Cyt
Nuc
Leading logFC dim 6 (3%)
Leading logFC dim 2 (5%)
C11
N21
C21
C62
N62
N52
C52
C42
C32
N32
N42
N11
N41
C61
C51
C41
N61
N51
N12
N31
C12
C22
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
−1.5
−0.5
0.5

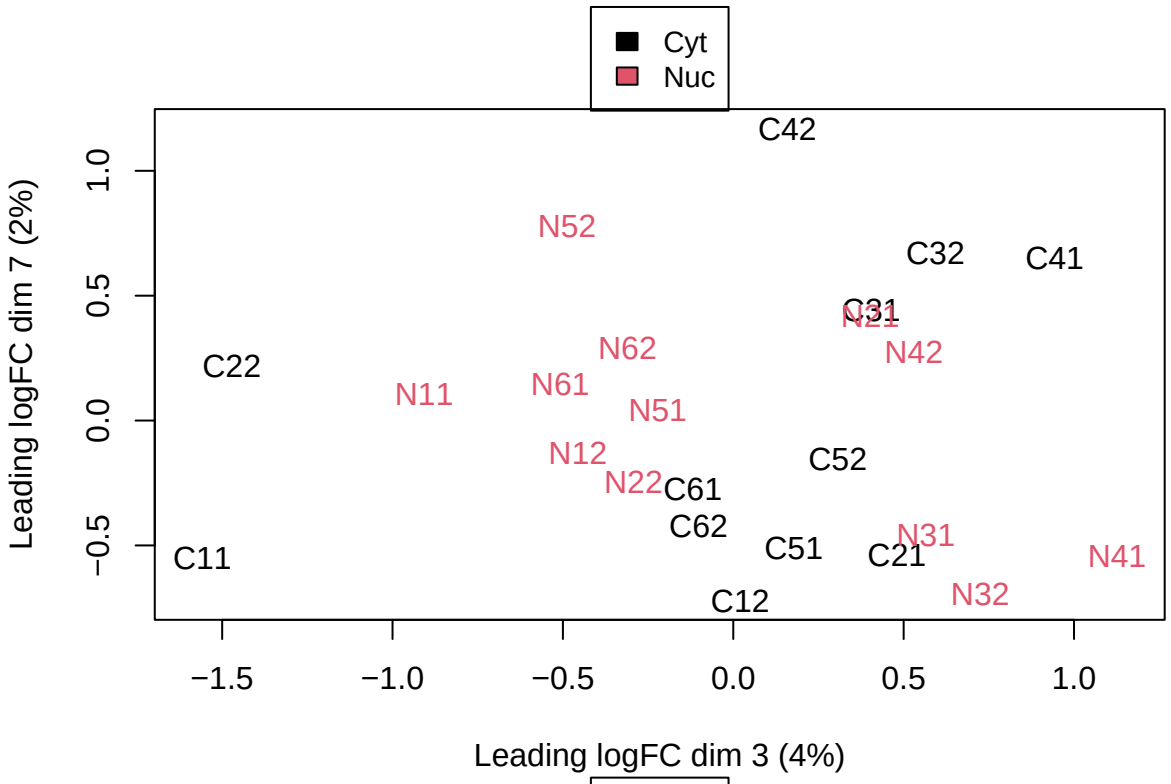

Cyt
Nuc
Leading logFC dim 7 (2%)
Leading logFC dim 2 (5%)
C42
N52
C32
C41
N21
C31
N62
N42
C22
N61
N11
N51
C52
N12
N22
C61
C62
N31
C51
C11
N41
C21
C12
N32
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
−0.5
0.0
0.5
1.0

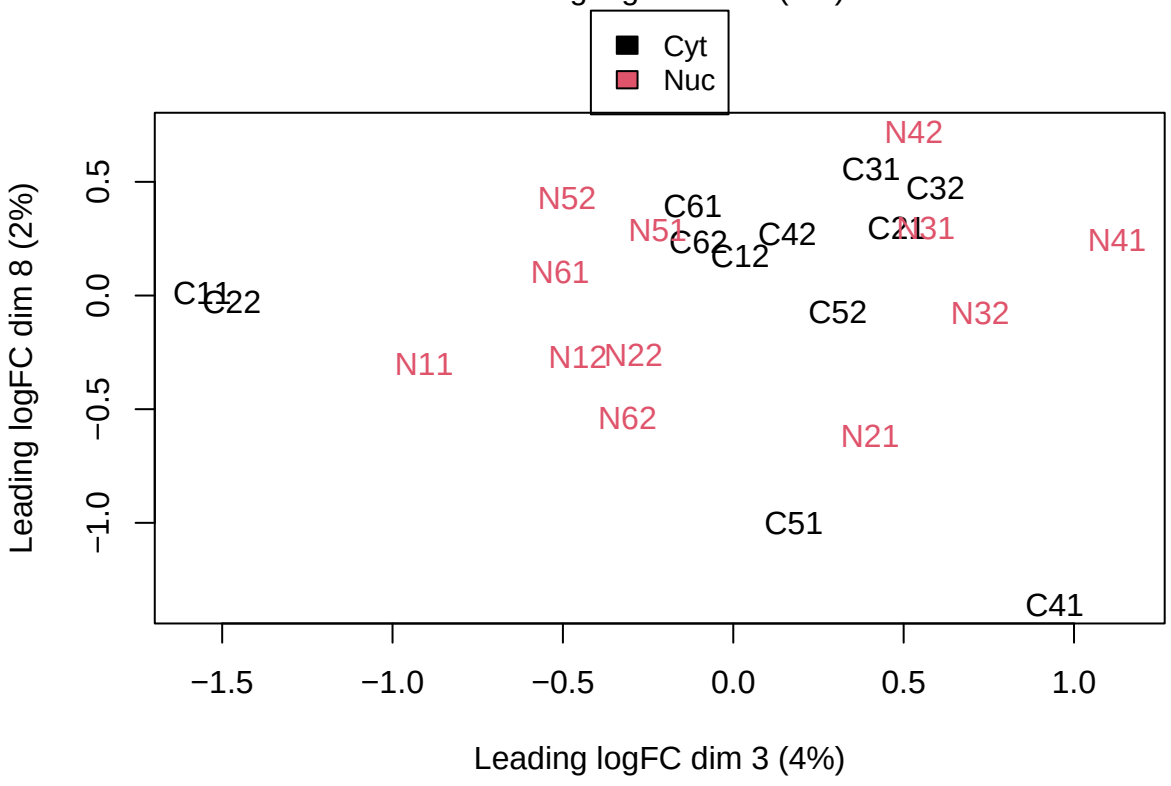

Cyt
Nuc
N42
C31
C32
N52
C61
C21
C62
N51
C42
N31
N41
C12
N61
C22
C11
C52
N32
N12
N62
N21
C51
C41
Leading logFC dim 8 (2%)
Leading logFC dim 2 (5%)
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
−1.0
−0.5
0.0
0.5

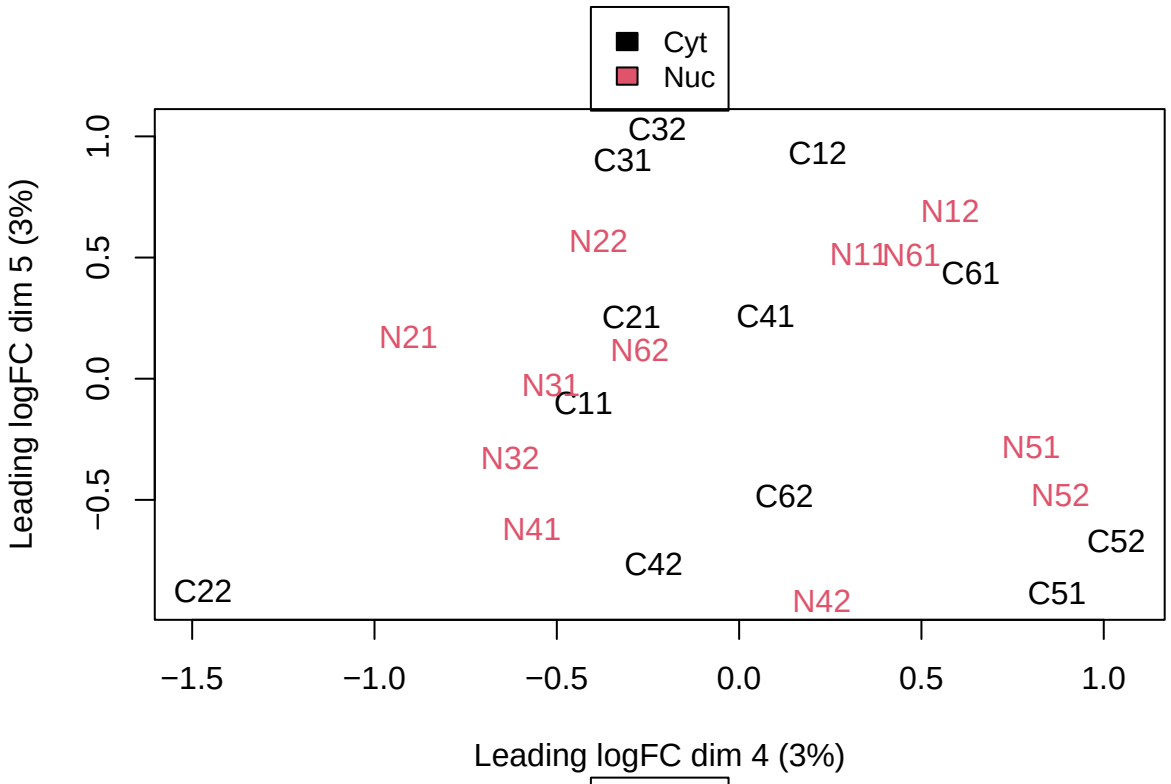

Cyt
Nuc
C52
N52
C51
N51
C61
N12
N61
N11
C12
N42
C62
C41
C42
C32
N62
N22
C11
N31
N32
N41
N21
C22
Leading logFC dim 4 (3%)
Leading logFC dim 3 (4%)
−1.5
−1.0
−0.5
0.0
0.5
1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0

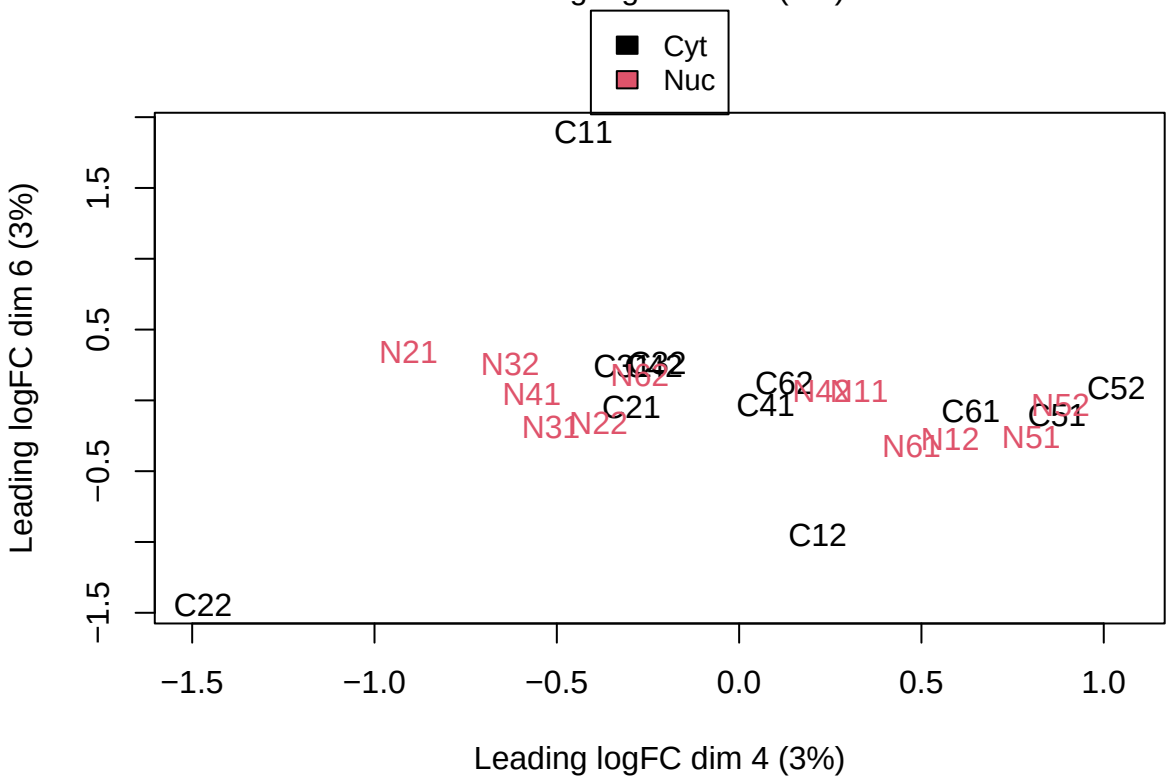

Cyt
Nuc
Leading logFC dim 5 (3%)
1.0
0.5
0.0
−0.5
C32
C12
C31
N12
N11
N61
N22
C61
C21
C41
N21
N62
N31
C11
N51
N32
N52
C62
N41
C52
C42
C51
C22
N42
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 3 (4%)

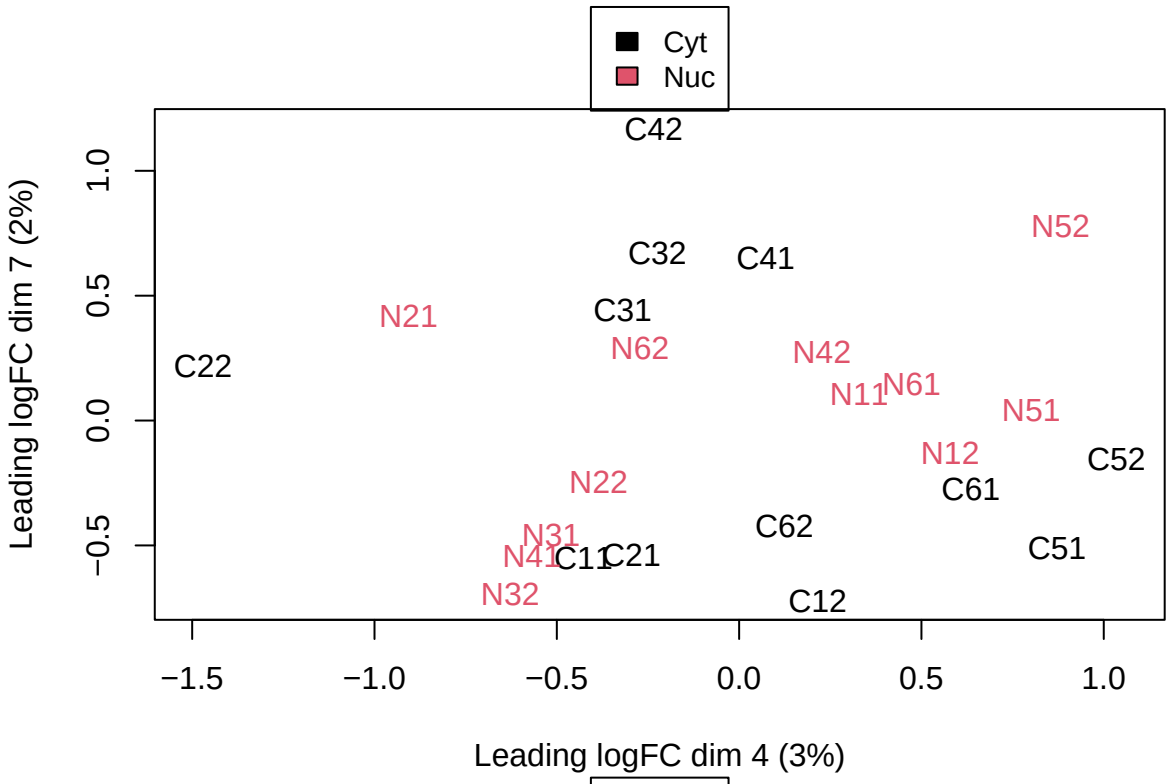

Cyt
Nuc
Leading logFC dim 6 (3%)
1.5
0.5
−0.5
−1.5
C11
N21
C42
C31
C32
N32
N62
C62
C52
N42
N11
N41
N52
C61
C51
C21
C41
N22
N31
N12
N51
N61
C12
C22
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 3 (4%)

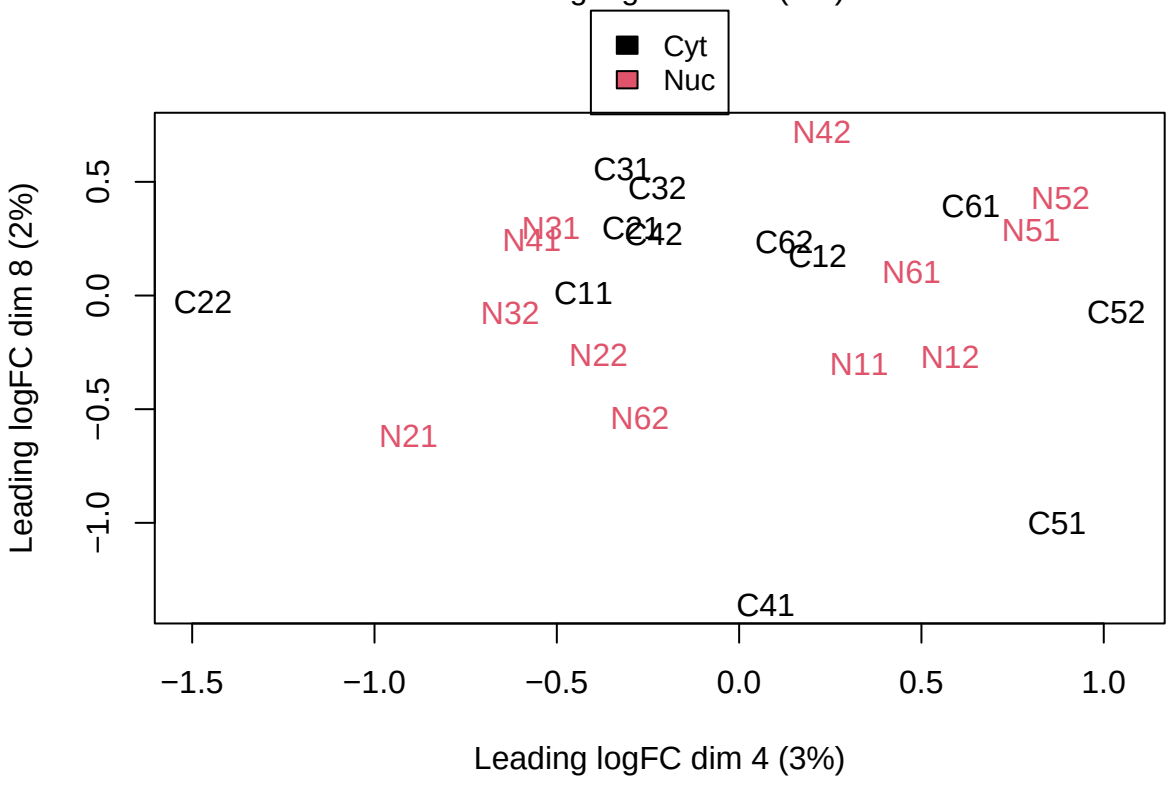

Cyt
Nuc
C42
N52
C32
C41
C31
N21
N62
N42
C22
N11
N61
N51
N12
C52
N22
C61
C62
N31
C11
C51
C21
N41
C12
N32
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 7 (2%)
Leading logFC dim 3 (4%)

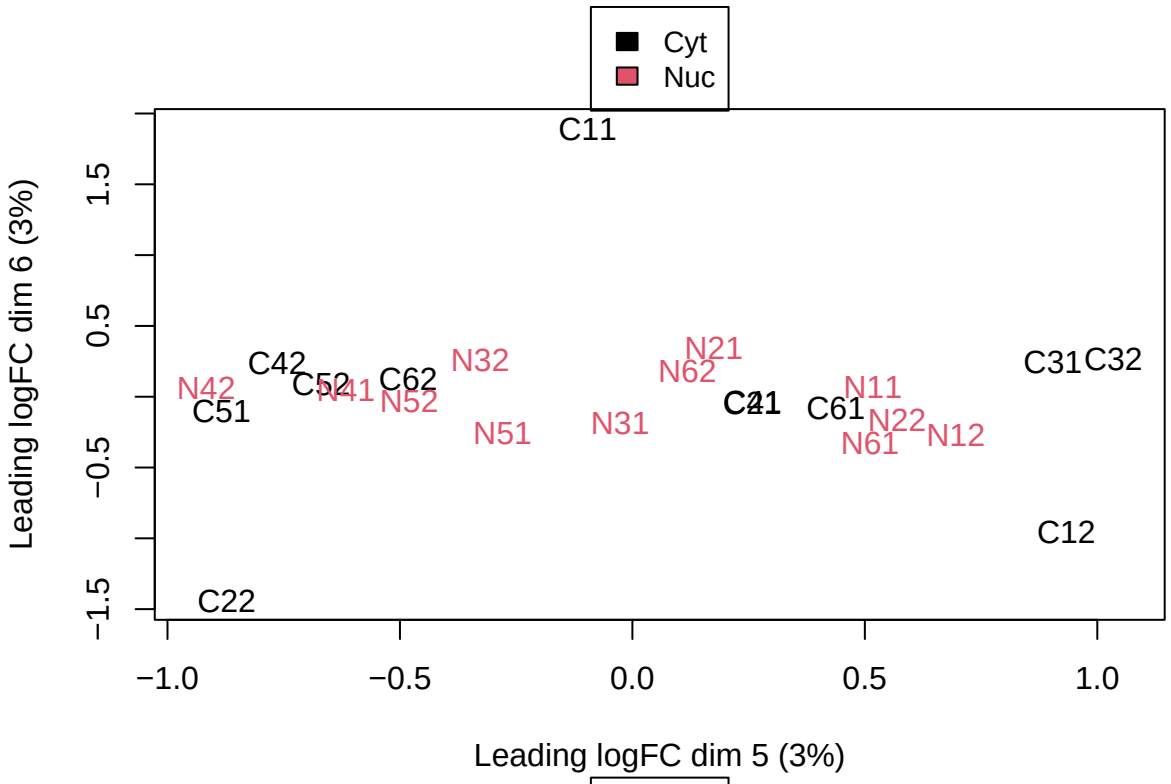

Cyt
Nuc
N42
C31
C32
N52
C61
N51
C62
C42
C21
N31
N41
C12
N61
C11
C22
C52
N32
N11
N12
N22
N62
N21
C51
C41
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 8 (2%)
Leading logFC dim 3 (4%)

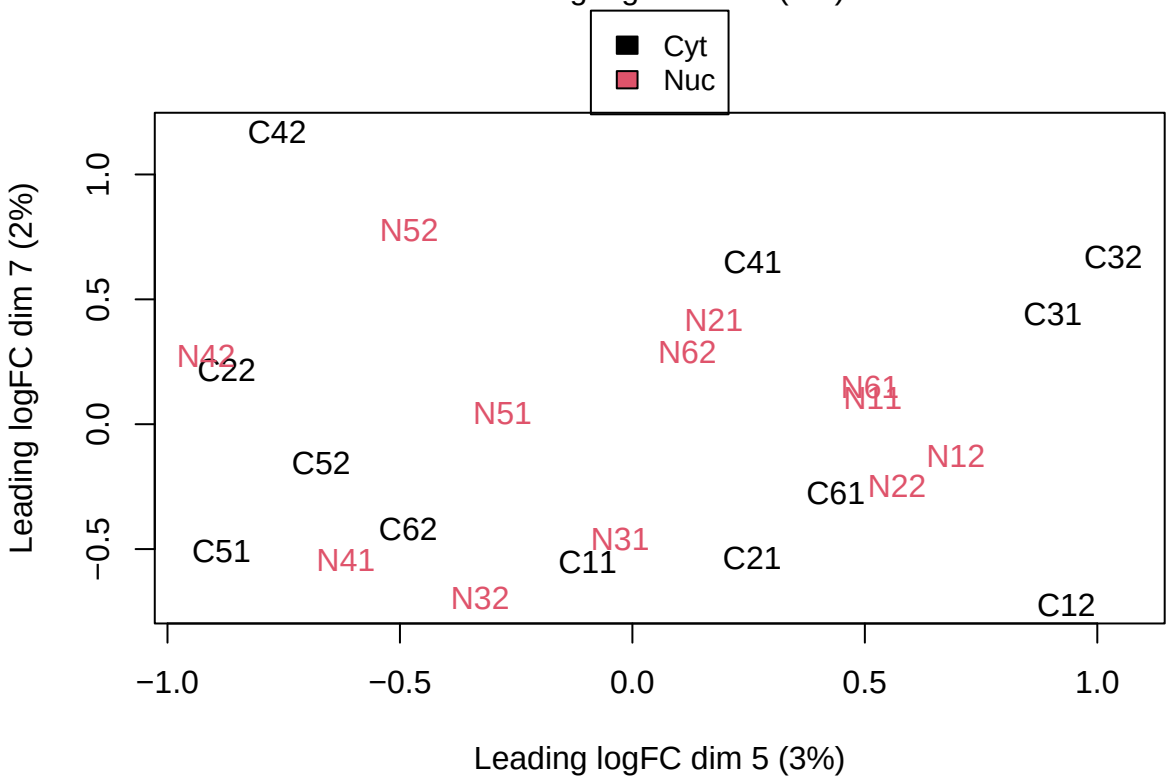

Cyt
Nuc
C32
C31
C12
N12
N22
N11
N61
C61
C21
C41
N21
N62
N31
C11
N51
N32
C62
N52
N41
C52
C42
C22
N42
C51
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 5 (3%)
Leading logFC dim 4 (3%)

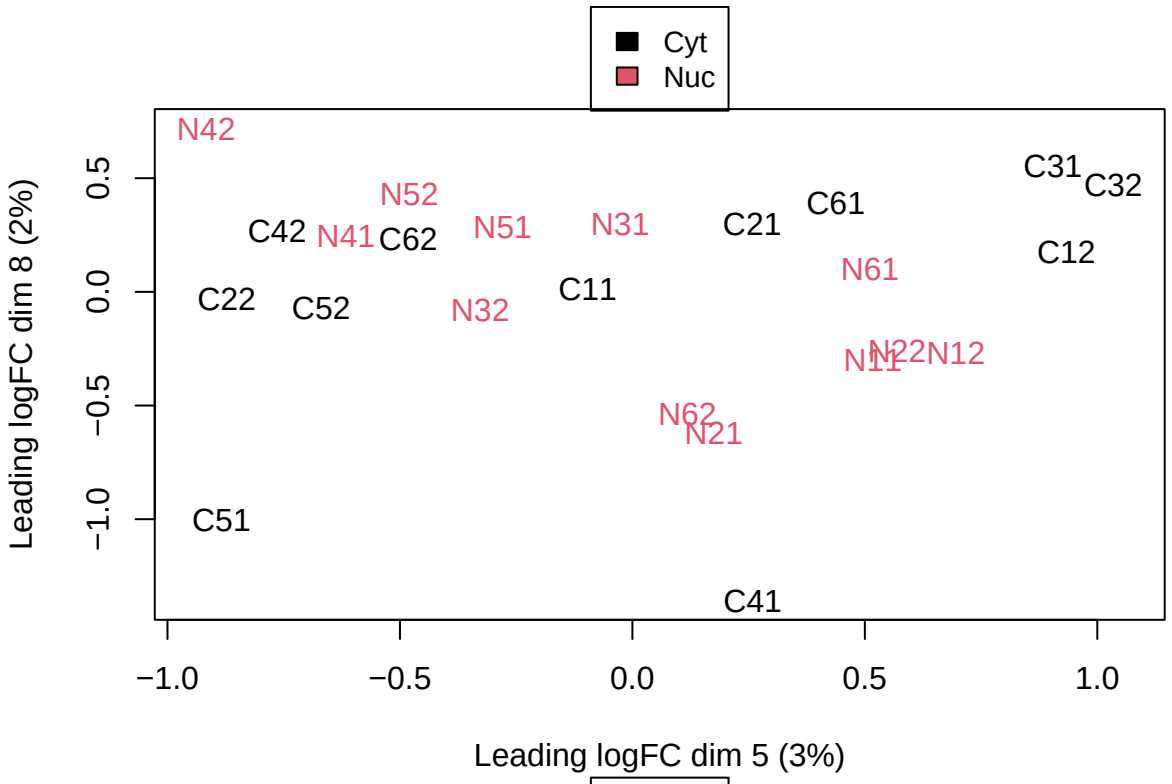

Cyt
Nuc
C11
N21
N32
N41
N31
N22
C21
C41
C62
C52
C61
C51
N12
N51
C12
C22
1.5
0.5
−0.5
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 6 (3%)
Leading logFC dim 4 (3%)

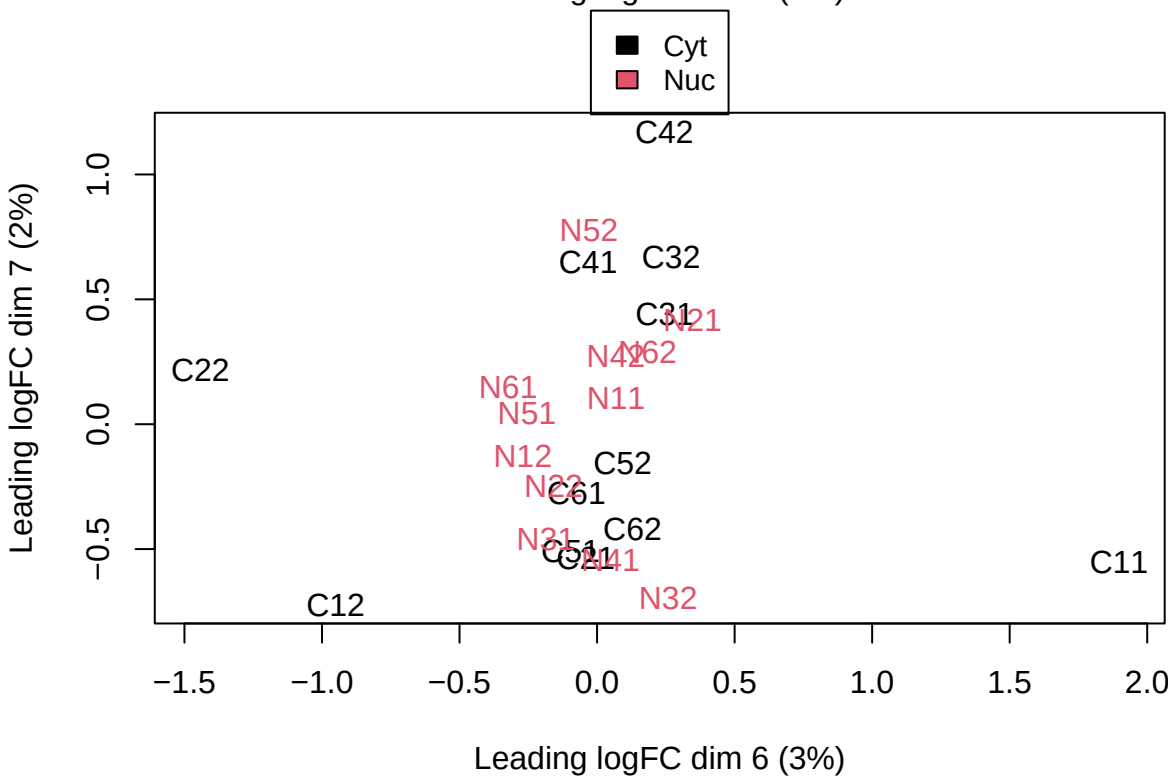

Cyt
Nuc
C42
N52
C32
C41
N21
C31
N62
N42
C22
N11
N61
N51
N12
C52
N22
C61
C62
N31
N41
C11
C21
C51
N32
C12
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 7 (2%)
Leading logFC dim 4 (3%)

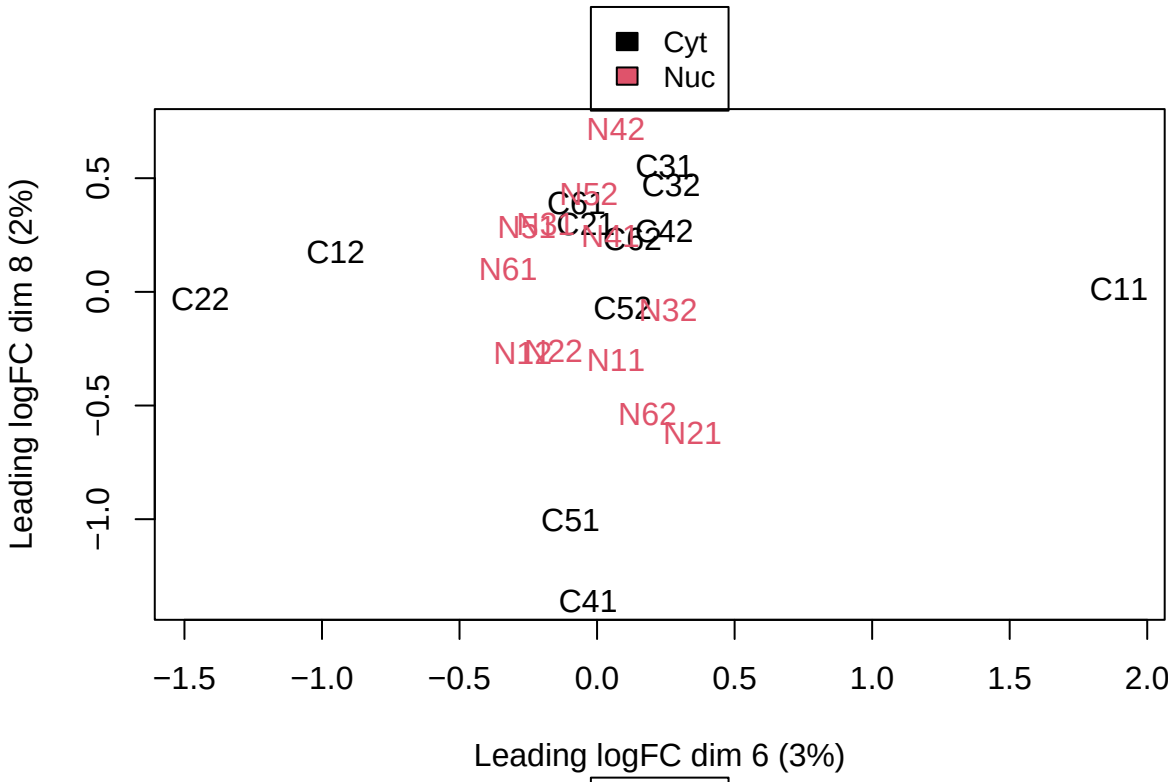

Cyt
Nuc
N42
C31
C32
C61
N52
N31
C21
N41
C42
C62
N51
C12
N61
C22
C11
C52
N32
N22
N11
N12
N62
N21
C51
C41
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
Leading logFC dim 8 (2%)
Leading logFC dim 4 (3%)

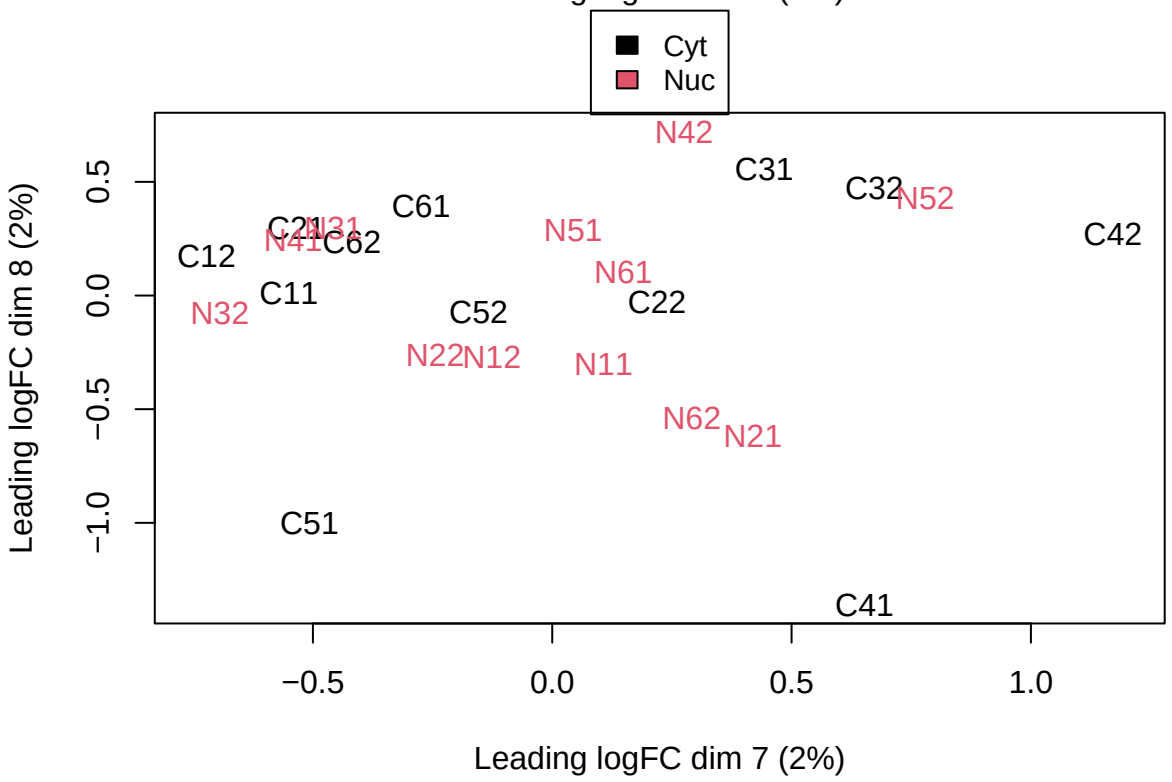

Cyt
Nuc
C11
C42
N42
C52
N41
C62
N32
N52
C51
N51
N21
N62
N31
C21
C61
N11
N22
N61
N12
C31
C32
C12
C22
Leading logFC dim 6 (3%)
Leading logFC dim 5 (3%)
1.5
0.5
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0


Cyt
Nuc
C42
N52
C41
C32
C31
N21
N62
N42
C22
N61
N11
N51
N12
C52
C61
N22
C62
N31
C51
N41
C11
C21
N32
C12
Leading logFC dim 7 (2%)
Leading logFC dim 5 (3%)
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
1.0


Cyt
Nuc
N42
N52
C42
N41
C62
N51
N31
C21
C61
C31
C32
C12
N61
C22
C52
N32
C11
N11
N22
N12
N62
N21
C51
C41
Leading logFC dim 8 (2%)
Leading logFC dim 5 (3%)
−1.0
−0.5
0.0
0.5
1.0

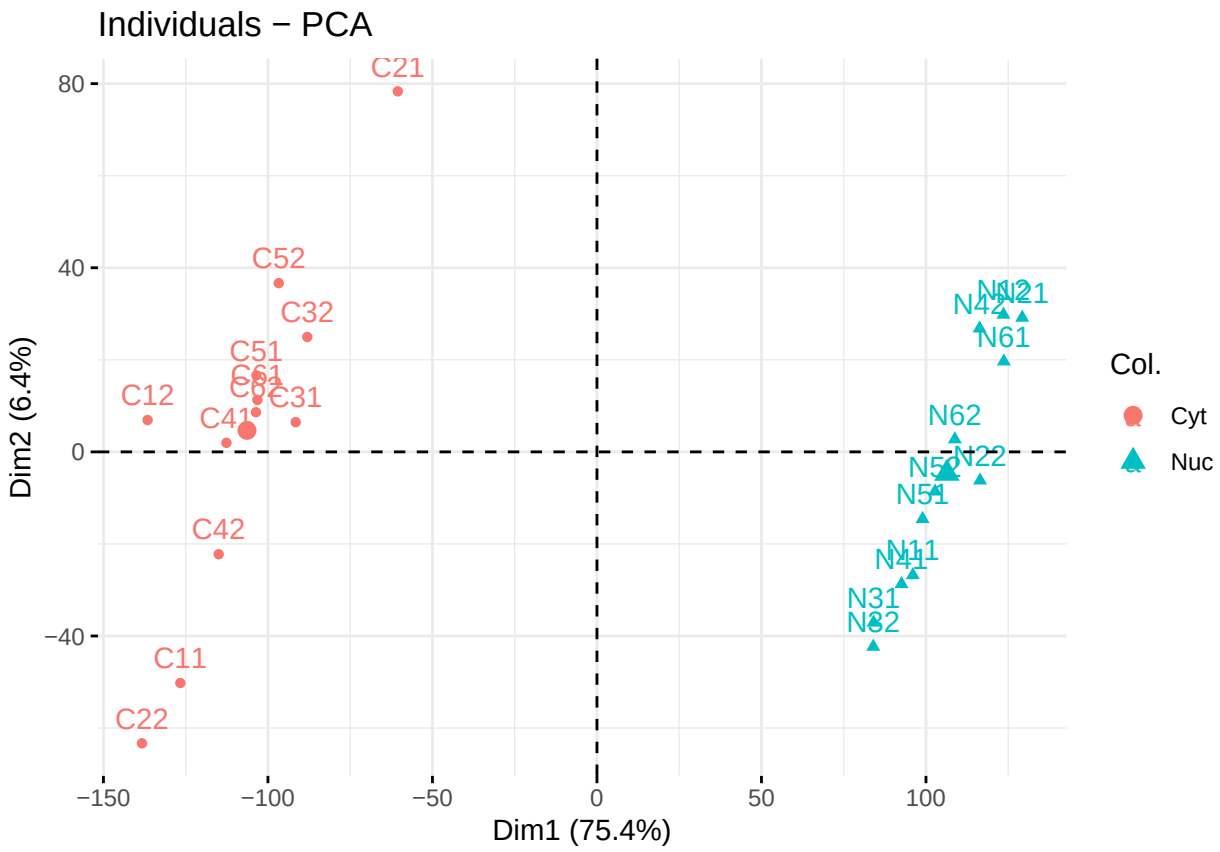

Cyt
Nuc
C42
N52
C41
C32
C31
N21
N62
N42
C22
N61
N11
N51
N12
C52
N22
C61
C62
N31
C51
C41
C11
C12
N32
Leading logFC dim 7 (2%)
Leading logFC dim 6 (3%)
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Cyt
Nuc
N42
C31
C32
C61
C21
C42
C62
C12
N61
C22
C11
C52
N32
N11
N62
N21
C51
C41
Leading logFC dim 8 (2%)
Leading logFC dim 6 (3%)


Cyt
Nuc
N42
C31
C32
N52
C61
N51
C42
C12
C62
N61
C11
C22
N32
C52
N22
N12
N11
N62
N21
C51
C41
Leading logFC dim 8 (2%)
Leading logFC dim 7 (2%)

PCA

```
library(FactoMineR)

pca.raw.y <- log2(y$counts+1)

pca.y <- PCA(t(pca.raw.y),graph = F)
fviz_pca_ind(pca.y, col.ind = group)
```




```
pdf("PCA.pdf")
fviz_pca_ind(pca.y, col.ind = group)
dev.off()
```

```
## pdf
##   2
```

Análisis de expresión diferencial (DE)

El objetivo del análisis de expresión diferencial es seleccionar genes cuya expresión difiere entre grupos.

Selección de genes usando limma-Voom

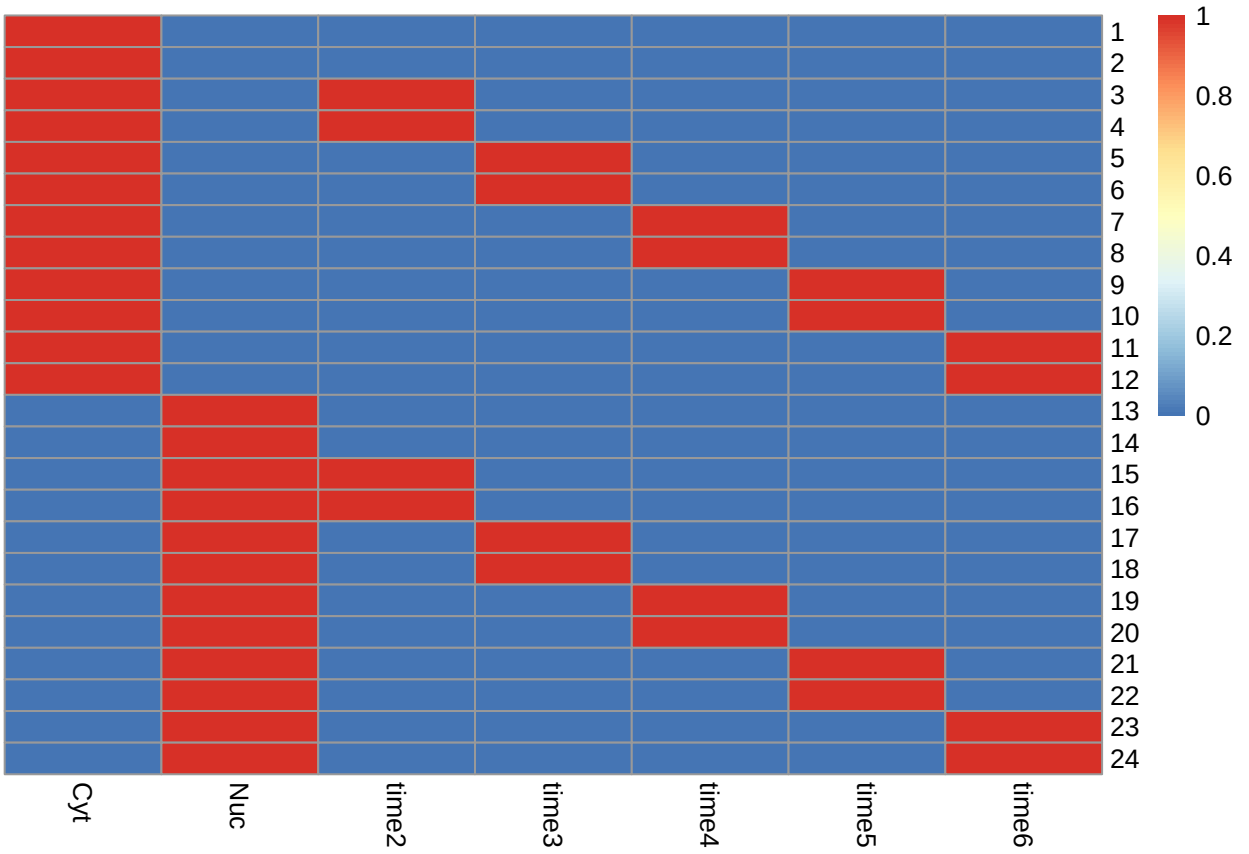
La ventaja principal de esta aproximación es que permite trabajar con toda la flexibilidad de los modelos lineales para representar diseños experimentales, y, en muchos casos , aprovechar la experiencia previa del

usuario en el manejo de limma.

Matriz de diseño

Utilizando la variable group podemos definir una matriz de diseño y, sobre ésta, los contrastes que nos interesan.

```
mod <- model.matrix(~0+group+time)
colnames(mod)=gsub("group","",colnames(mod))
pheatmap(mod,cluster_rows = FALSE,cluster_cols = FALSE)
```




```
mod
```

```
##    Cyt Nuc time2 time3 time4 time5 time6
## 1    1   0     0     0     0     0     0
## 2    1   0     0     0     0     0     0
## 3    1   0     1     0     0     0     0
## 4    1   0     1     0     0     0     0
## 5    1   0     0     1     0     0     0
## 6    1   0     0     1     0     0     0
## 7    1   0     0     0     1     0     0
## 8    1   0     0     0     1     0     0
## 9    1   0     0     0     0     1     0
## 10   1   0     0     0     0     1     0
## 11   1   0     0     0     0     0     1
```

```
## 12   1   0     0     0     0     0     1
## 13   0   1     0     0     0     0     0
## 14   0   1     0     0     0     0     0
## 15   0   1     1     0     0     0     0
## 16   0   1     1     0     0     0     0
## 17   0   1     0     1     0     0     0
## 18   0   1     0     1     0     0     0
## 19   0   1     0     0     1     0     0
## 20   0   1     0     0     1     0     0
## 21   0   1     0     0     0     1     0
## 22   0   1     0     0     0     1     0
## 23   0   1     0     0     0     0     1
## 24   0   1     0     0     0     0     1
## attr(,"assign")
## [1] 1 1 2 2 2 2 2
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
##
## attr(,"contrasts")$time
## [1] "contr.treatment"
```

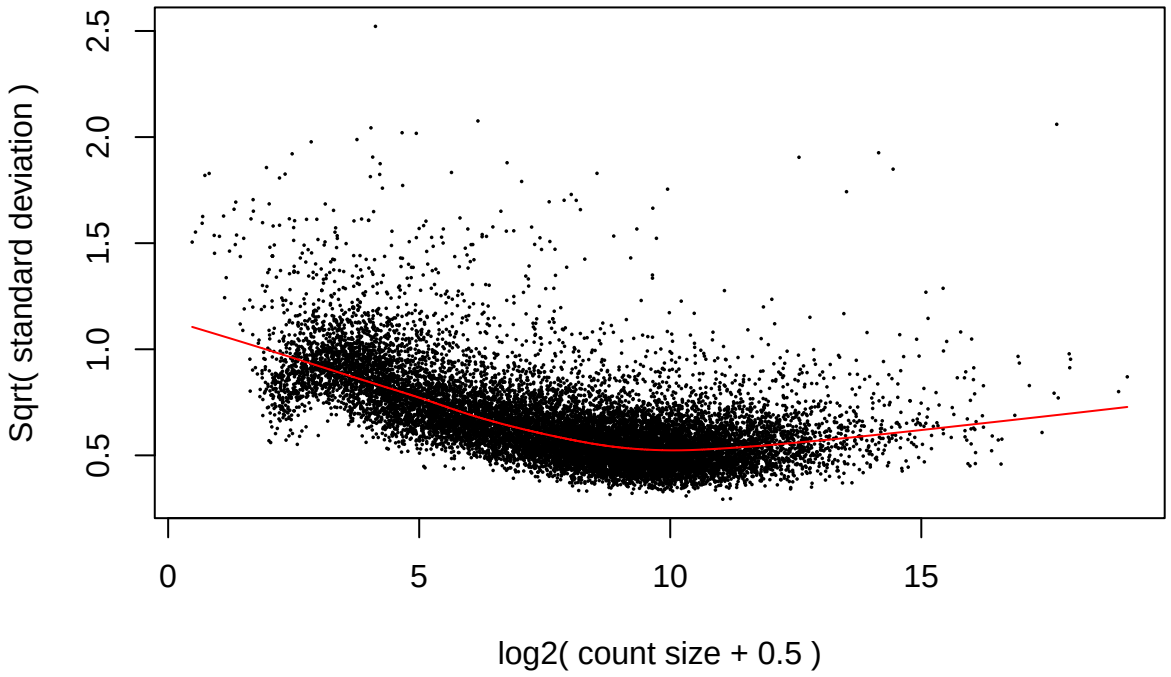
Matriz de contrastes

```
contr.matrix <- makeContrasts(
  Cyt_vs_Nuc=Cyt-Nuc,
levels=colnames(mod))
contr.matrix
```

```
##        Contrasts
## Levels  Cyt_vs_Nuc
##   Cyt            1
##   Nuc           -1
##   time2          0
##   time3          0
##   time4          0
##   time5          0
##   time6          0
```

Transformación de los contajes

```
v=voom(y,mod, plot = T)
```




```
v
```

```
## An object of class "EList"
## $targets
##     group lib.size norm.factors
## C11     1 11893148    0.5528799
## C12     1 12577493    0.4939034
## C21     1 21640913    0.7523256
## C22     1 11257449    0.5919619
## C31     1 16651577    0.6551185
## 19 more rows ...
##
## $E
##                                C11       C12        C21       C22        C31
## ENSMUSG00000000001.5   7.41769041  6.999624  7.0038817  7.238086  6.9286106
## ENSMUSG00000000028.16 -0.04849691  0.102115  0.2922309  0.151048  0.4970019
## ENSMUSG00000000049.12 10.74821284 10.904003 10.4399878 10.101458 11.0084391
## ENSMUSG00000000056.8   5.41520515  5.178535  5.3333218  5.477298  6.4556471
## ENSMUSG00000000058.7   2.87915225  3.122015  3.0839467  3.432004  2.0192286
##                                C32       C41        C42       C51        C52
## ENSMUSG00000000001.5   7.1213797  7.5322438  7.4404081  7.7162041  7.5372625
## ENSMUSG00000000028.16  0.9388105  0.7172853  0.6417667  0.5589109 -0.2178553
## ENSMUSG00000000049.12 10.3230318 10.6003962 10.6754778 10.5115282 10.9675970
## ENSMUSG00000000056.8   6.5584880  6.4496432  6.2147974  5.6526796  5.5305690
## ENSMUSG00000000058.7   2.3513058  2.8507683  3.2211265  2.7658733  2.6036439
##                                C61        C62       N11       N12        N21
## ENSMUSG00000000001.5   7.4699733  7.4457710 4.8594862 4.9100551 4.37901034
## ENSMUSG00000000028.16  0.3822517  0.2782851 0.5811576 0.7681821 0.05029308
## ENSMUSG00000000049.12 10.4799306 10.6746295 7.7264934 7.7242776 7.52791135
```

```
## ENSMUSG00000000056.8   5.4436065  5.4814015 5.1433559 4.6164185 5.07976263
## ENSMUSG00000000058.7   2.6021286  2.6157262 1.4365482 1.8382977 0.85115229
##                             N22       N31       N32       N41       N42
## ENSMUSG00000000001.5  4.4754608 4.5391206 4.8311813 4.4924810 4.7786400
## ENSMUSG00000000028.16 0.2360476 0.1913597 0.5474949 0.4860961 0.9127675
## ENSMUSG00000000049.12 7.5154059 7.4654490 7.9571668 8.0575674 7.8792975
## ENSMUSG00000000056.8  5.3094740 6.1197101 6.5386601 6.1744931 6.2318682
## ENSMUSG00000000058.7  1.0395571 1.4206632 1.7478820 1.5364012 1.5636271
##                             N51       N52      N61       N62
## ENSMUSG00000000001.5  4.6245347 4.5773720 4.504675 4.7119103
## ENSMUSG00000000028.16 0.3556403 0.6074139 1.027058 0.2392526
## ENSMUSG00000000049.12 7.3085262 7.4395512 8.013401 7.7266461
## ENSMUSG00000000056.8  4.9096192 5.1444748 5.344425 5.1824569
## ENSMUSG00000000058.7  1.2099927 1.1474948 1.426433 1.5248966
## 15427 more rows ...
##
## $weights
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 12.547479 12.437271 11.698924 12.977361 12.139134 11.914876 12.073884
## [2,]  1.852957  1.905666  2.370339  1.701688  2.355846  2.492315  2.343584
## [3,]  7.628909  7.526299  7.032741  8.233991  7.134699  6.953395  7.189490
## [4,] 11.515778 11.726411 13.252763 11.858217 12.965680 12.788434 13.226580
## [5,]  5.527684  5.703175  7.036548  4.917736  5.638416  5.976410  6.193725
##           [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## [1,] 12.235896 11.849880 11.730319 12.001646 12.065517 13.024856 12.610321
## [2,]  2.249940  2.130968  2.192628  2.236290  2.201264  4.878809  5.600988
## [3,]  7.327404  7.370893  7.272326  7.191495  7.245162  8.484761  8.011763
## [4,] 13.279295 12.843215 12.948143 12.895669 12.825994 12.751247 12.287418
## [5,]  5.935504  5.605907  5.781865  5.803560  5.706921  7.646862  8.622208
##          [,15]     [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
## [1,] 12.873796 13.054206 13.275616 13.277499 13.071180 12.690895 12.920226
## [2,]  5.370612  5.025751  4.966340  4.940544  5.442182  6.185743  4.888062
## [3,]  8.444388  8.676364  8.814380  8.832054  8.534257  8.077586  8.747755
## [4,] 11.892832 12.138628 11.024589 11.044582 11.126802 10.589689 12.422532
## [5,]  8.170995  7.710774  6.187996  6.159326  7.409608  8.325369  6.727684
##          [,22]     [,23]     [,24]
## [1,] 12.872513 12.623669 12.857035
## [2,]  4.972178  5.895131  5.484920
## [3,]  8.690574  8.076708  8.330871
## [4,] 12.366467 11.882399 12.158307
## [5,]  6.829084  7.852080  7.350246
## 15427 more rows ...
##
## $design
##   Cyt Nuc time2 time3 time4 time5 time6
## 1   1   0     0     0     0     0     0
## 2   1   0     0     0     0     0     0
## 3   1   0     1     0     0     0     0
## 4   1   0     1     0     0     0     0
## 5   1   0     0     1     0     0     0
## 19 more rows ...
```

Selección de genes diferencialmente expresados

Como en el caso de los microarrays el objeto v y las matrices de diseño y contrastes se utilizaran para ajustar un modelo y, a continuación realizar las comparaciones especificadas sobre el modelo ajustado. El proceso finaliza con la regularización del estimador del error usando la función eBayes.

```
fit=lmFit(v,mod)
fit2 <- contrasts.fit(fit, contr.matrix)
fit2 <- eBayes(fit2)

(results<-topTable(fit2, coef = 1, adjust="BH"))
```

```
##                            logFC  AveExpr         t      P.Value    adj.P.Val
## ENSMUSG00000035545.14 -6.459982 5.423469 -56.59511 2.779228e-25 4.288905e-21
## ENSMUSG00000028416.14  4.534671 5.410686  49.97719 4.174486e-24 1.107089e-20
## ENSMUSG00000043923.9  -4.113639 4.638951 -50.92809 2.769516e-24 1.107089e-20
## ENSMUSG00000022194.16 -4.014916 6.527244 -50.40338 3.469971e-24 1.107089e-20
## ENSMUSG00000020803.4   4.339140 5.013828  49.82193 4.466999e-24 1.107089e-20
## ENSMUSG00000086583.5   5.032834 7.522693  49.11441 6.098155e-24 1.107089e-20
## ENSMUSG00000036438.15  3.220144 6.371528  48.78012 7.075162e-24 1.107089e-20
## ENSMUSG00000054733.10  3.850395 6.043818  48.74903 7.173982e-24 1.107089e-20
## ENSMUSG00000028248.16 -4.522672 6.168446 -48.76093 7.135985e-24 1.107089e-20
## ENSMUSG00000064368.1   7.182302 2.830871  49.10757 6.116657e-24 1.107089e-20
##                             B
## ENSMUSG00000035545.14 46.83333
## ENSMUSG00000028416.14 45.26813
## ENSMUSG00000043923.9  45.22739
## ENSMUSG00000022194.16 45.20594
## ENSMUSG00000020803.4  45.20456
## ENSMUSG00000086583.5  44.80913
## ENSMUSG00000036438.15 44.75978
## ENSMUSG00000054733.10 44.75175
## ENSMUSG00000028248.16 44.45591
## ENSMUSG00000064368.1  44.43370
```

```
summary(decideTests(fit2))
```

```
##        Cyt_vs_Nuc
## Down         7432
## NotSig       2122
## Up           5878
```

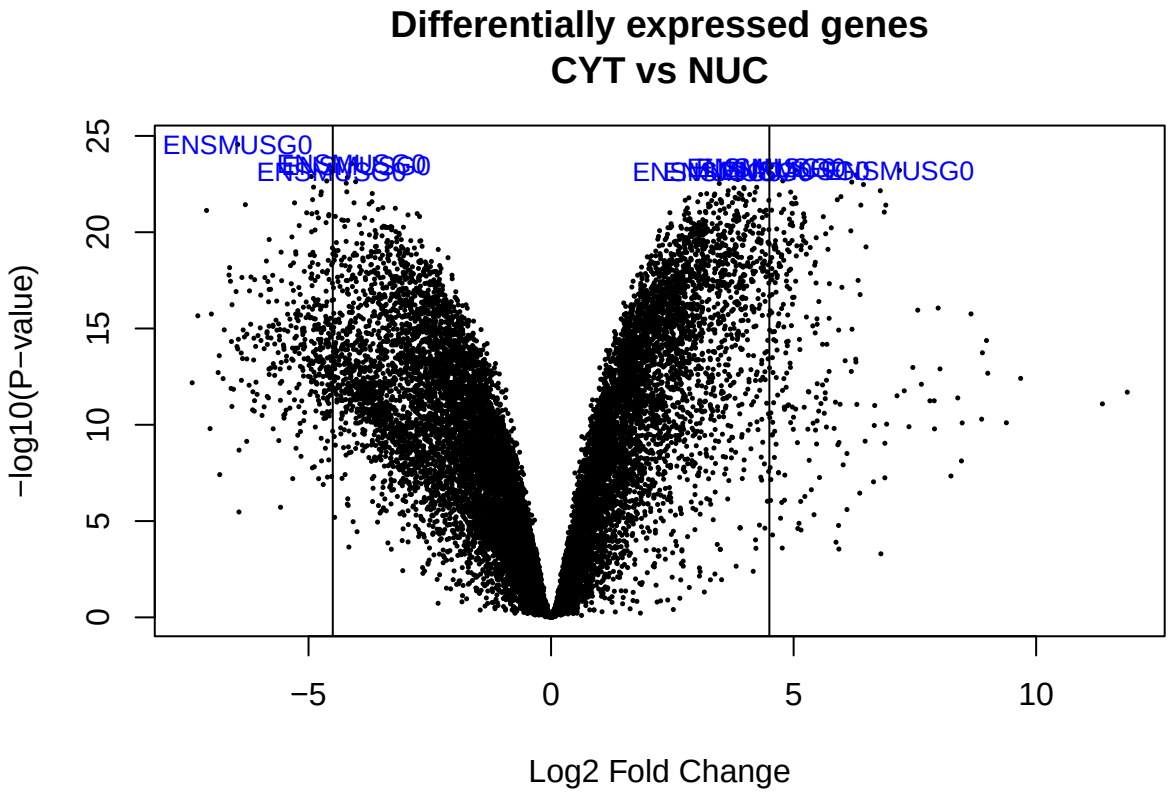
```
summa.fit <- decideTests(fit2, p.value = 0.01, lfc = 4.5)
summary(summa.fit)
```

```
##        Cyt_vs_Nuc
## Down          371
## NotSig      14809
## Up            252
```

Visualización de los resultados

Volcano Plot

```
volcanoplot(fit2, coef = 1, highlight = 10,names=rownames(fit2) ,main =paste( "Differentially expressed
abline(v=c(-4.5,4.5))
```




```
pdf("volcanoplot.pdf")
volcanoplot(fit2, coef = 1, highlight = 10,names=rownames(fit2) ,main =paste( "Differentially expressed
abline(v=c(-4.5,4.5))
dev.off()
```

```
## pdf
##   2
```

Perfiles de expresión

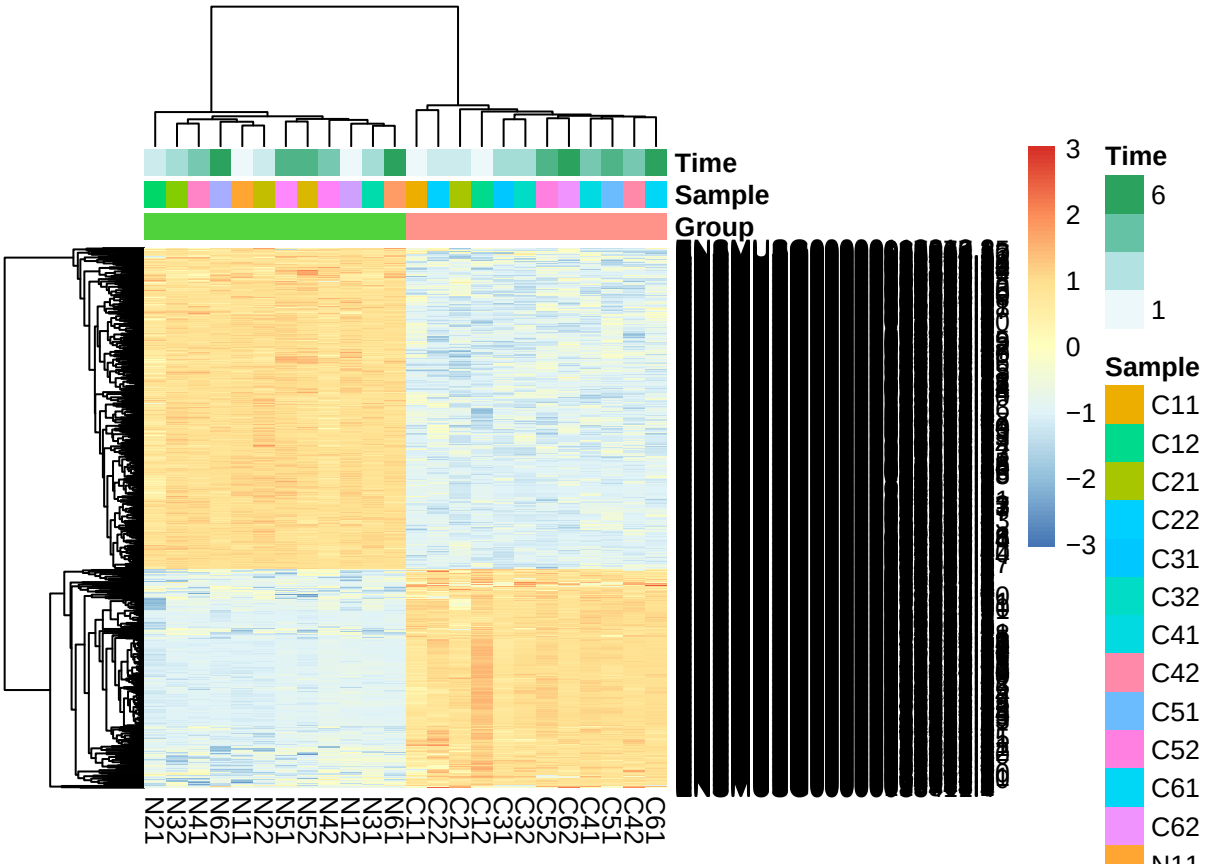
Con el fin de observar si existen perfiles de expresión diferenciados podemo realizar un mapa de colores con los genes más diferencialmente expresados.

Es decir, fijamos un criterio de selección de genes y retenemos aquellos componentes de la tabla de resultados que lo cumplen. Por ejemplo: Genes con un p-valor ajustado inferior a 0.001 y un 'fold-change' superior a 6 o inferior a -6.

mapa de colores

```
for (i in  colnames(fit2$coefficients)){
  top=topTable(fit2,coef=i,sort="p", n=Inf)
  genes=rownames(top[which(top$adj.P.Val<0.01 & abs(top$logFC)>4.5),])
  write.table(top,paste(i,"_limma_voom.txt",sep=""),quote=F)
  term1=strsplit(i,split="_vs_")[[1]][1]
  term2=strsplit(i,split="_vs_")[[1]][2]
  samples=rownames(subset(info,group==term1 | group==term2))
  expr=v$E[genes,samples]
  rownames(expr)=do.call(rbind, strsplit(genes, ','))[,1]
  if (length(genes) >1) {
    pdf(paste("pheatmap_DE_genes__01_",i,".pdf",sep=""), width = 10, height = 12)
    pheatmap(expr,scale="row",annotation_col=info[,c("Group","Sample", "Time")], border_color = "NA",sh
    dev.off()
  }}

write.table(v$E,"logcpm.txt",quote=F)
```

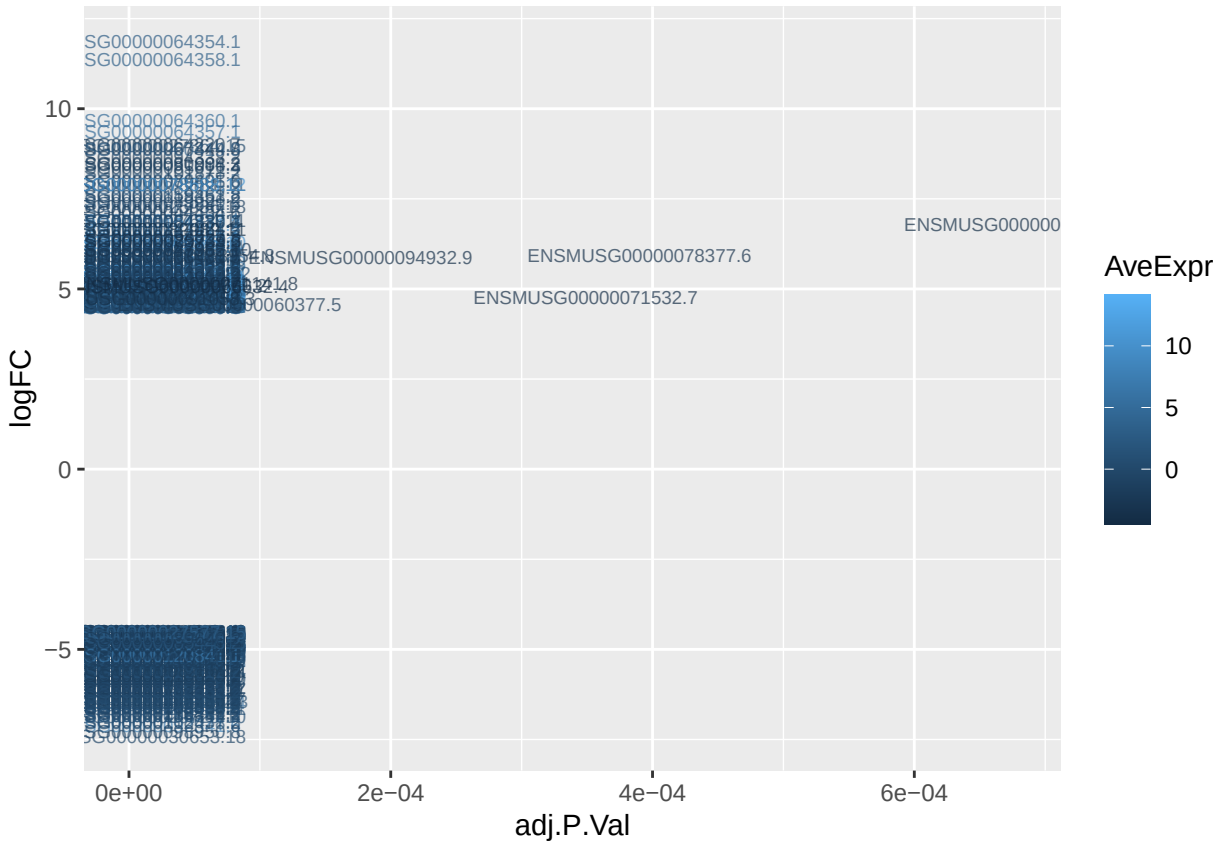
```
for (i in  colnames(fit2$coefficients)){
  top=topTable(fit2,coef=i,sort="p", n=Inf)
  genes=rownames(top[which(top$adj.P.Val<0.01 & abs(top$logFC)>4.5),])
  write.table(top,paste(i,"_limma_voom.txt",sep=""),quote=F)
  term1=strsplit(i,split="_vs_")[[1]][1]
  term2=strsplit(i,split="_vs_")[[1]][2]
  samples=rownames(subset(info,group==term1 | group==term2))
  expr=v$E[genes,samples]
  rownames(expr)=do.call(rbind, strsplit(genes, ','))[,1]
  if (length(genes) >1) {
    pheatmap(expr,scale="row",annotation_col=info[,c("Group","Sample", "Time")], border_color = "NA",sh
  }}
```




```
length(which(top$adj.P.Val < 0.01 & abs(top$logFC) > 4.5))
```

```
## [1] 623
```

```
p_data <-   top  %>% filter(adj.P.Val <0.01 & abs(logFC) > 4.5)

p_data %>%  ggplot(aes(x=adj.P.Val,y=logFC)) +
  geom_text(label=rownames(p_data), size=2.5,alpha=0.7, aes(col=AveExpr))
```



Top tables

```
genes_sin_version <- sub("\\.\\d+$", "", rownames(top))
top$Gene <- rownames(top) <- genes_sin_version
DEGs <- top %>%  arrange(logFC) %>% filter(adj.P.Val <0.01 & abs(logFC) > 4.5)
head(DEGs)
```

```
##                         logFC    AveExpr         t      P.Value    adj.P.Val
## ENSMUSG00000030653 -7.398417 -0.5122127 -14.80990 6.595322e-13 2.545403e-12
## ENSMUSG00000098950 -7.281809 -0.5707764 -21.85511 2.188555e-16 2.026022e-15
## ENSMUSG00000092341 -7.101681  7.3674403 -39.37151 7.415307e-22 1.272453e-19
## ENSMUSG00000112478 -7.029979  0.7886766 -11.17571 1.582125e-10 4.097222e-10
## ENSMUSG00000108037 -7.002784  0.2829931 -22.07625 1.770873e-16 1.684840e-15
## ENSMUSG00000029720 -6.862932  0.9067325 -15.73757 1.940354e-13 8.352453e-13
##                           B               Gene
## ENSMUSG00000030653 19.70602 ENSMUSG00000030653
## ENSMUSG00000098950 27.44390 ENSMUSG00000098950
## ENSMUSG00000092341 39.98658 ENSMUSG00000092341
## ENSMUSG00000112478 14.20125 ENSMUSG00000112478
## ENSMUSG00000108037 27.68690 ENSMUSG00000108037
## ENSMUSG00000029720 20.91868 ENSMUSG00000029720
```

```
write.table(DEGs, file = "./DEG_liver.txt", row.names = F, sep = "\t", quote = F)
```

```
genes_sin_version <- sub("\\.\\d+$", "", rownames(top))
top$Gene <- rownames(top) <- genes_sin_version
top <- top[,c("Gene", names(top)[1:6])]
write.table(top, file = "./Cyt_v_Nuc.txt", row.names = F, sep = "\t", quote = F)
```

Análisis de significació biológica

Nos centraremos únicamente en la lista de genes "up-regulados" y "down-regulados" es decir diferencialmente expresados con un logFC mayor que seis (más expresados en "cytosol" que en "nucleo").

Para el análisis de enriquecimiento utilizaremos la función `enrichGO` del paquete `clusterProfiler muy parecida a las de otros paquetes como`GOstats'.

Análisis de sobrerrepresentación usando clusterProfiler

```
head(top)
```

```
##                                   Gene     logFC  AveExpr         t      P.Value
## ENSMUSG00000035545 ENSMUSG00000035545 -6.459982 5.423469 -56.59511 2.779228e-25
## ENSMUSG00000043923 ENSMUSG00000043923 -4.113639 4.638951 -50.92809 2.769516e-24
## ENSMUSG00000022194 ENSMUSG00000022194 -4.014916 6.527244 -50.40338 3.469971e-24
## ENSMUSG00000028416 ENSMUSG00000028416  4.534671 5.410686  49.97719 4.174486e-24
## ENSMUSG00000020803 ENSMUSG00000020803  4.339140 5.013828  49.82193 4.466999e-24
## ENSMUSG00000086583 ENSMUSG00000086583  5.032834 7.522693  49.11441 6.098155e-24
##                       adj.P.Val        B
## ENSMUSG00000035545 4.288905e-21 46.83333
## ENSMUSG00000043923 1.107089e-20 45.22739
## ENSMUSG00000022194 1.107089e-20 45.20594
## ENSMUSG00000028416 1.107089e-20 45.26813
## ENSMUSG00000020803 1.107089e-20 45.20456
## ENSMUSG00000086583 1.107089e-20 44.80913
```

```
allEntrezs <- genes_sin_version
selectedEntrezsUP <- rownames(subset(top, (abs(logFC) > 4.5) & (adj.P.Val < 0.01)))
length(allEntrezs); length(selectedEntrezsUP)
```

```
## [1] 15432
```

```
## [1] 623
```

```
library(clusterProfiler)
library(org.Mm.eg.db)
ego <- enrichGO(gene = selectedEntrezsUP,
                universe = allEntrezs,
                keyType = "ENSEMBL",
                OrgDb = org.Mm.eg.db,
                ont = "BP",
                pAdjustMethod = "BH",
                qvalueCutoff = 0.01,
                readable = TRUE)
```

El objeto resultante almacena las categorías GO enriquecidas, los genes anotados en ellas y los valores de los estadísticos que llevan a afirmar que dichas categorías se encuentran significativamente sobre-representadas como resultado de un test de enriquecimiento.

```
head(ego)
```

```
##                    ID                                          Description
## GO:0002181 GO:0002181                              cytoplasmic translation
## GO:0006119 GO:0006119                            oxidative phosphorylation
## GO:0009060 GO:0009060                                 aerobic respiration
## GO:0045333 GO:0045333                               cellular respiration
## GO:0042776 GO:0042776 proton motive force-driven mitochondrial ATP synthesis
## GO:0015986 GO:0015986              proton motive force-driven ATP synthesis
##            GeneRatio   BgRatio       pvalue     p.adjust       qvalue
## GO:0002181    34/223 137/12773 4.768053e-30 8.825665e-27 8.783255e-27
## GO:0006119    31/223 136/12773 3.235070e-26 2.994057e-23 2.979669e-23
## GO:0009060    31/223 178/12773 1.852241e-22 1.142832e-19 1.137341e-19
## GO:0045333    32/223 225/12773 2.268437e-20 1.049719e-17 1.044675e-17
## GO:0042776    20/223  65/12773 4.158979e-20 1.539654e-17 1.532255e-17
## GO:0015986    20/223  69/12773 1.596193e-19 4.924255e-17 4.900592e-17
##
## GO:0002181 Rpl23/Rpl21/Rps3a1/Rpsa/Rpl26/Rpl23a/Rpl31/Rpl32/Rpl38/Rps3/Rps14/Rps23/Rpl19/Rpl39/Rps20/
## GO:0006119                            ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5mf/Mli
## GO:0009060                            ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5mf/Mli
## GO:0045333                       ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5mf/Mlxipl/C
## GO:0042776
## GO:0015986
##            Count
## GO:0002181    34
## GO:0006119    31
## GO:0009060    31
## GO:0045333    32
## GO:0042776    20
## GO:0015986    20
```

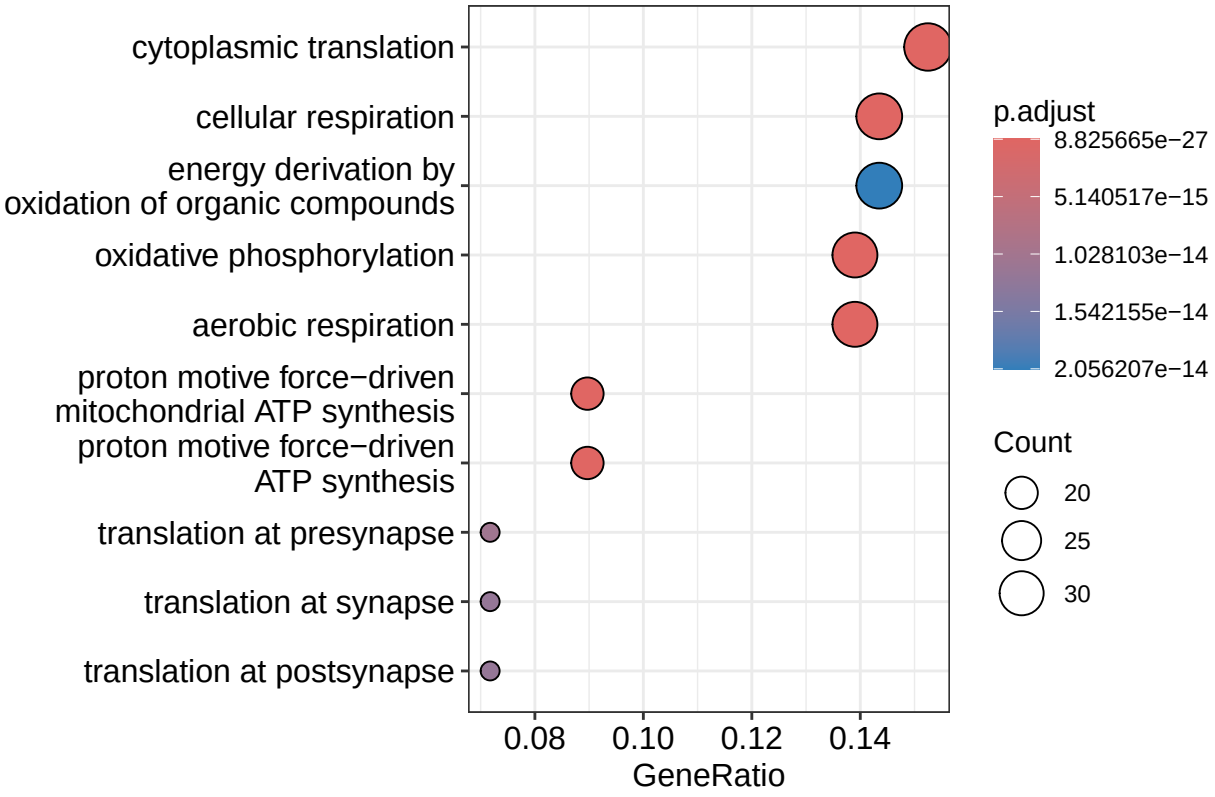
```
ego_results <- data.frame(ego)
write.csv(ego_results, "clusterProfiler_ORAresults_UpGO.csv")
```

Visualización de los resultados del análisis de enriquecimiento

Uno de los aspectos interesantes del paquete `clusterProfiler` es que permite visualizar los resultados mediante algunos gráficos creados específicamente para tal fin.

##Dotplot de los 9 términos más enriquecidos Este gráfico compara visualmente las categorías enriquecidas (de más a menos enriquecidas) visualizando simultáneamente cuan enriquecidas estan y el p-valor del test de enriquecimiento.

```
dotplot(ego, showCategory=10)
```




```
pdf(paste("dotplot1.pdf",sep=""))
dotplot(ego, showCategory=10)
dev.off()
```

```
## pdf
##   2
```

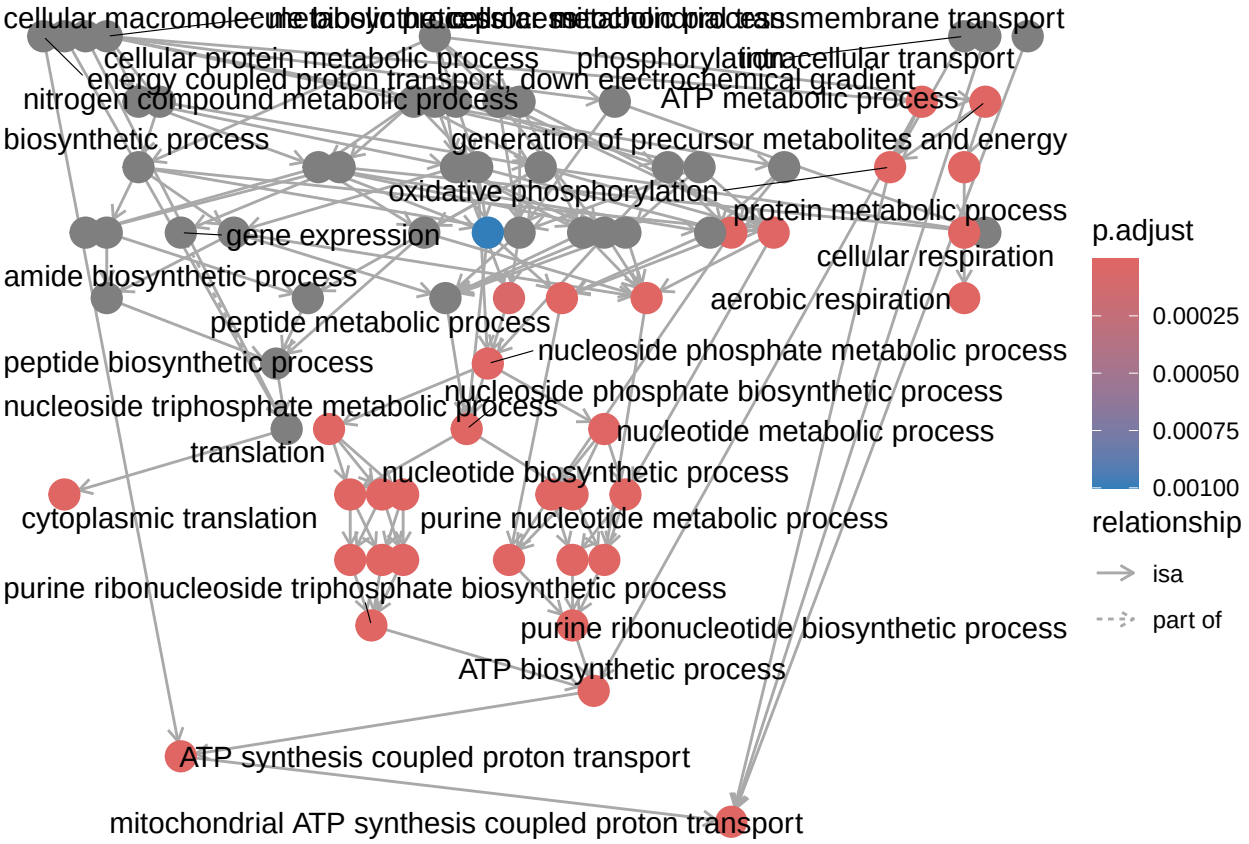
Visualización jerárquica de los términos GO

Este gráfico permite visualizar los términos seleccionados dentro del sub-grafo de la GO que los contiene. Esto nos, permite por ejemplo, hacernos una idea de si estan muy dispersos, o no, en la jerarquía y de si se trata de términos muy generales o más específicos.

```
pdf(paste("GO1.pdf",sep=""))
goplot(ego, showCategory=5, cex=0.5)
dev.off()
```

```
## pdf
##   2
```

```
goplot(ego, showCategory=5, cex=0.5)
```

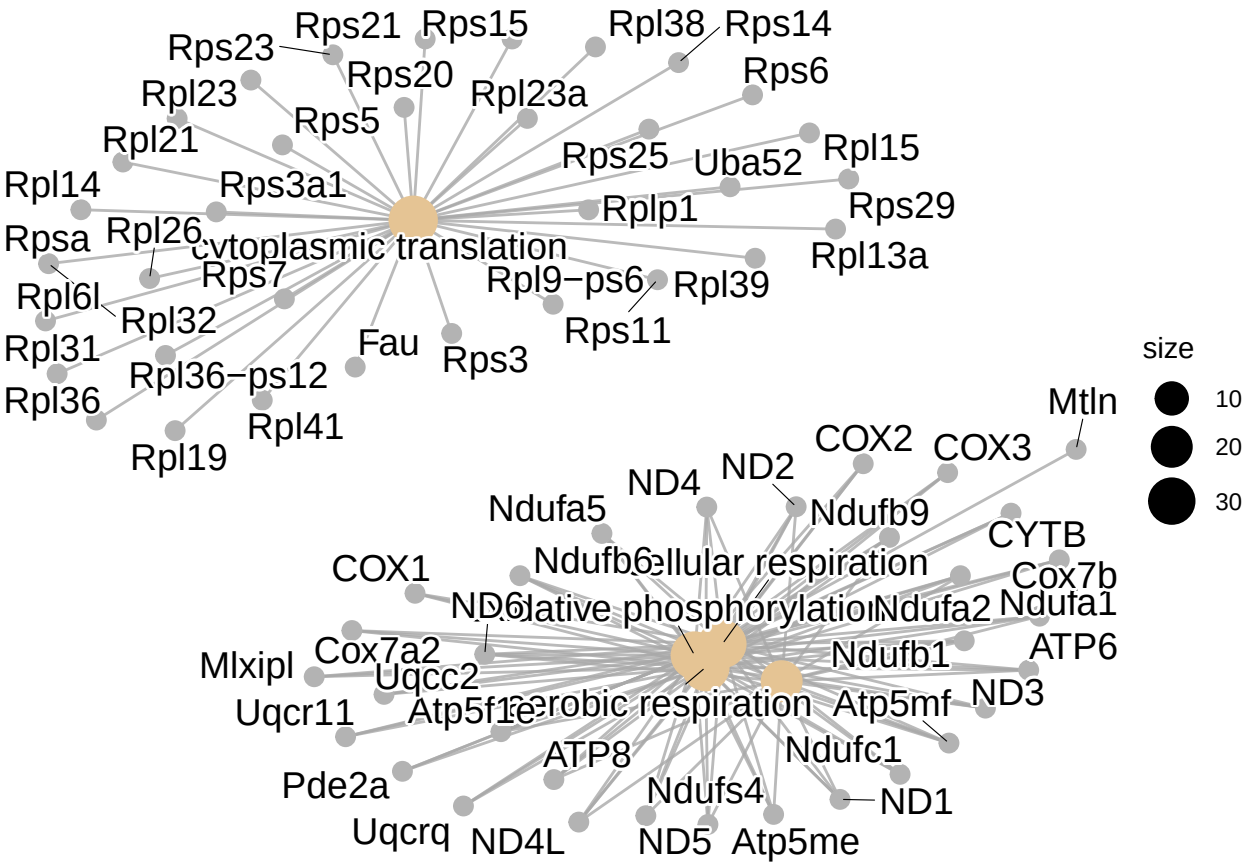


De forma parecida una red de genes nos permite visualizar la asociación entre los genes y las categorías seleccionadas en las que éstos genes estan anotados.

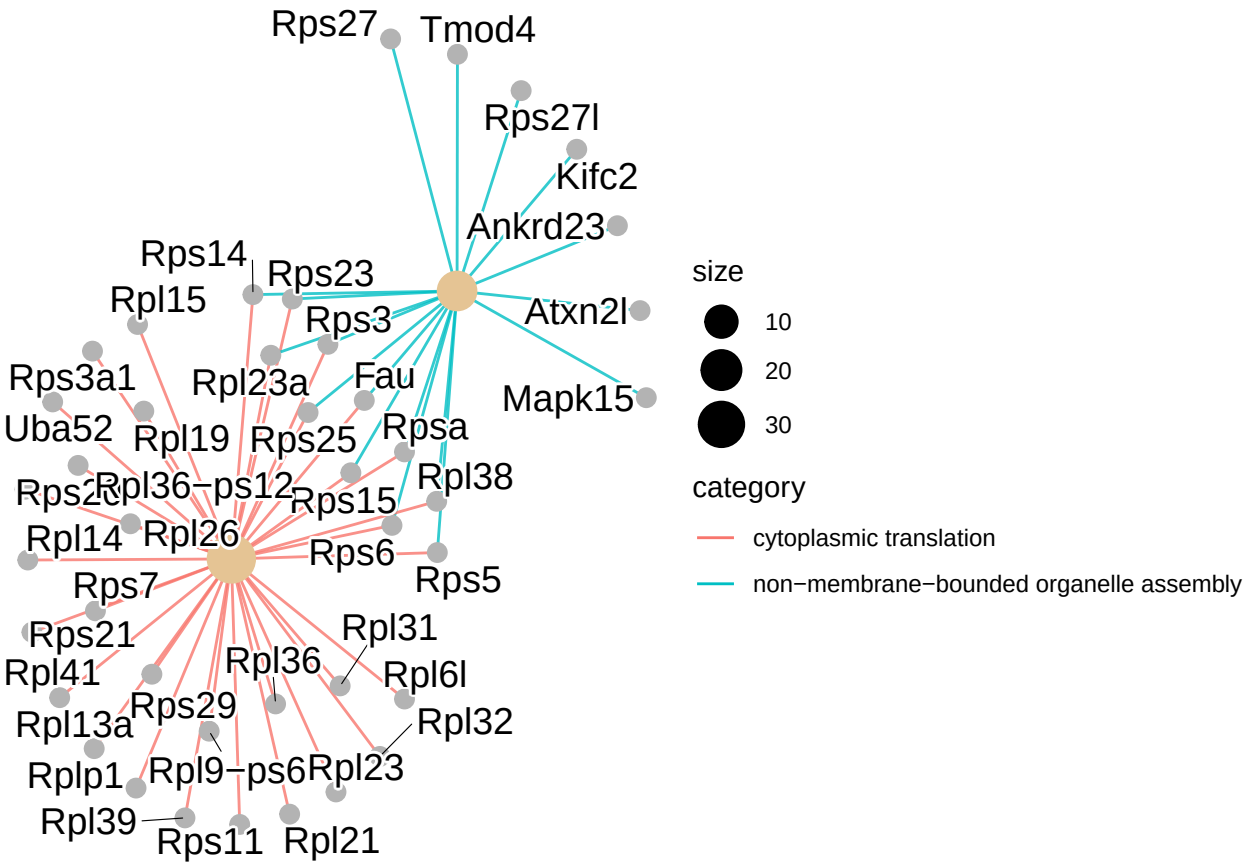
```
## Gene network para los términos seleccionados
pdf(paste("cneplot1.pdf",sep=""))
cnetplot(ego)
dev.off()
```

```
## pdf
##   2
```

```
cnetplot(ego)
```




```
library(clusterProfiler)
library(ggplot2)
ego2 = clusterProfiler::simplify(ego, cutoff = 0.01, by = "p.adjust")
png("./cnetplot_transp1.png", units = "in", width = 24, height = 16, res = 600,
    bg = "transparent")
par(bg = NA)
a <- cnetplot(ego2, showCategory = 5, cex_category = 1, cex_label_category = 2.5,
    cex_gene = 1, cex_label_gene = 1, circular = FALSE, colorEdge = TRUE)
a
invisible(dev.off())
a
```

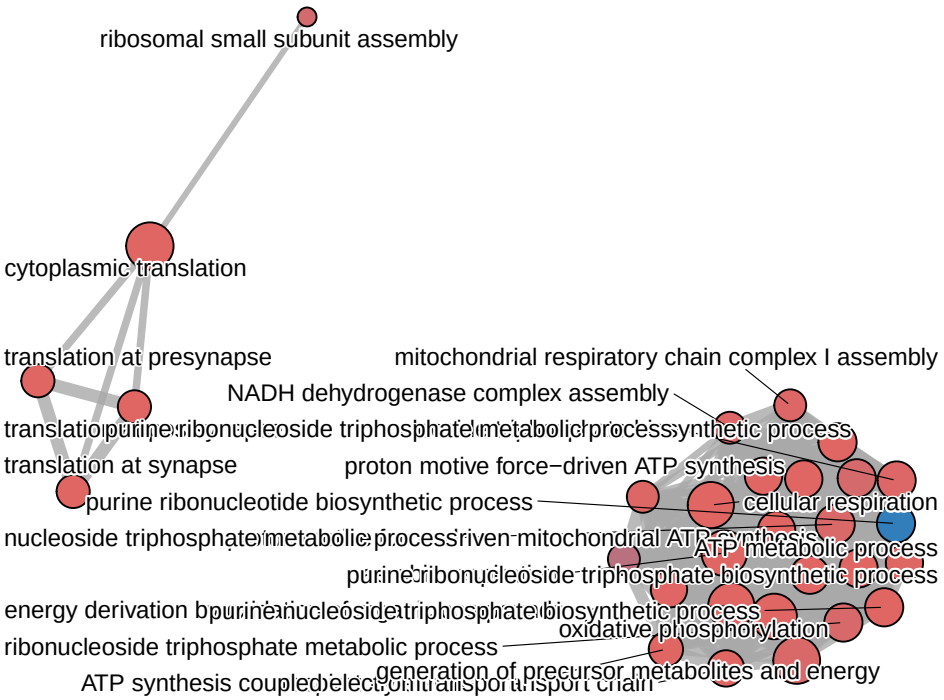


Finalmente este gráfico permite simplificar las visualizaciones y agrupa los 104 términos más significativos basándose en alguna medida de similaridad entre los mismos (por ejemplo "similaridad semántica" definida a partir de su interdistancia dentro del grafo).

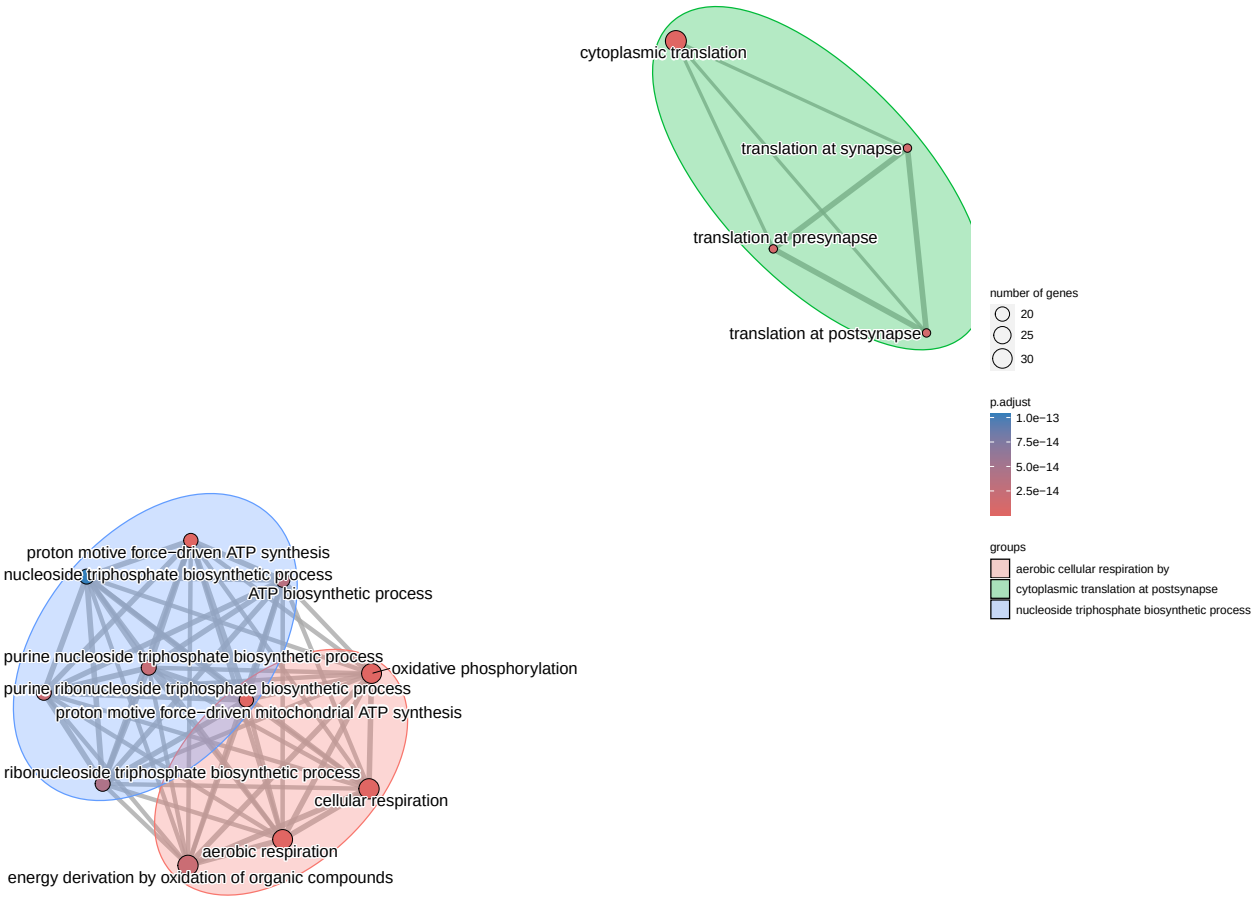
```
## Enrichment Map
library(enrichplot)
ego_sim <- pairwise_termsim(ego)
pdf(paste("emaplot1.pdf",sep=""))
emapplot(ego_sim, cex_label_category=0.6)
dev.off()
```

```
## pdf
##   2
```

```
emapplot(ego_sim, cex_label_category=0.6)
```



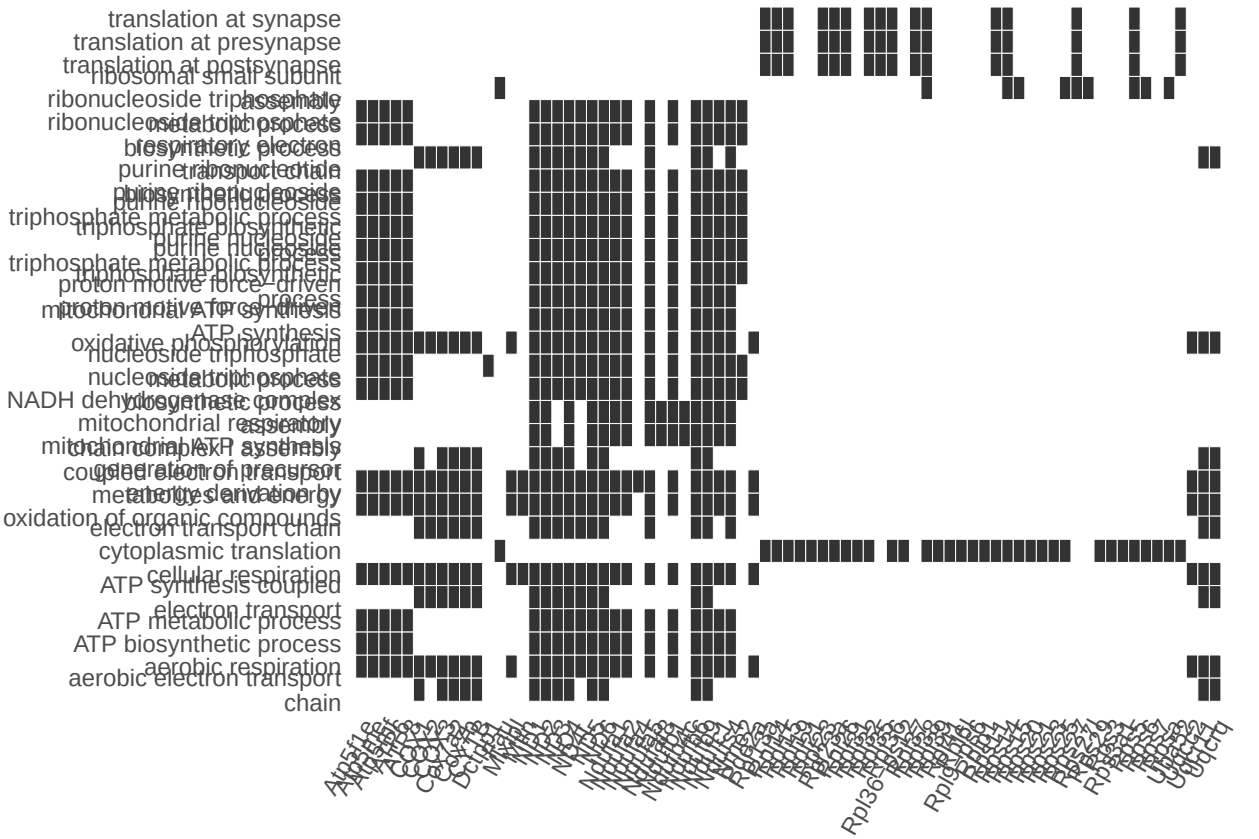

```
term_similarity_matrix = pairwise_termsim(ego)
emapplot(term_similarity_matrix, showCategory = 15, group_category = TRUE, group_legend = TRUE)
```



```
pdf(paste("emaplot_grouped1.pdf",sep=""),width = 10, height = 10)
emapplot(term_similarity_matrix, showCategory = 15, group_category = TRUE, group_legend = TRUE)
dev.off()
```

```
## pdf
##   2
```

```
library(enrichplot)
heatplot(ego)
```




```
pdf(paste("heatplot_ego1.pdf",sep=""),width = 9, height = 10)
heatplot(ego)
dev.off()
```

```
## pdf
##   2
```

```
library(geneset)
library(genekitr)
```

```
## Welcome to use genekitr! (Vignette: https://www.genekitr.fun)
## Citation for genekitr:
## Liu, Y., Li, G. Empowering biologists to decode omics data: the Genekitr R package and web server. BI
```

```
selectedEntrezsUP <- rownames(subset(top, (abs(logFC) > 4.5) & (adj.P.Val < 0.01)))
mm10_gs <- geneset::getGO(org = "mouse",ont = "bp")
dim(mm10_gs); length(selectedEntrezsUP)
```

```
## NULL
```

```
## [1] 623
```

```
# ORA analysis
ego3 <- genORA(selectedEntrezsUP, geneset = mm10_gs, p_cutoff = 0.01, q_cutoff = 0.01)
# next we only show ten sample terms
ego3 <- ego3[1:10, ]
head(ego3)
```

```
##              Mm_BP_ID                                                Description
## GO:0002181 GO:0002181                                  cytoplasmic translation
## GO:0006119 GO:0006119                                oxidative phosphorylation
## GO:0009060 GO:0009060                                     aerobic respiration
## GO:0045333 GO:0045333                                   cellular respiration
## GO:0042776 GO:0042776 proton motive force-driven mitochondrial ATP synthesis
## GO:0015986 GO:0015986               proton motive force-driven ATP synthesis
##             GeneRatio   BgRatio       pvalue     p.adjust       qvalue
## GO:0002181 0.12749004 148/29312 1.107852e-35 2.079439e-32 2.013959e-32
## GO:0006119 0.11952191 149/29312 1.709183e-32 1.604068e-29 1.553557e-29
## GO:0009060 0.11952191 202/29312 2.540830e-28 1.589712e-25 1.539654e-25
## GO:0045333 0.12350598 261/29312 3.485518e-26 1.574982e-23 1.525387e-23
## GO:0042776 0.07968127  65/29312 4.195476e-26 1.574982e-23 1.525387e-23
## GO:0015986 0.08764940 101/29312 6.665334e-25 2.085139e-22 2.019479e-22
##
## GO:0002181 ENSMUSG00000071415/ENSMUSG00000041453/ENSMUSG00000028081/ENSMUSG00000032518/ENSMUSG000000
## GO:0006119                                        ENSMUSG00000064368/ENSMUSG00000064370/ENSMUSG000000
## GO:0009060                                        ENSMUSG00000064368/ENSMUSG00000064370/ENSMUSG000000
## GO:0045333                    ENSMUSG00000064368/ENSMUSG00000064370/ENSMUSG00000064351/ENSMUSG000000
## GO:0042776
## GO:0015986
##
## GO:0002181 Rpl23/Rpl21/Rps3a1/Rpsa/Rpl26/Rpl23a/Rpl31/Rpl32/Rpl38/Rps3/Rps14/Rps23/Rpl19/Rpl39/Rps20/
## GO:0006119                   ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5j2/Mlxipl/Cox
## GO:0009060                   ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5j2/Mlxipl/Cox
## GO:0045333              ND6/CYTB/COX1/Ndufs4/ND1/Ndufb9/ND2/Ndufa5/Ndufa1/ND4/Atp5j2/Mlxipl/Cox7a2/ND
## GO:0042776                                                                                  ND6/Ndufs4/ND
## GO:0015986                                                                 ND6/Ndufs4/ND1/Ndufb9/ND2/
##             Count FoldEnrich RichFactor
## GO:0002181    32   25.24992  0.2162162
## GO:0006119    30   23.51293  0.2013423
## GO:0009060    30   17.34369  0.1485149
## GO:0045333    31   13.87053  0.1187739
## GO:0042776    20   35.93258  0.3076923
## GO:0015986    22   25.43742  0.2178218
```

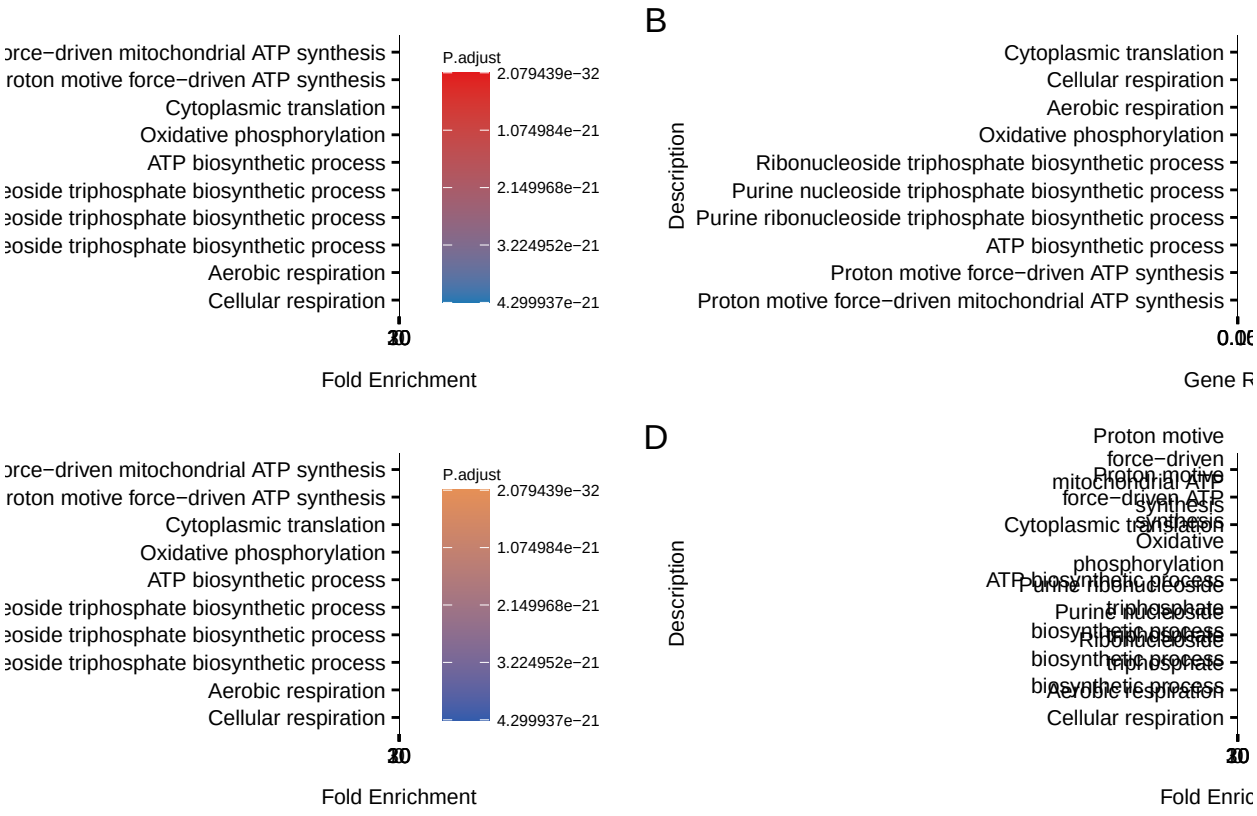
Bar Plot

```
library(patchwork)
```

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:genefilter':
##
##     area
```

```
p1 <- plotEnrich(ego3, plot_type = "bar")
p2 <- plotEnrich(ego3, plot_type = "bar", term_metric = "GeneRatio", stats_metric = "pvalue")
p3 <- plotEnrich(ego3, plot_type = "bar", up_color = "#E69056", down_color = "#325CAC")
p4 <- plotEnrich(ego3, plot_type = "bar", wrap_length = 25)
p1 + p2 + p3 + p4 + plot_annotation(tag_levels = "A")
```

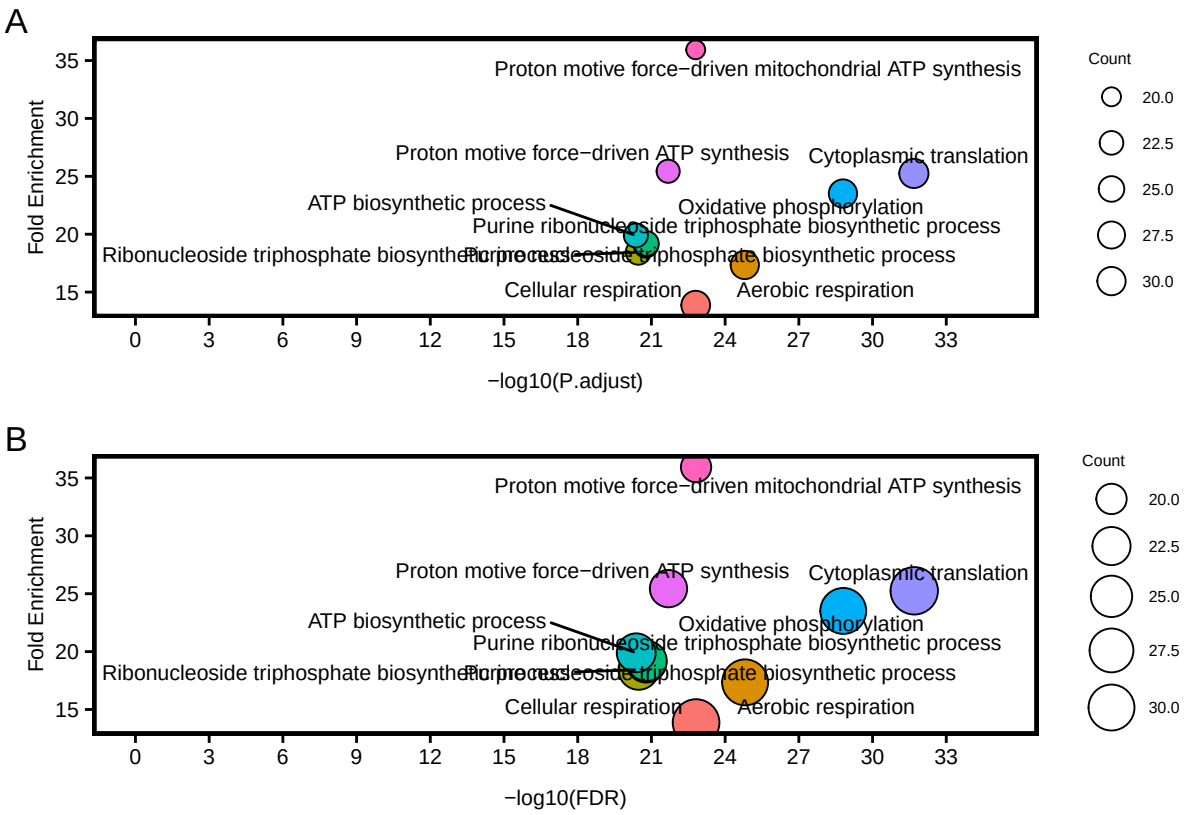



```
pdf(paste("BarPlot.pdf",sep=""))
p1 + p2 + p3 + p4 + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

Bubble Plot

```
library(patchwork)
p1 <- plotEnrich(ego3, plot_type = "bubble")
p2 <- plotEnrich(ego3, plot_type = "bubble",
                 scale_ratio = 0.5, stats_metric = "qvalue")
p1 / p2 + plot_annotation(tag_levels = "A")
```



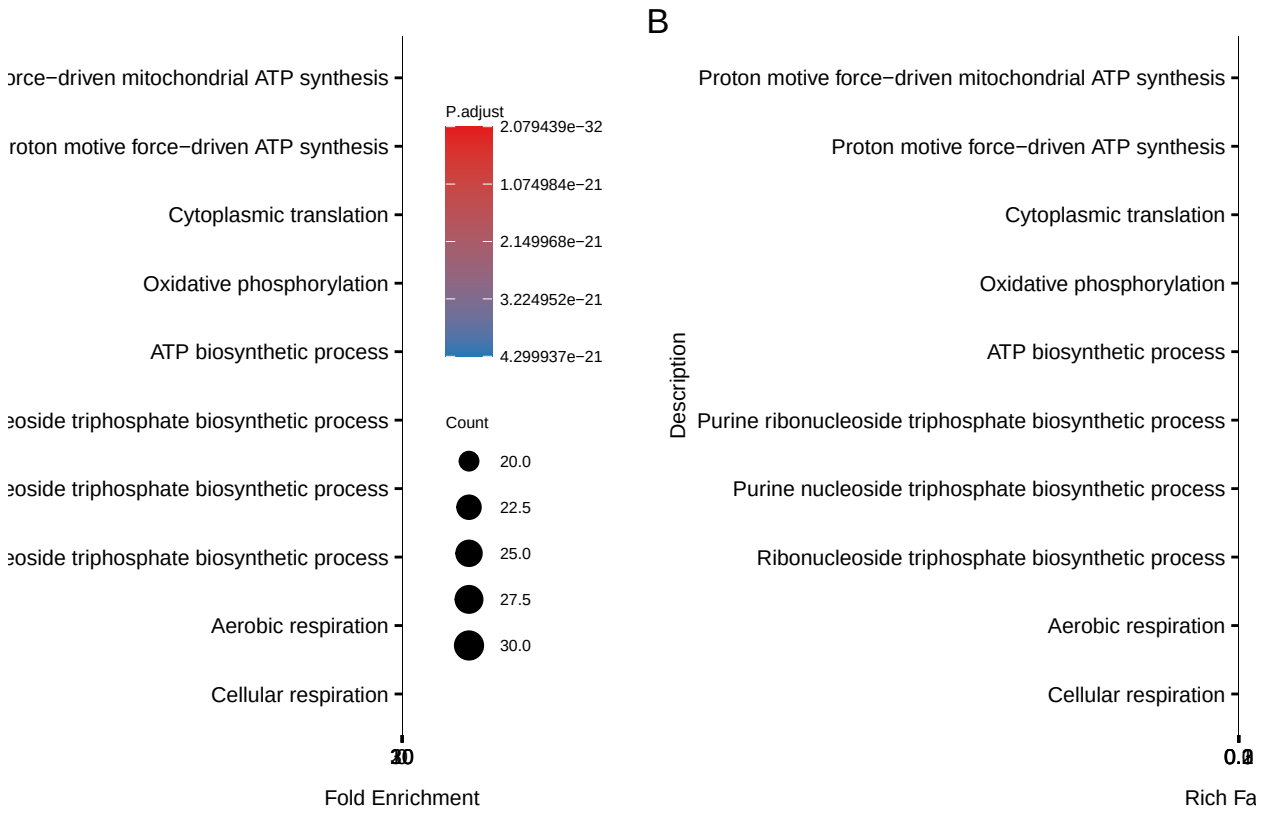

```
pdf(paste("BubblePlot.pdf",sep=""))
p1 / p2 + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

Dot Plot

```
library(patchwork)
p1 <- plotEnrich(ego3, plot_type = "dot")
p2 <- plotEnrich(ego3,
  plot_type = "dot",
  scale_ratio = 0.2,
  stats_metric = "pvalue",
```

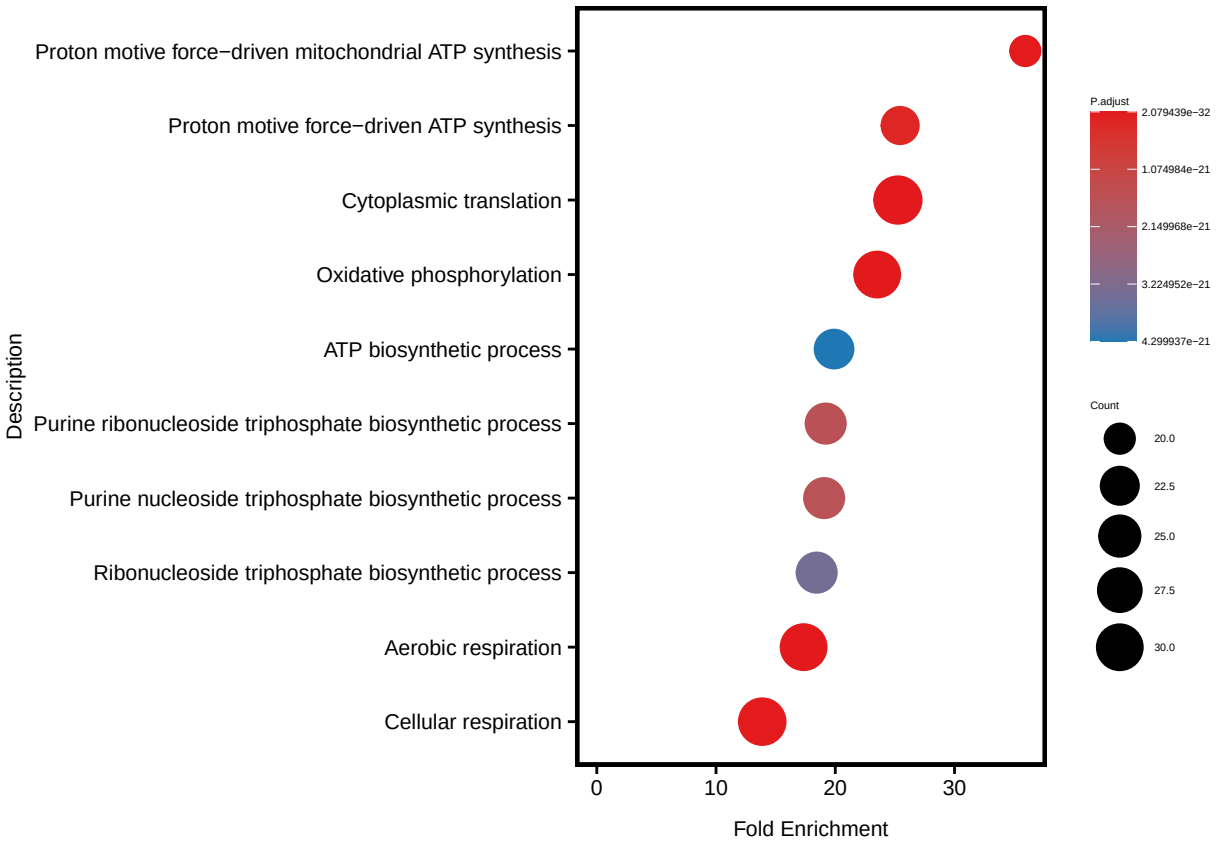
```
  term_metric = "RichFactor"
)
p1 + p2 + plot_annotation(tag_levels = "A")
```




```
pdf(paste("DOTPlot.pdf",sep=""))
p1 + p2 + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

```
plotEnrich(ego3,
           plot_type = 'dot',
           scale_ratio = 0.5, # dot size
           main_text_size = 8,
           legend_text_size = 4,
           n_term = 6) # show terms
```



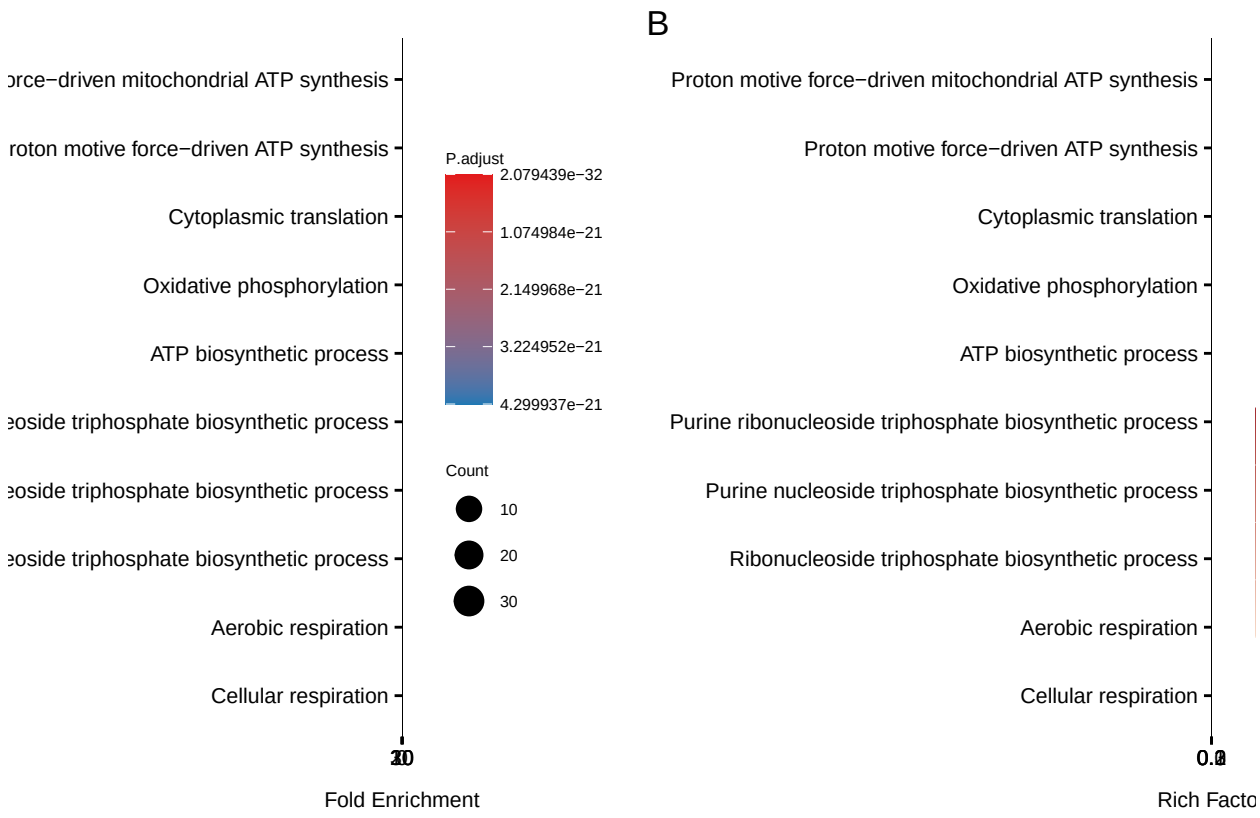

```
pdf(paste("PlotEnrich.pdf",sep=""))
plotEnrich(ego3,
           plot_type = 'dot',
           scale_ratio = 0.5, # dot size
           main_text_size = 8,
           legend_text_size =4,
           n_term = 6) # show terms
dev.off()
```

```
## pdf
##   2
```

Lollipop Plot

```
library(patchwork)
p1 <- plotEnrich(ego3, plot_type = "lollipop")
p2 <- plotEnrich(ego3,
  plot_type = "lollipop",
  scale_ratio = .5,
  stats_metric = "pvalue",
  term_metric = "RichFactor",
  up_color = "#a32a31",
  down_color = "#f7dcca"
```

```
)
p1 + p2 + plot_annotation(tag_levels = "A")
```

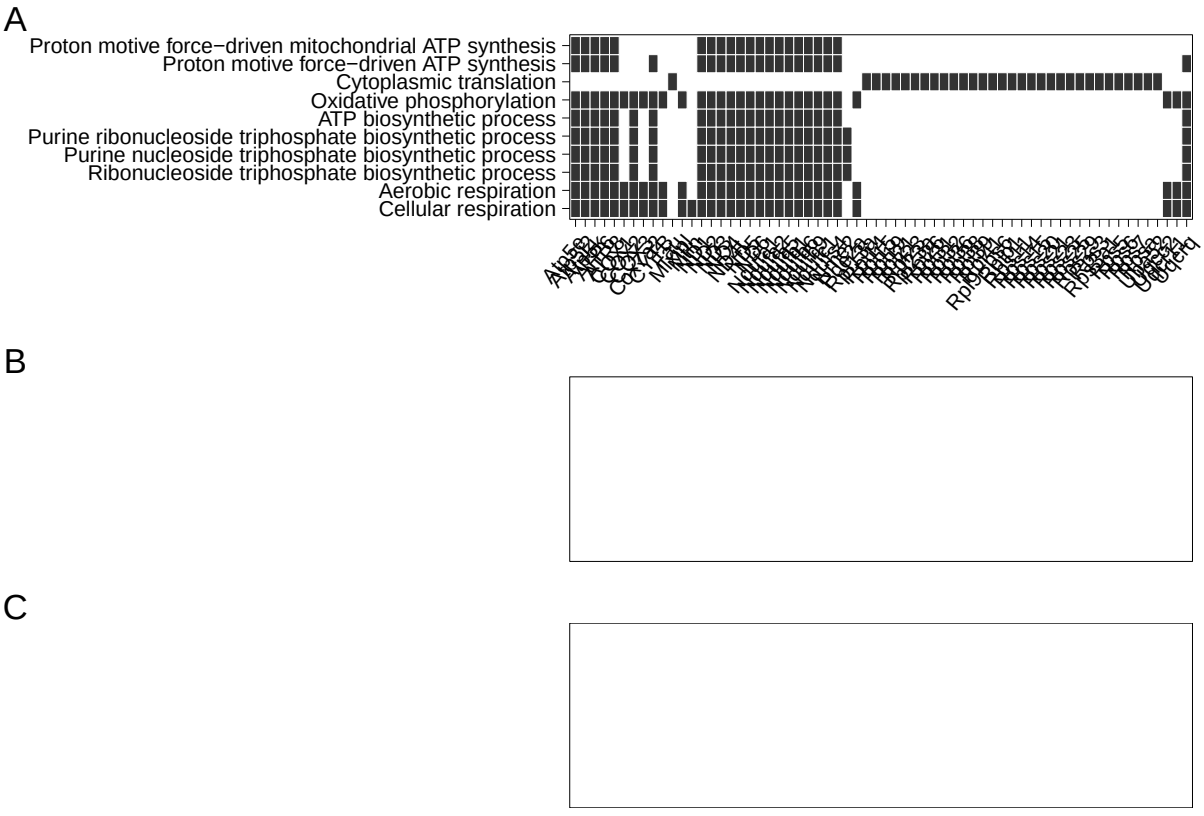



```
pdf(paste("Lollipop.pdf",sep=""))
p1 + p2 + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

Heatmap Plot

```
library(patchwork)
p1 <- plotEnrich(ego3, plot_type = "geneheat")
show_gene = c('malat1','abcc2','abcc2','abcc2')
p2 <- plotEnrich(ego3, plot_type = "geneheat", show_gene = show_gene)
p3 <- plotEnrich(ego3, plot_type = "geneheat", show_gene = show_gene)
p1 / p2 / p3 + plot_annotation(tag_levels = "A")
```



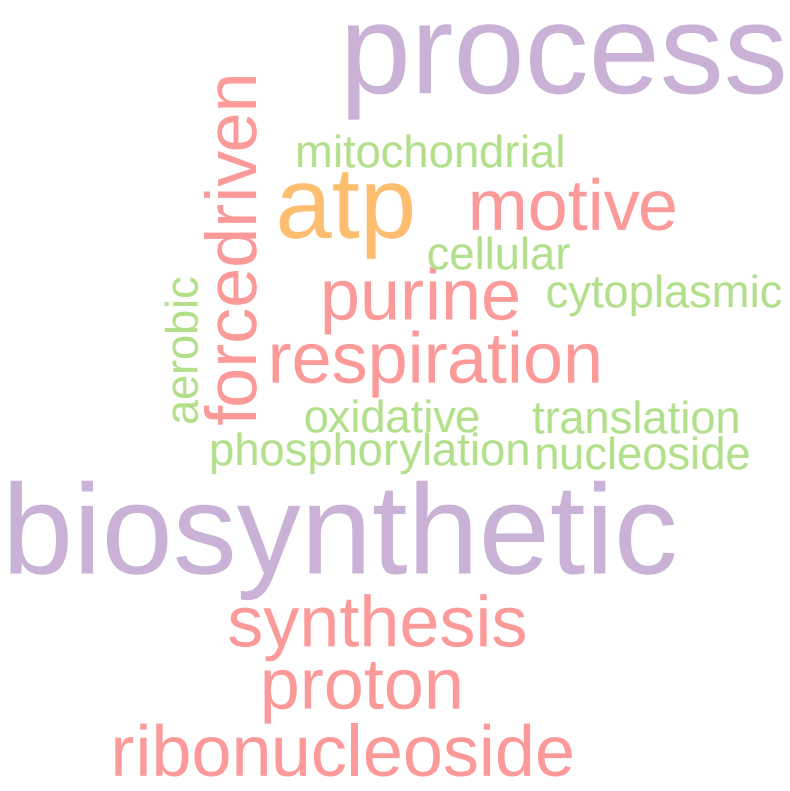

```
pdf(paste("Heatmapplot.pdf",sep=""))
p1  + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

Wordcloud Plot

```
plotEnrich(ego3, plot_type = "wordcloud")
```

```
## Warning in wordcloud::wordcloud(d$word, d$freq, min.freq = 0, colors = mypal):
## triphosphate could not be fit on page. It will not be plotted.
```




```
## NULL
```

```
pdf(paste("Wordcloudplot.pdf",sep=""))
plotEnrich(ego3, plot_type = "wordcloud")
```

```
## NULL
```

```
dev.off()
```

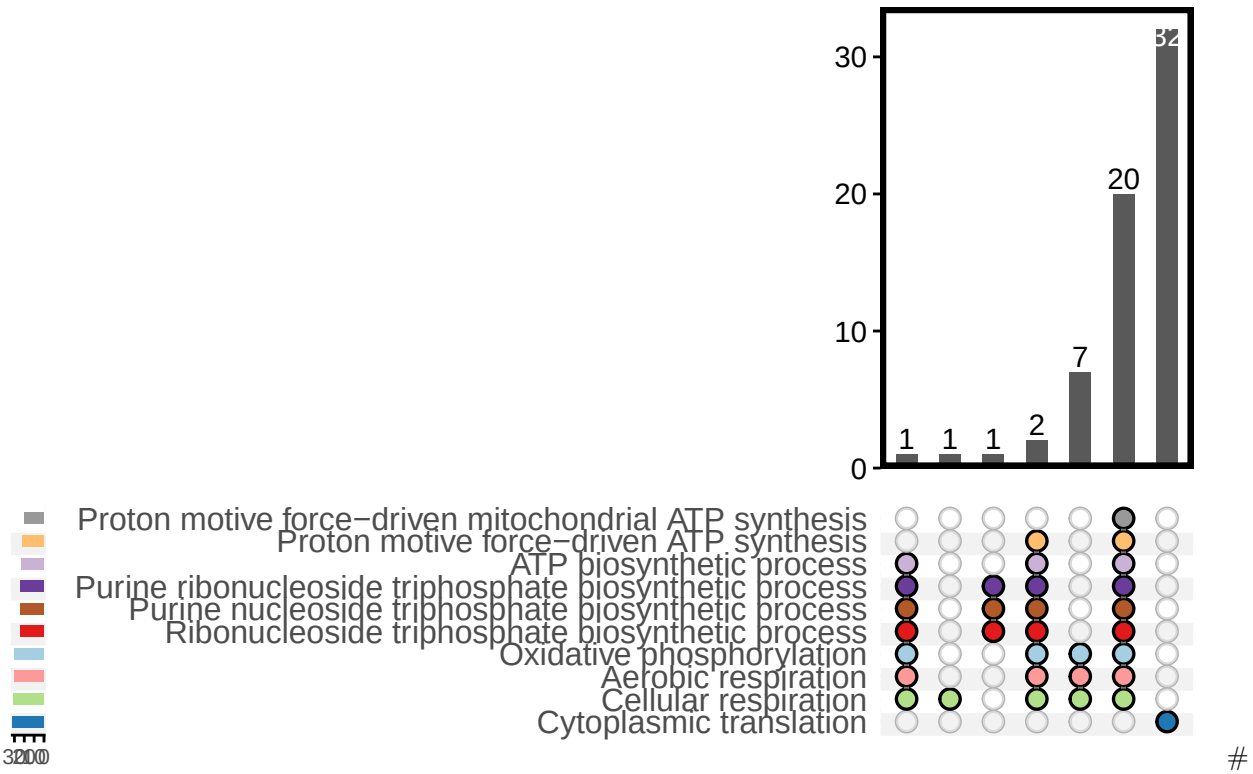
```
## pdf
##   2
```

Upset Plot

```
pdf(paste("Upsetplot.pdf",sep=""))
plotEnrich(ego3, plot_type = "upset",main_text_size = 15,legend_text_size = 8)
dev.off()
```

```
## pdf
##   2
```

```
plotEnrich(ego3, plot_type = "upset",main_text_size = 15,legend_text_size = 8)
```

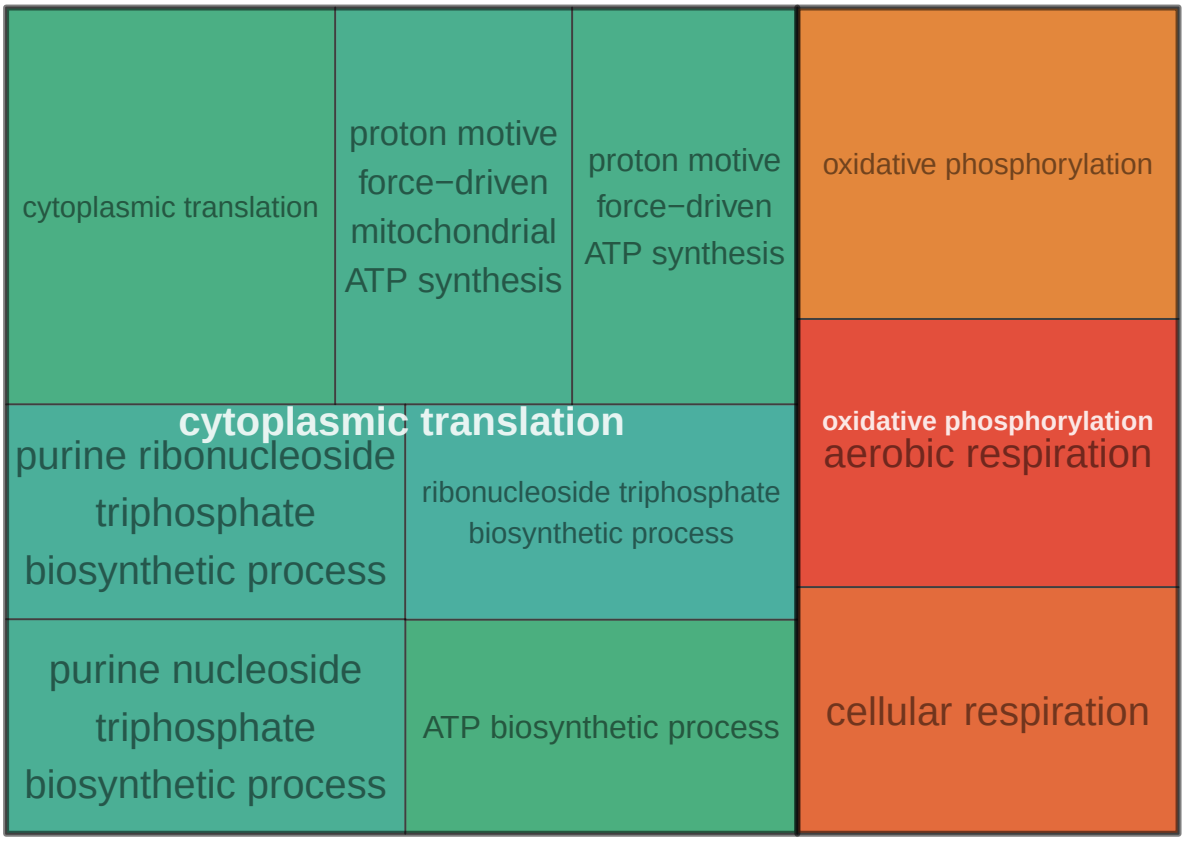


#

GO-specific: Terms Tangram Plot

```
plotEnrich(ego3, plot_type = "gotangram",main_text_size = 15,legend_text_size = 8, scale_ratio = 0.5, si
```

```
## Warning in RColorBrewer::brewer.pal(length(unique(reducedTerms$parent)), : minimal value for n is 3,
```




```
#"bar", "wego", "dot", "bubble", "lollipop", "geneheat", "genechord", "network", "gomap", "goheat", "go

pdf(paste("gotangram.pdf",sep=""))
plotEnrich(ego3, plot_type = "gotangram",main_text_size = 15,legend_text_size = 8, scale_ratio = 0.5, s
```

```
## Warning in RColorBrewer::brewer.pal(length(unique(reducedTerms$parent)), : minimal value for n is 3,
```

```
dev.off()
```

```
## pdf
##   2
```

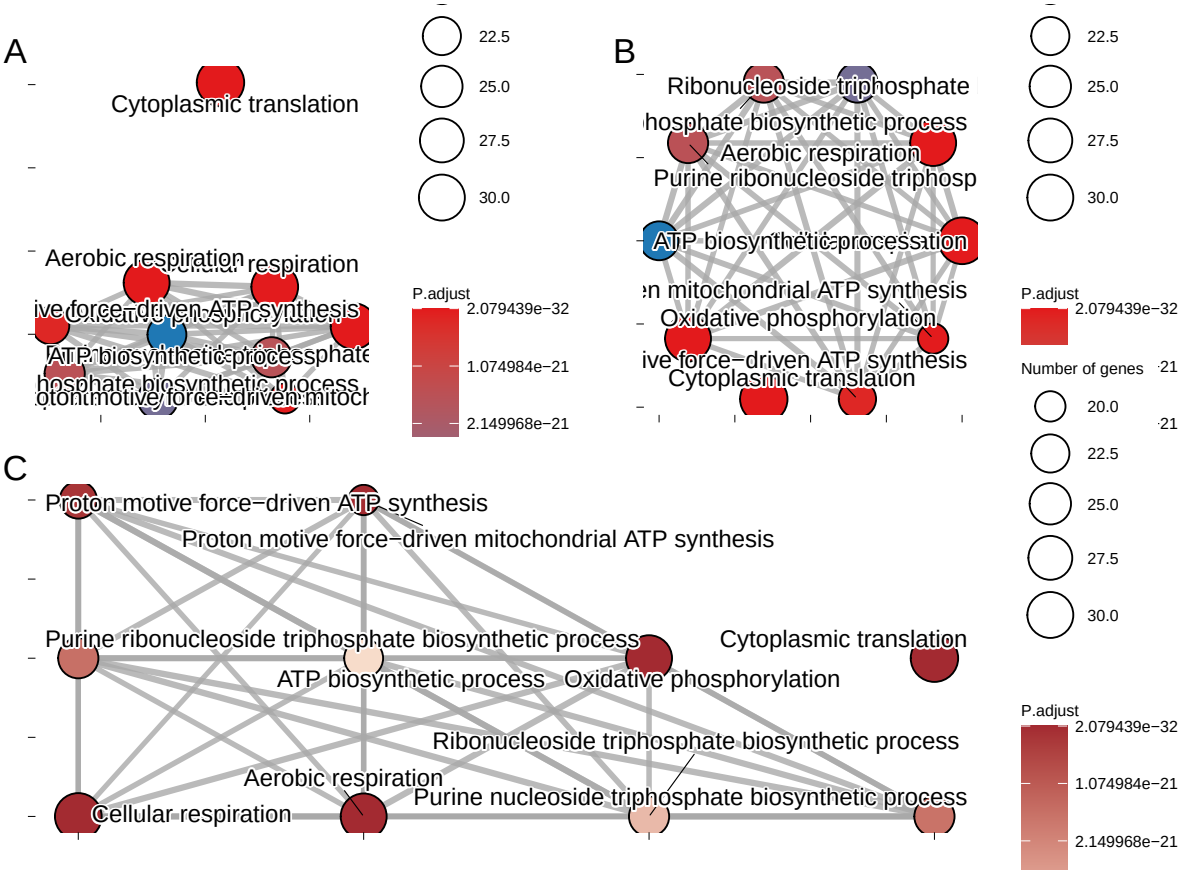
Network Plot

```
library(patchwork)
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following object is masked from 'package:IRanges':
##
##     union

## The following object is masked from 'package:S4Vectors':
##
##     union

## The following objects are masked from 'package:BiocGenerics':
##
##     normalize, path, union

## The following object is masked from 'package:clusterProfiler':
##
##     simplify

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(ggraph)
p1 <- plotEnrich(ego3, plot_type = "network", scale_ratio = 0.5)
p2 <- plotEnrich(ego3, plot_type = "network",
                 layout = "circle", scale_ratio = 0.5)
p3 <- plotEnrich(ego3, plot_type = "network",
                 layout = "grid", sim_method = "Wang",
                 up_color = "#a32a31", down_color = "#f7dcca",
                 scale_ratio = 0.5)
(p1 + p2) / p3 + plot_annotation(tag_levels = "A")
```




```
pdf(paste("Network.pdf",sep=""))
(p1 + p2) / p3 + plot_annotation(tag_levels = "A")
dev.off()
```

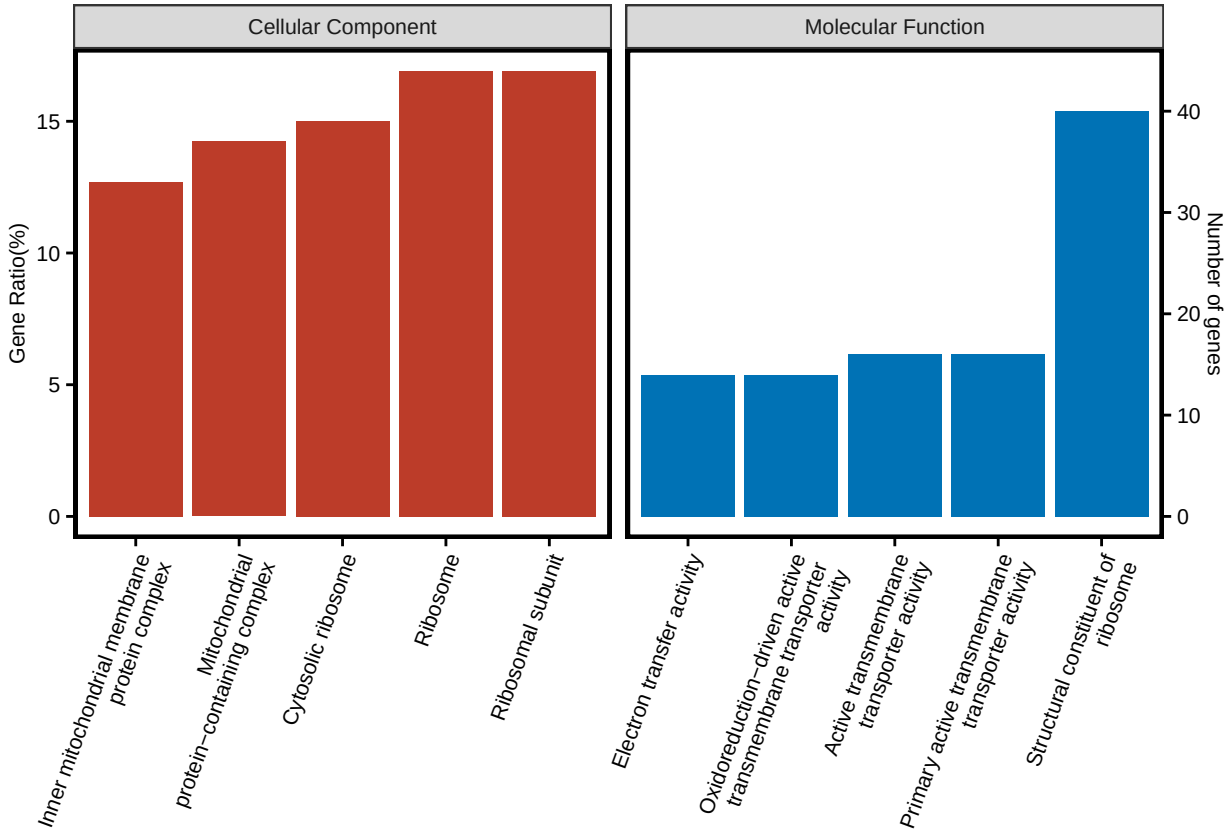
```
## pdf
##   2
```

GO-specific: WEGO Plot

```
# 1st step: prepare input IDs
id<-selectedEntrezsUP

# 2nd step: prepare CC and MF gene sets
go_cc <- geneset::getGO(org = "mouse",ont = "cc")
go_mf <- geneset::getGO(org = "mouse",ont = "mf")


# 3rd step: analysis
ego_cc <- genORA(id, geneset = go_cc)
ego_mf <- genORA(id, geneset = go_mf)


# 4th step: merge two data frames
```

```
# Note: each data frame should add new column "Ontology"
ego_cc <- ego_cc %>% dplyr::mutate(Ontology = "cc") %>% dplyr::rename(ID = 1)
ego_mf <- ego_mf %>% dplyr::mutate(Ontology = "mf") %>% dplyr::rename(ID = 1)

all_ego <- rbind(ego_cc,ego_mf)
```

```
plotEnrich(all_ego, plot_type = "wego", n_term = 5)
```



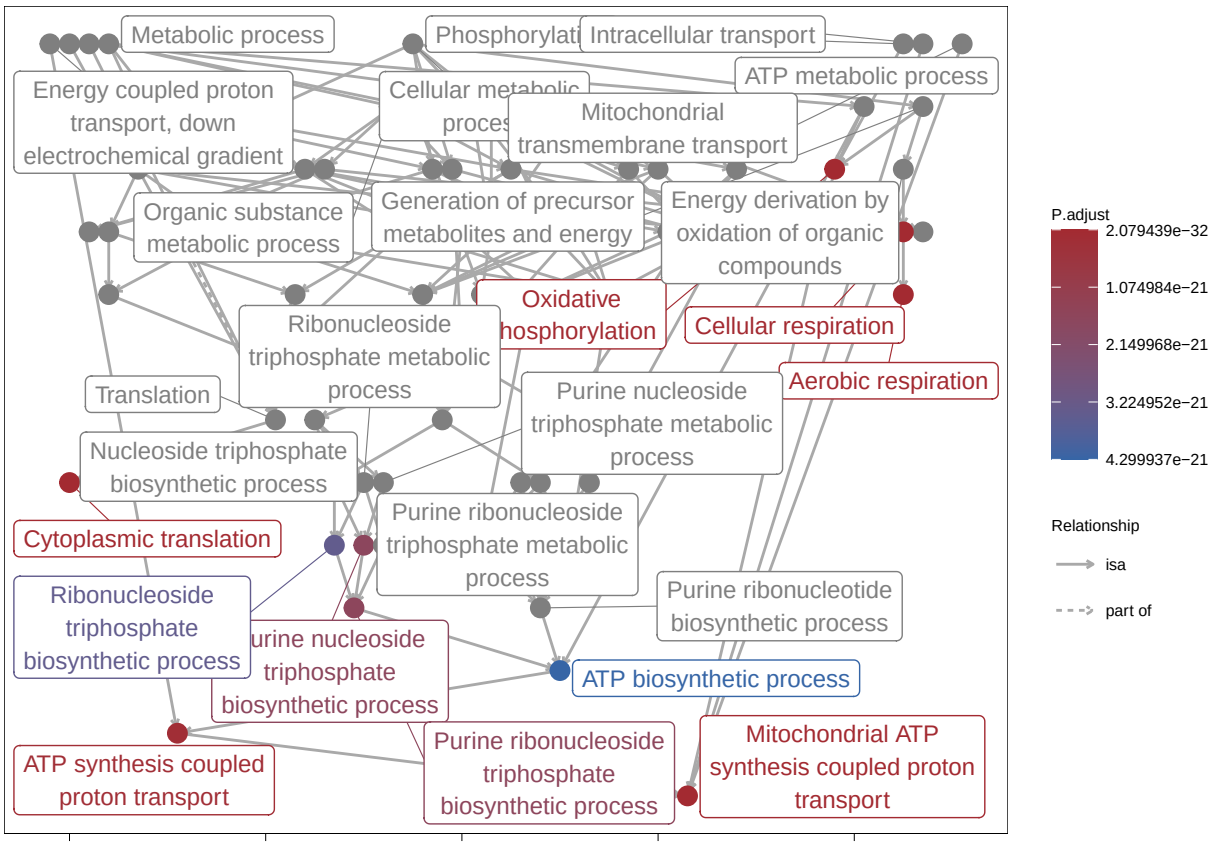

```
pdf(paste("WEGOPlot.pdf",sep=""))
plotEnrich(all_ego, plot_type = "wego", n_term = 5)
dev.off()
```

```
## pdf
##   2
```

GO-specific: Map Plot

```
library(igraph)
library(ggraph)
plotEnrich(ego3, plot_type = "gomap", wrap_length = 25,
           up_color = '#a32a31',down_color = '#3665a6')
```

```
## Warning: Removed 49 rows containing missing values (`geom_label_repel()`).
```




```
pdf(paste("MapPlot.pdf",sep=""))
plotEnrich(ego3, plot_type = "gomap", wrap_length = 25,
           up_color = '#a32a31',down_color = '#3665a6')
```

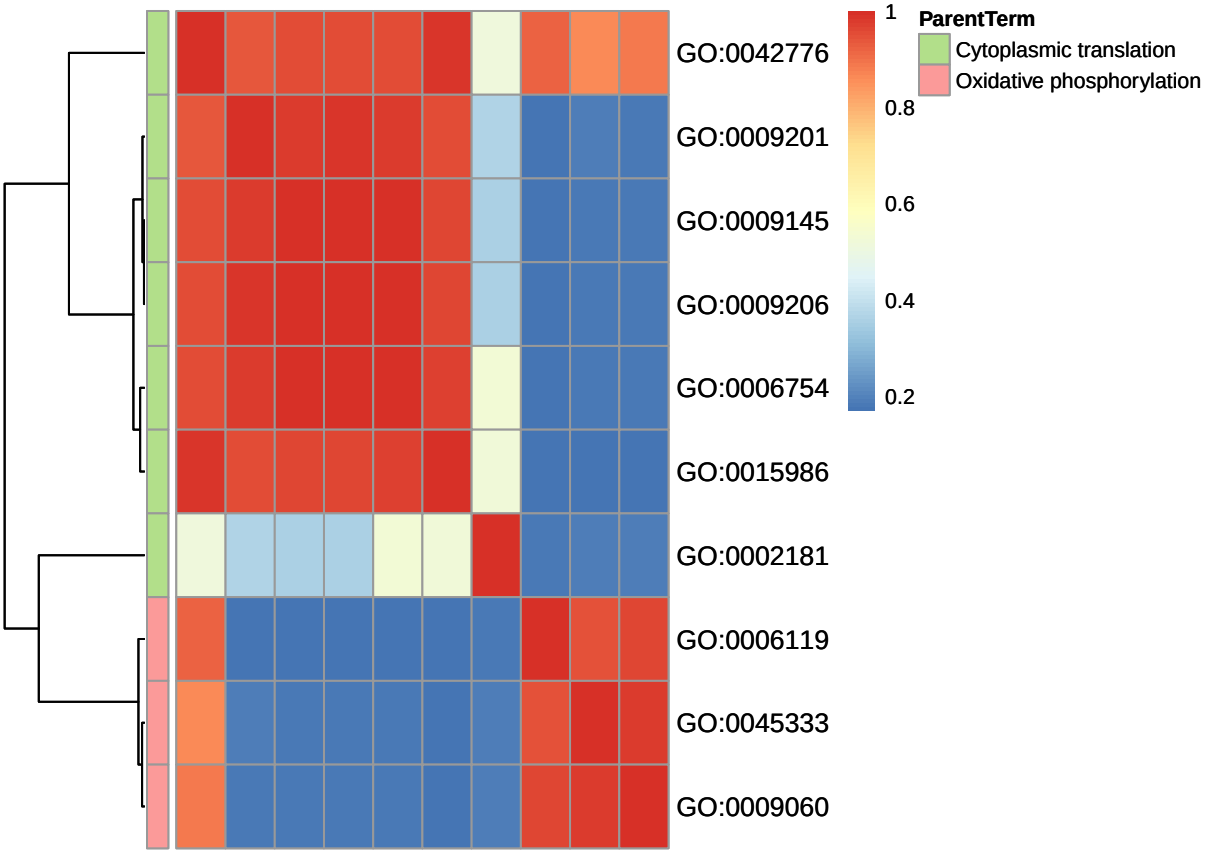
```
## Warning: Removed 49 rows containing missing values (‘geom_label_repel()‘).
```

```
dev.off()
```

```
## pdf
##   2
```

GO-specific: Terms Heatmap Plot

```
plotEnrich(ego3, plot_type = "goheat", sim_method = "Rel")
```



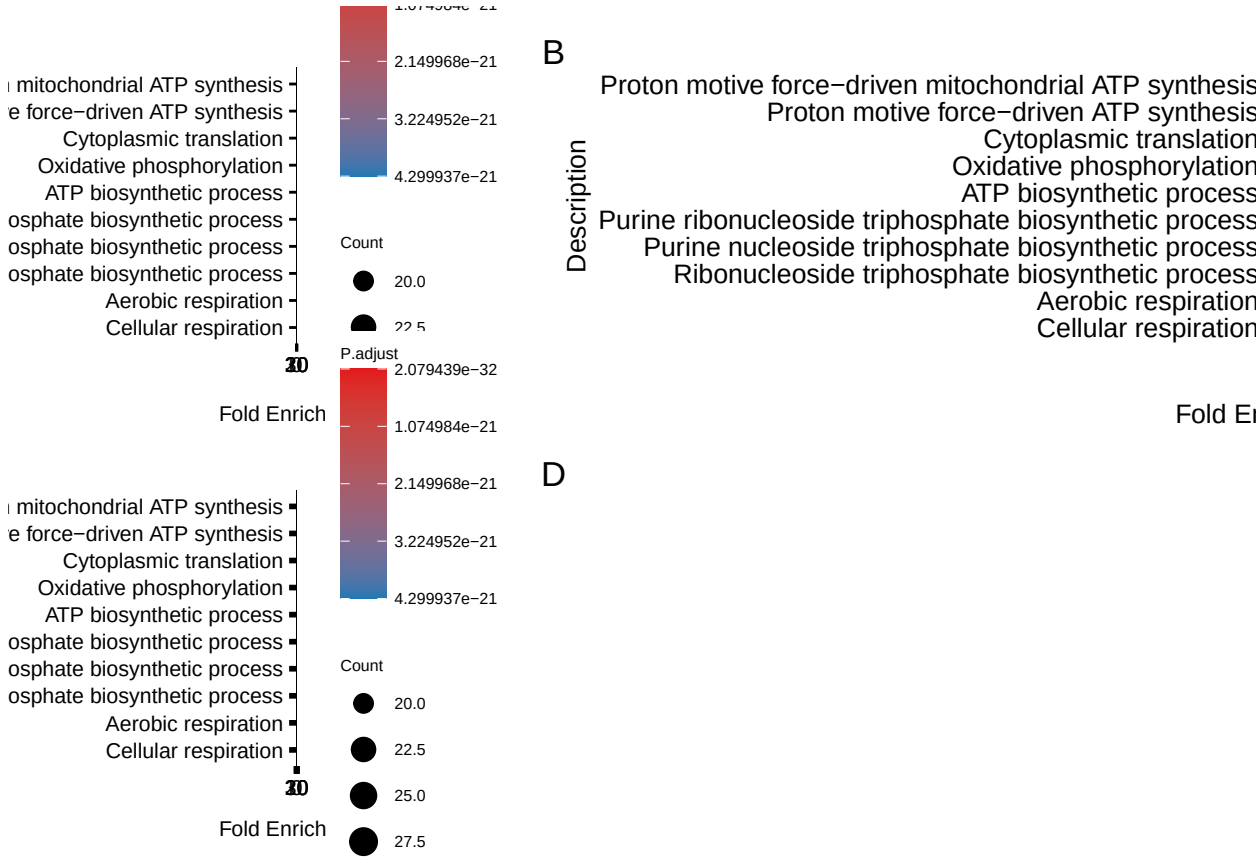

```
pdf(paste("TermsHeatmap.pdf",sep=""))
plotEnrich(ego3, plot_type = "goheat", sim_method = "Rel")
dev.off()
```

```
## pdf
##   3
```

Plot Theme

```
library(patchwork)
p1 <- plotEnrich(ego3, plot_type = "dot")
p2 <- plotEnrich(ego3,
  plot_type = "dot",
  main_text_size = 10,
  legend_text_size = 10
)

p3 <- plotEnrich(ego3,
  plot_type = "dot",
  border_thick = 3,
  remove_grid = F
)
```

```
p4 <- plotEnrich(ego3,
  plot_type = "dot",
  remove_main_text = T,
  remove_legend_text = T,
  remove_legend = T
)
p1 + p2 + p3 + p4 + plot_annotation(tag_levels = "A")
```




```
pdf(paste("PlotTheme.pdf",sep=""))
p1 + p2 + p3 + p4 + plot_annotation(tag_levels = "A")
dev.off()
```

```
## pdf
##   2
```

Advanced Plot

```
# 1st step: prepare input IDs
# Since the geneList is logFC decreasing ordered, we could take first 100 as up-regulated genes and vic

up_genes <-rownames(subset(top, logFC > 8 & (adj.P.Val < 0.01)))
down_genes <- rownames(subset(top, logFC < -3 & (adj.P.Val < 0.01)))
```

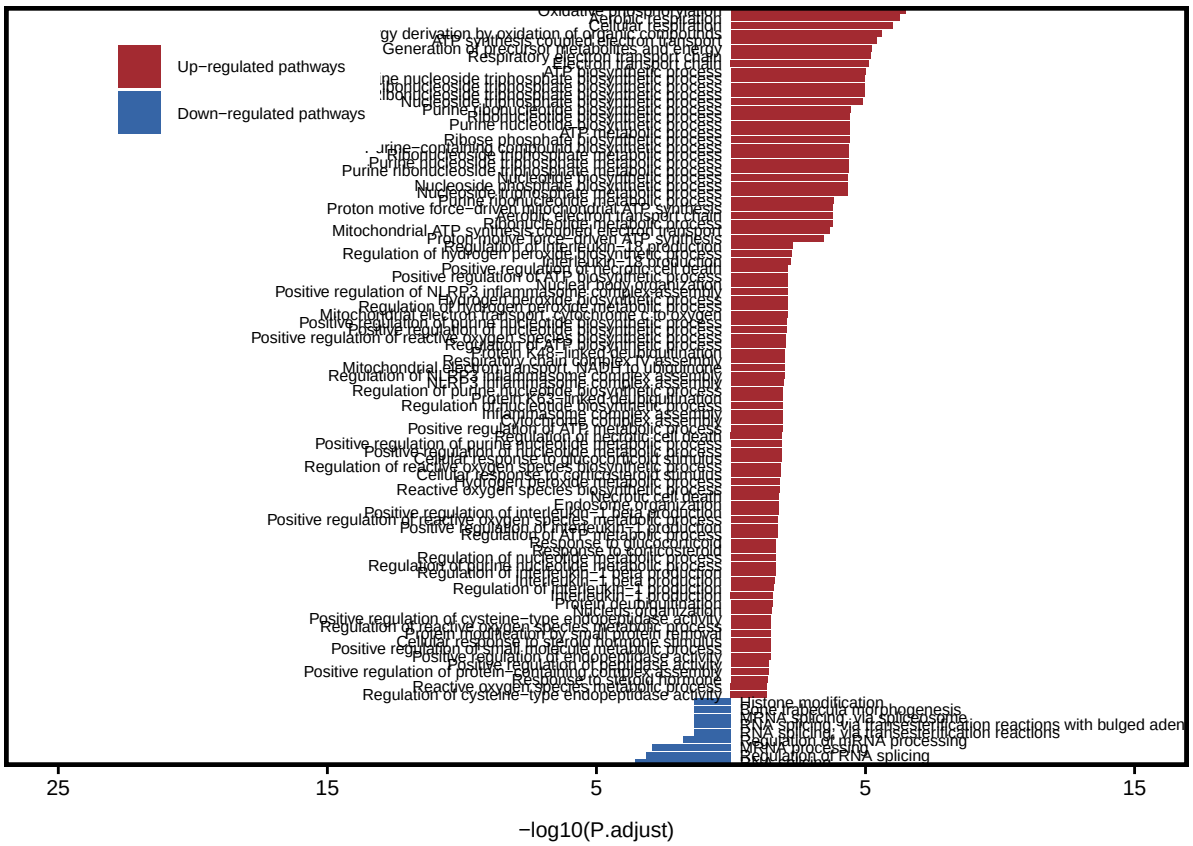
```
# 2nd step: prepare gene set
mm_gs <- geneset::getGO(org = "mouse",ont = "bp")

# 3rd step: ORA analysis separately
up_go <- genORA(up_genes, geneset = mm_gs)
down_go <- genORA(down_genes, geneset = mm_gs)

dim(up_go)
```

```
## [1] 91 12
```

```
dim(down_go)
```

```
## [1]  9 12
```

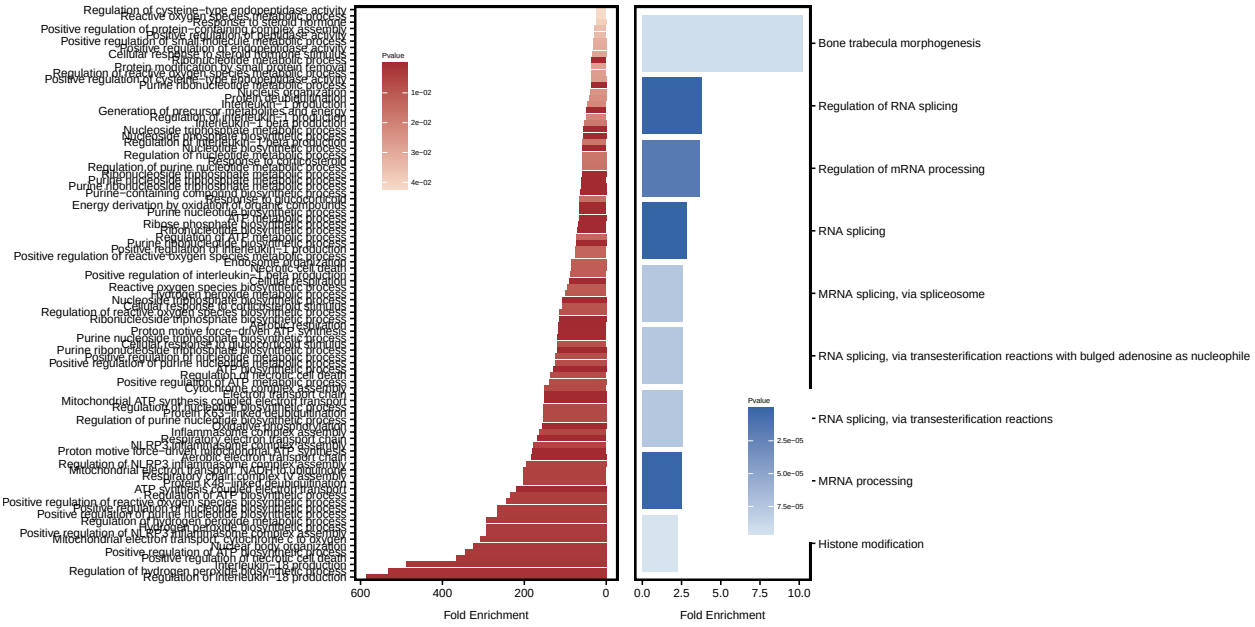
```
plotEnrichAdv(up_go, down_go,
              plot_type = "one",
              term_metric = "FoldEnrich",
              stats_metric = "p.adjust",
              xlim_left = 25, xlim_right = 15) +
  theme(legend.position = c(0.2, 0.9))
```



```
pdf(paste("Advanced.pdf",sep=""))
plotEnrichAdv(up_go, down_go,
              plot_type = "one",
              term_metric = "FoldEnrich",
              stats_metric = "p.adjust",
              xlim_left = 25, xlim_right = 15) +
  theme(legend.position = c(0.2, 0.9))
dev.off()
```

```
## pdf
##   2
```

```
plotEnrichAdv(up_go, down_go,
              plot_type = "two",
              term_metric = "FoldEnrich",
              stats_metric = "pvalue",
              legend_text_size = 5
) +
  theme(legend.position = "none")
```




```
pdf(paste("Advanced2.pdf",sep=""))
plotEnrichAdv(up_go, down_go,
              plot_type = "two",
              term_metric = "FoldEnrich",
              stats_metric = "pvalue",
              legend_text_size = 5
) +
  theme(legend.position = "none")
dev.off()
```

```
## pdf
##   2
```

Subcellular RNA fractions have a different transcript composition.Scatter plot and correlation matrix of all sequenced samples. The color intensity of the correlation boxes (r values) depicts the relative strength of the correlation, ranging between 0.39 and 0.97.

(B) RNA species content of each sequenced fraction in counts per million. CPM, counts per million; lincRNA, long intergenic noncoding RNA; snoRNA, small nucleolar RNA; snRNA, small nuclear RNA

```
biotypes<- read_delim("./multiqc_featurecounts_biotype_plot.txt")
```

```
## Rows: 120 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): Sample
## dbl (37): TEC, snRNA, protein_coding, processed_pseudogene, lncRNA, miRNA, s...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
biotypes<-as.data.frame(biotypes)
biotypes<-biotypes[-c(1:36, 61:120),]
genomic_origin<- read_delim("./mqc_qualimap_genomic_origin_1.txt")
```

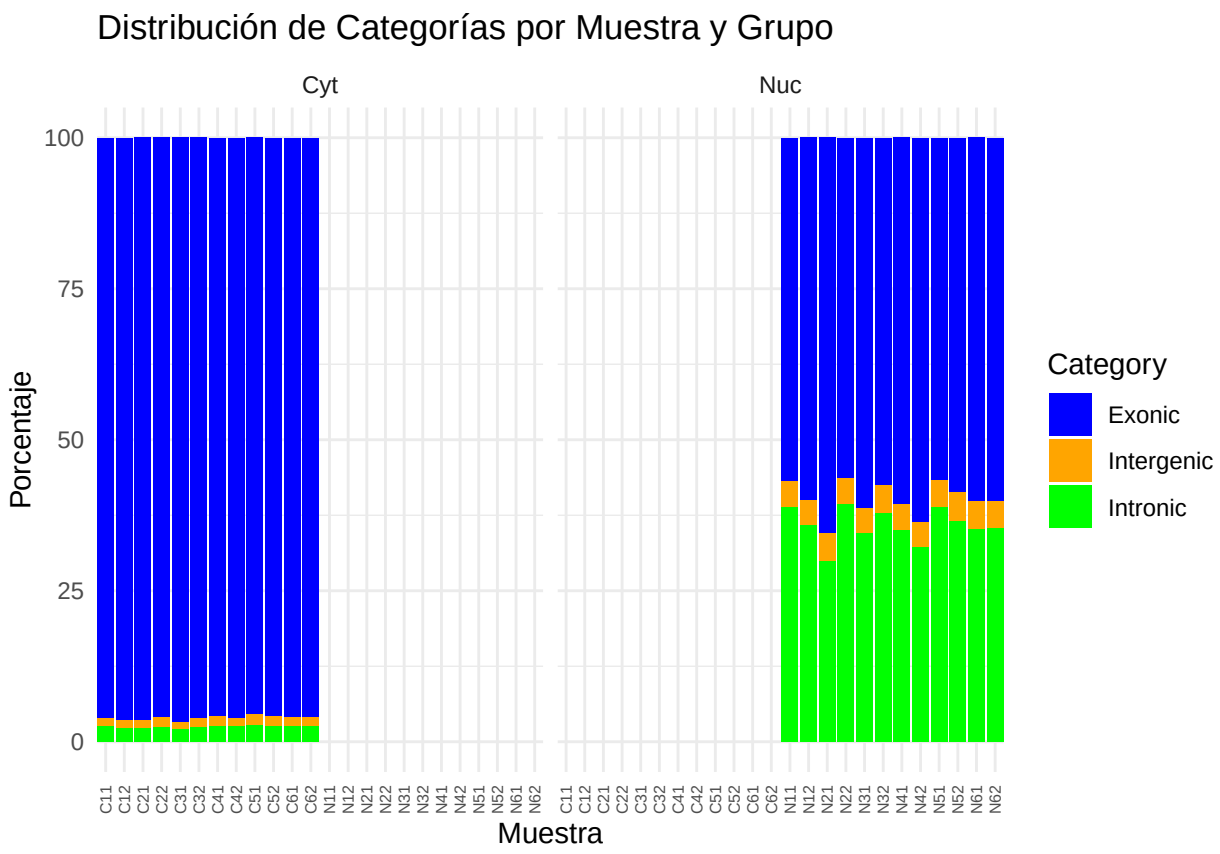
```
## Rows: 120 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): Sample
## dbl (3): Exonic, Intronic, Intergenic
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
genomic_origin<- as.data.frame(genomic_origin)
genomic_origin<-genomic_origin[-c(1:36, 61:120),]

merge_origin<-merge(x=info, y=genomic_origin, by.x = "Run", by.y = "Sample")
merge_origin<-merge_origin[,-c(2:11)]

merge_biotypes<-merge(x=info, y=biotypes, by.x = "Run", by.y = "Sample")
merge_biotypes<-merge_biotypes[,-c(2:11)]
```

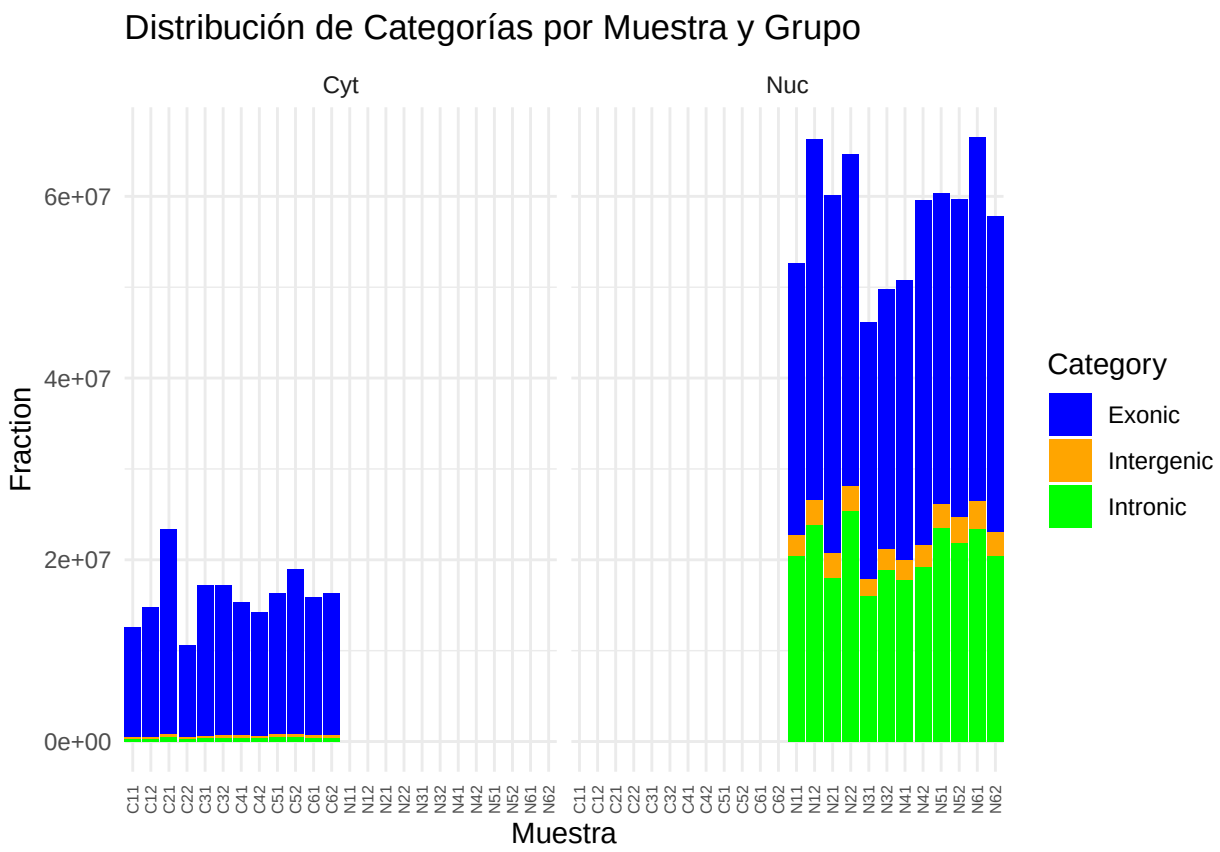
Distribució segons genomic origin

```
library(tidyr)
library(dplyr)
merge_origin_long <- gather(merge_origin, Category, Fraction, Exonic, Intronic, Intergenic)

merge_origin_long <- merge_origin_long %>%
  group_by(Sample) %>%
  mutate(Percentage = (Fraction / sum(Fraction)) * 100)
```

```r
# Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nucle
ggplot(merge_origin_long, aes(x = Sample, y = Percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Group, nrow = 1) +  # Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c("Exonic" = "blue", "Intronic" = "green", "Intergenic" = "orange")) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6))
```




```r
# Crear un gráfico de barras apiladas con colores diferentes para cada categoría
ggplot(merge_origin_long, aes(x = Sample, y = Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_grid(. ~ Group) +
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Fraction") +
  scale_fill_manual(values = c("Exonic" = "blue", "Intronic" = "green", "Intergenic" = "orange")) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6))
```

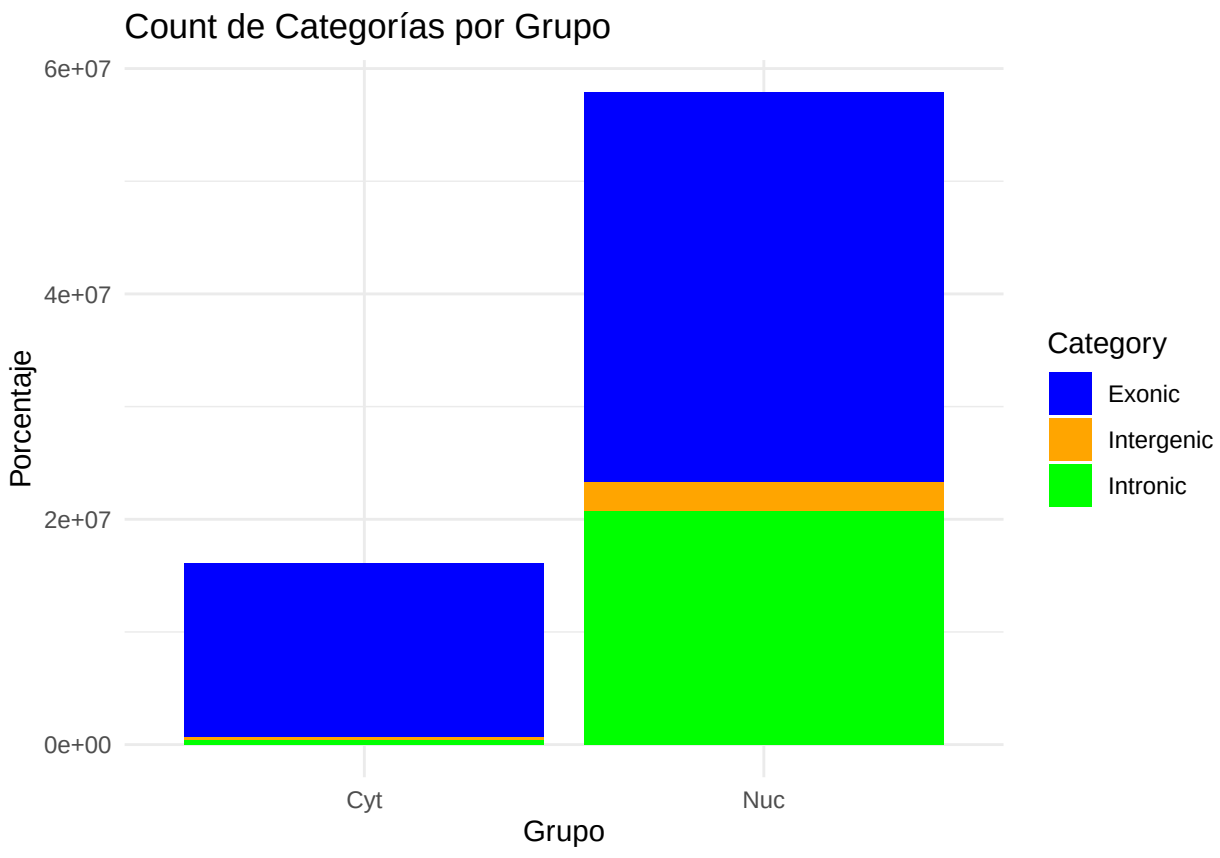



```r
# Crear una lista de data frames, uno por cada grupo (Cytoplasmic y Nuclear)
groups <- split(merge_origin_long, merge_origin_long$Group)

# Calcular la media para cada grupo
mean_data <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_percentage = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Percentage[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data <- do.call(rbind, mean_data)

pdf(paste("Grafic1.pdf",sep=""))
# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data, aes(x = Group, y = mean_percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c("Exonic" = "blue", "Intronic" = "green", "Intergenic" = "orange")) +
  theme_minimal()+
  theme(
    legend.text = element_text(size = 40),      # Tamaño del texto de la leyenda
    legend.title = element_text(size = 40)  )    # Tamaño del título de la leyenda
```

```r
dev.off()
```

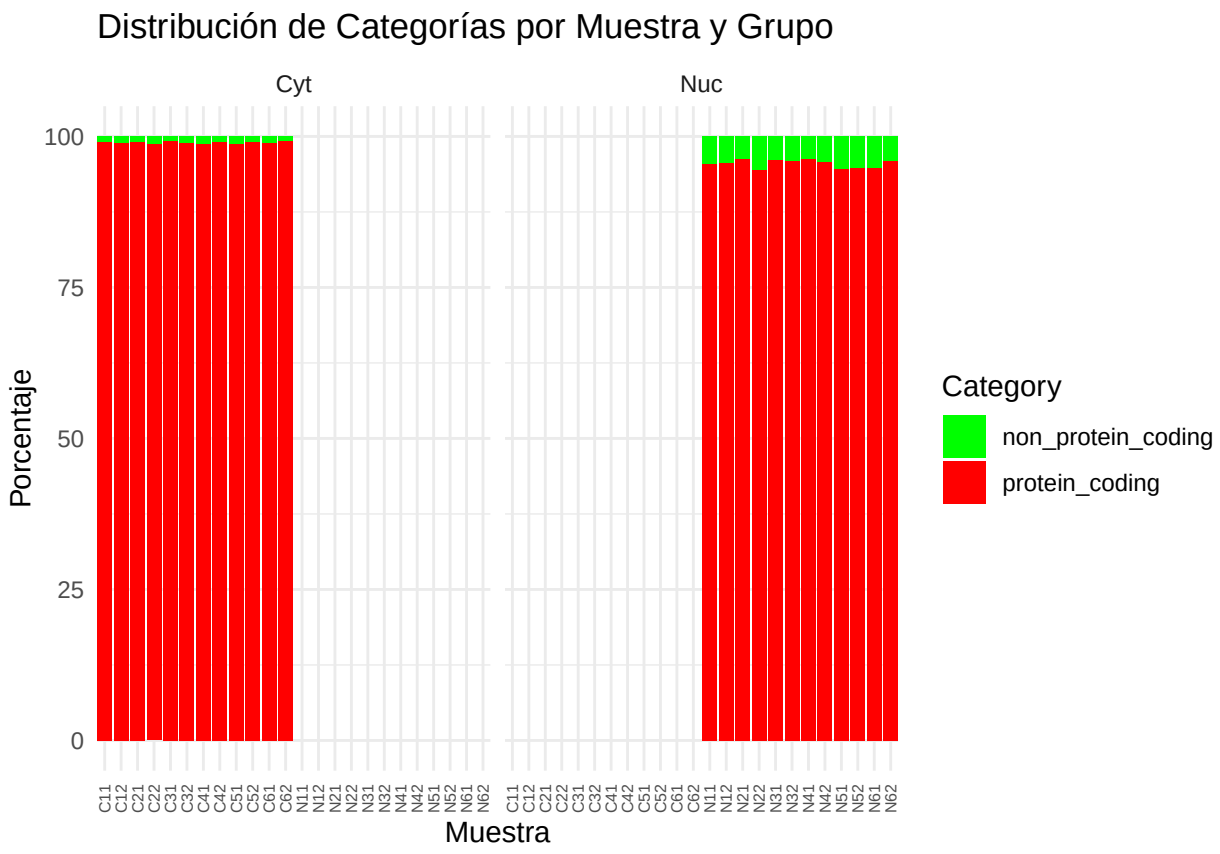
```
## pdf
##   2
```

```r
# Calcular la media para cada grupo
mean_data2 <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_Fraction = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Fraction[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data2 <- do.call(rbind, mean_data2)

# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data2, aes(x = Group, y = mean_Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Count de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c("Exonic" = "blue", "Intronic" = "green", "Intergenic" = "orange")) +
  theme_minimal()
```



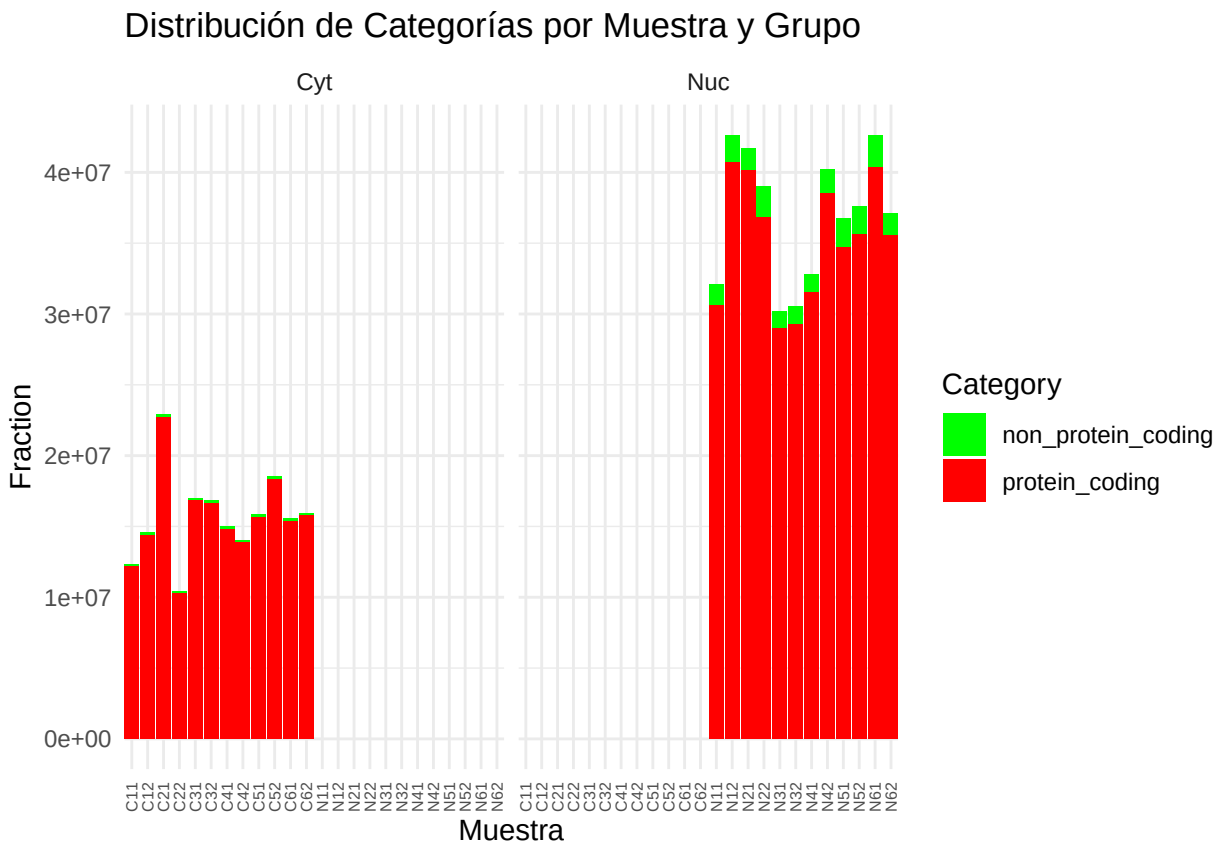
Distribució sengons sengons biotypes (protein_coding vs non_protein_coding (la resta))

```
merge_biotypes$non_protein_coding <- rowSums(merge_biotypes[, c(7:8, 10:43)], na.rm = TRUE)
merge_biotypes_long_simple<- gather(merge_biotypes, Category, Fraction,  protein_coding ,non_protein_co
merge_biotypes_long_simple <- merge_biotypes_long_simple %>%
  group_by(Sample) %>%
  mutate(Percentage = (Fraction / sum(Fraction)) * 100)


# Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nucle
ggplot(merge_biotypes_long_simple, aes(x = Sample, y = Percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Group, nrow = 1) +  # Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c("protein_coding" = "#FF0000", "non_protein_coding" = "green")) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6))
```




```
# Crear un gráfico de barras apiladas con colores diferentes para cada categoría
ggplot(merge_biotypes_long_simple, aes(x = Sample, y = Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_grid(. ~ Group) +
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Fraction") +
```

```
  scale_fill_manual(values = c("protein_coding" = "#FF0000", "non_protein_coding" = "green")) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6))
```

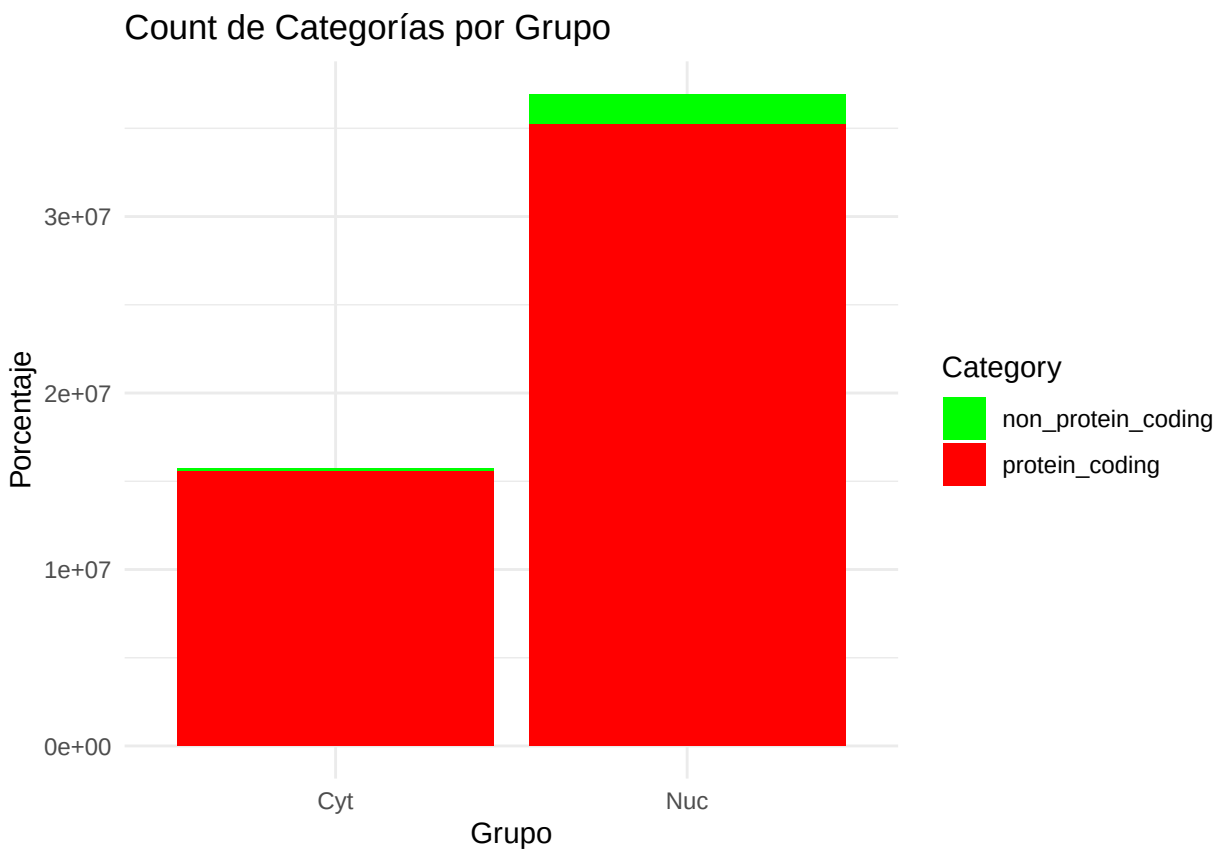



```
groups <- split(merge_biotypes_long_simple, merge_biotypes_long_simple$Group)

# Calcular la media para cada grupo
mean_data <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_percentage = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Percentage[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data <- do.call(rbind, mean_data)

pdf(paste("Grafic2.pdf",sep=""))
# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data, aes(x = Group, y = mean_percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Media de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
```

```
  scale_fill_manual(values = c("protein_coding" = "#FF0000", "non_protein_coding" = "green")) +
  theme_minimal()
dev.off()
```

```
## pdf
##   2
```

```
# Calcular la media para cada grupo
mean_data2 <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_fraction = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Fraction[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data2 <- do.call(rbind, mean_data2)


# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data2, aes(x = Group, y = mean_fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Count de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c("protein_coding" = "#FF0000", "non_protein_coding" = "green")) +
  theme_minimal()
```



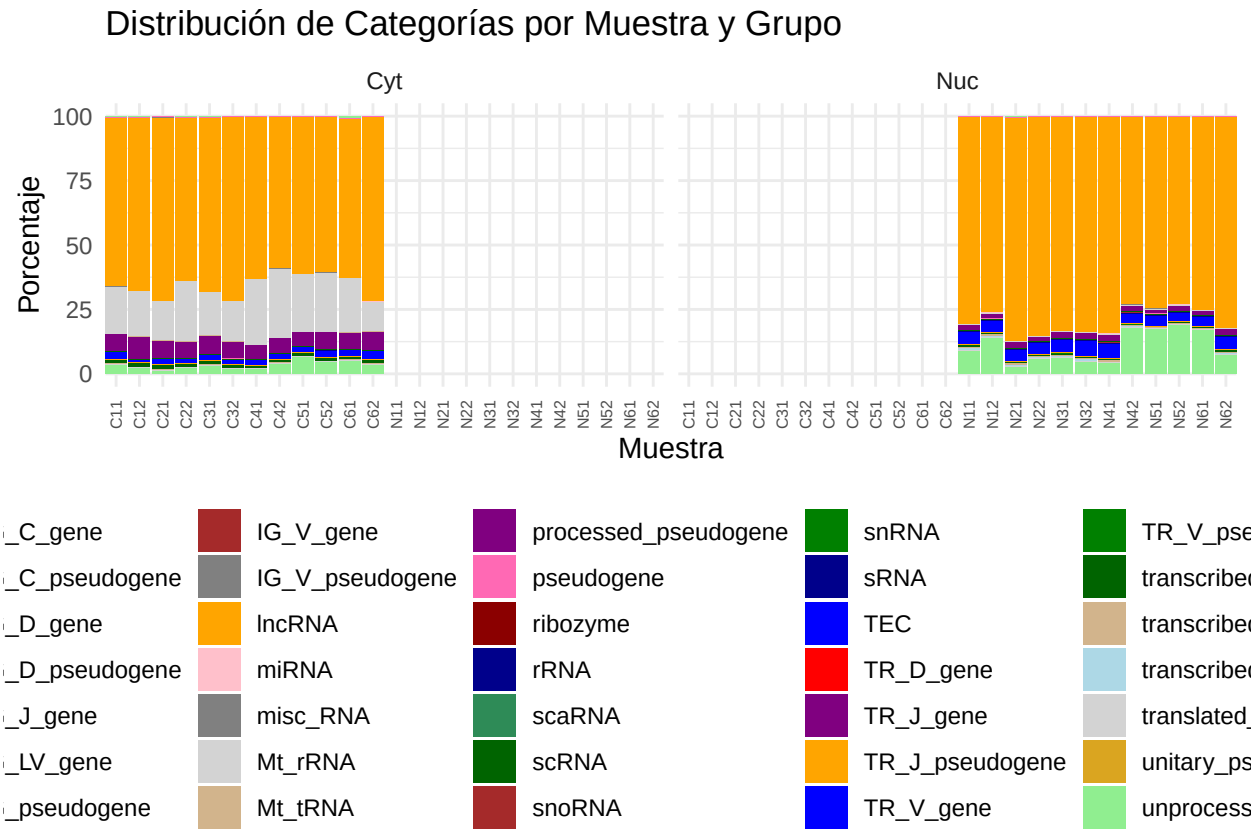
#

Distribució sengons biotypes (protein_coding exclós)

```
merge_biotypes_long <- gather(merge_biotypes, Category, Fraction, TEC , snRNA  ,processed_pseudogene ,lr

# protein_coding
merge_biotypes_long <- merge_biotypes_long %>%
  group_by(Sample) %>%
  mutate(Percentage = (Fraction / sum(Fraction)) * 100)
```

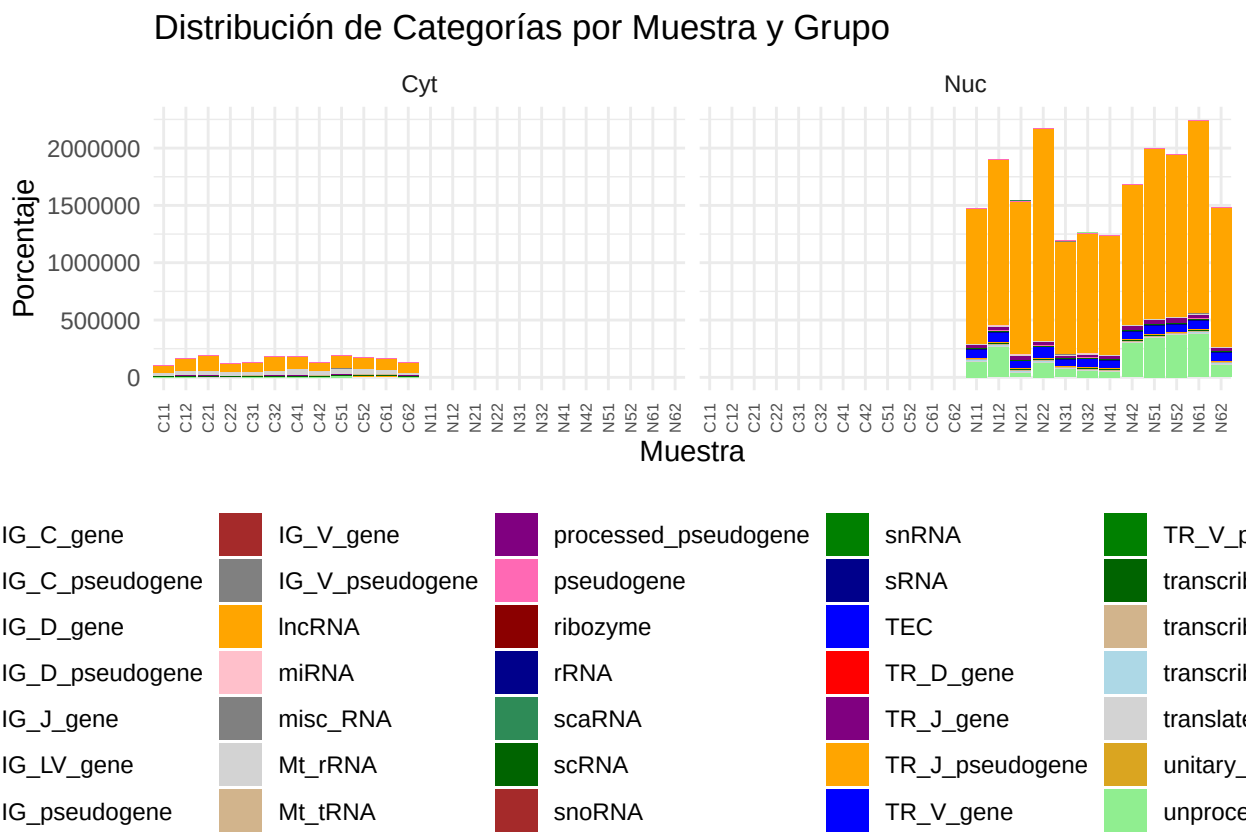
```
# Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nucle
ggplot(merge_biotypes_long, aes(x = Sample, y = Percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Group, nrow = 1) +  # Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "protein_coding" = "#0FBBF0",
    "TEC" = "#0000FF",  # Azul
    "snRNA" = "#008000",  # Verde
    "processed_pseudogene" = "#800080",  # Morado
    "lncRNA" = "#FFA500",  # Naranja
    "miRNA" = "#FFC0CB",  # Rosa
    "snoRNA" = "#A52A2A",  # Marrón
    "misc_RNA" = "#808080",  # Gris
    "transcribed_unprocessed_pseudogene" = "#ADD8E6",  # Azul claro
    "unprocessed_pseudogene" = "#90EE90",  # Verde claro
    "rRNA" = "#00008B",  # Azul oscuro
    "transcribed_processed_pseudogene" = "#006400",  # Verde oscuro
```

```
    "ribozyme" = "#8B0000",  # Rojo oscuro
    "unitary_pseudogene" = "#DAA520",  # Amarillo
    "scaRNA" = "#2E8B57",  # Verde marino
    "pseudogene" = "#FF69B4",  # Rosa claro
    "transcribed_unitary_pseudogene" = "#D2B48C",  # Marrón claro
    "translated_unprocessed_pseudogene" = "#D3D3D3",  # Gris claro
    "TR_V_gene" = "#0000FF",
    "TR_V_pseudogene" = "#008000",
    "TR_D_gene" = "#FF0000",
    "TR_J_gene" = "#800080",
    "TR_J_pseudogene" = "#FFA500",
    "IG_LV_gene" = "#FFC0CB",
    "IG_V_gene" = "#A52A2A",
    "IG_V_pseudogene" = "#808080",
    "IG_J_gene" = "#ADD8E6",
    "IG_C_gene" = "#90EE90",
    "sRNA" = "#00008B",
    "scRNA" = "#006400",
    "IG_C_pseudogene" = "#8B0000",
    "IG_D_gene" = "#DAA520",
    "IG_D_pseudogene" = "#2E8B57",
    "IG_pseudogene" = "#FF69B4",
    "Mt_tRNA" = "#D2B48C",
    "Mt_rRNA" = "#D3D3D3"
  )) +
 theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```




```
# Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nucle
ggplot(merge_biotypes_long, aes(x = Sample, y = Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_grid(. ~ Group) +
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "protein_coding" = "#0FBBF0",
    "TEC" = "#0000FF",  # Azul
    "snRNA" = "#008000",  # Verde
    "processed_pseudogene" = "#800080",  # Morado
    "lncRNA" = "#FFA500",  # Naranja
    "miRNA" = "#FFC0CB",  # Rosa
    "snoRNA" = "#A52A2A",  # Marrón
    "misc_RNA" = "#808080",  # Gris
    "transcribed_unprocessed_pseudogene" = "#ADD8E6",  # Azul claro
    "unprocessed_pseudogene" = "#90EE90",  # Verde claro
    "rRNA" = "#00008B",  # Azul oscuro
    "transcribed_processed_pseudogene" = "#006400",  # Verde oscuro
    "ribozyme" = "#8B0000",  # Rojo oscuro
    "unitary_pseudogene" = "#DAA520",  # Amarillo
    "scaRNA" = "#2E8B57",  # Verde marino
    "pseudogene" = "#FF69B4",  # Rosa claro
    "transcribed_unitary_pseudogene" = "#D2B48C",  # Marrón claro
    "translated_unprocessed_pseudogene" = "#D3D3D3",  # Gris claro
    "TR_V_gene" = "#0000FF",
    "TR_V_pseudogene" = "#008000",
    "TR_D_gene" = "#FF0000",
```

```
    "TR_J_gene" = "#800080",
    "TR_J_pseudogene" = "#FFA500",
    "IG_LV_gene" = "#FFC0CB",
    "IG_V_gene" = "#A52A2A",
    "IG_V_pseudogene" = "#808080",
    "IG_J_gene" = "#ADD8E6",
    "IG_C_gene" = "#90EE90",
    "sRNA" = "#00008B",
    "scRNA" = "#006400",
    "IG_C_pseudogene" = "#8B0000",
    "IG_D_gene" = "#DAA520",
    "IG_D_pseudogene" = "#2E8B57",
    "IG_pseudogene" = "#FF69B4",
    "Mt_tRNA" = "#D2B48C",
    "Mt_rRNA" = "#D3D3D3"
  )) +
 theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```



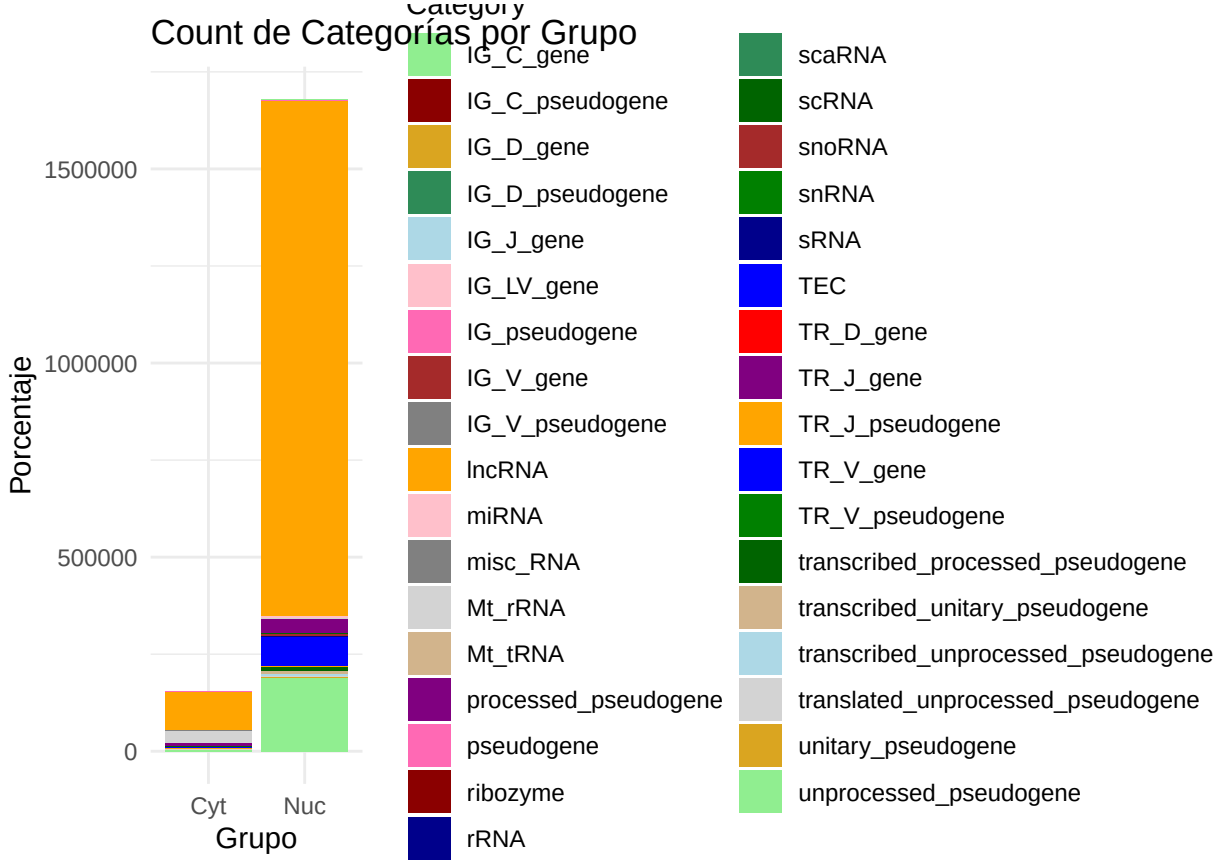
```r
# Crear una lista de data frames, uno por cada grupo (Cytoplasmic y Nuclear)
groups <- split(merge_biotypes_long, merge_biotypes_long$Group)

# Calcular la media para cada grupo
mean_data <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_percentage = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Percentage[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data <- do.call(rbind, mean_data)

pdf(paste("Grafic3.pdf",sep=""))
# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data, aes(x = Group, y = mean_percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "protein_coding" = "#0FBBF0",
    "TEC" = "#0000FF",  # Azul
    "snRNA" = "#008000",  # Verde
    "processed_pseudogene" = "#800080",  # Morado
    "lncRNA" = "#FFA500",  # Naranja
    "miRNA" = "#FFC0CB",  # Rosa
    "snoRNA" = "#A52A2A",  # Marrón
    "misc_RNA" = "#808080",  # Gris
    "transcribed_unprocessed_pseudogene" = "#ADD8E6",  # Azul claro
    "unprocessed_pseudogene" = "#90EE90",  # Verde claro
    "rRNA" = "#00008B",  # Azul oscuro
    "transcribed_processed_pseudogene" = "#006400",  # Verde oscuro
    "ribozyme" = "#8B0000",  # Rojo oscuro
    "unitary_pseudogene" = "#DAA520",  # Amarillo
    "scaRNA" = "#2E8B57",  # Verde marino
    "pseudogene" = "#FF69B4",  # Rosa claro
    "transcribed_unitary_pseudogene" = "#D2B48C",  # Marrón claro
    "translated_unprocessed_pseudogene" = "#D3D3D3",  # Gris claro
    "TR_V_gene" = "#0000FF",
    "TR_V_pseudogene" = "#008000",
    "TR_D_gene" = "#FF0000",
    "TR_J_gene" = "#800080",
    "TR_J_pseudogene" = "#FFA500",
    "IG_LV_gene" = "#FFC0CB",
    "IG_V_gene" = "#A52A2A",
    "IG_V_pseudogene" = "#808080",
    "IG_J_gene" = "#ADD8E6",
    "IG_C_gene" = "#90EE90",
    "sRNA" = "#00008B",
    "scRNA" = "#006400",
```

```
    "IG_C_pseudogene" = "#8B0000",
    "IG_D_gene" = "#DAA520",
    "IG_D_pseudogene" = "#2E8B57",
    "IG_pseudogene" = "#FF69B4",
    "Mt_tRNA" = "#D2B48C",
    "Mt_rRNA" = "#D3D3D3"
  )) +
  theme_minimal()
dev.off()
```

```
## pdf
##   2
```

```
# Calcular la media para cada grupo
mean_data2 <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_fraction = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Fraction[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data2 <- do.call(rbind, mean_data2)

# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data2, aes(x = Group, y = mean_fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Count de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "protein_coding" = "#0FBBF0",
    "TEC" = "#0000FF",  # Azul
    "snRNA" = "#008000",  # Verde
    "processed_pseudogene" = "#800080",  # Morado
    "lncRNA" = "#FFA500",  # Naranja
    "miRNA" = "#FFC0CB",  # Rosa
    "snoRNA" = "#A52A2A",  # Marrón
    "misc_RNA" = "#808080",  # Gris
    "transcribed_unprocessed_pseudogene" = "#ADD8E6",  # Azul claro
    "unprocessed_pseudogene" = "#90EE90",  # Verde claro
    "rRNA" = "#00008B",  # Azul oscuro
    "transcribed_processed_pseudogene" = "#006400",  # Verde oscuro
    "ribozyme" = "#8B0000",  # Rojo oscuro
    "unitary_pseudogene" = "#DAA520",  # Amarillo
    "scaRNA" = "#2E8B57",  # Verde marino
    "pseudogene" = "#FF69B4",  # Rosa claro
    "transcribed_unitary_pseudogene" = "#D2B48C",  # Marrón claro
    "translated_unprocessed_pseudogene" = "#D3D3D3",  # Gris claro
    "TR_V_gene" = "#0000FF",
    "TR_V_pseudogene" = "#008000",
    "TR_D_gene" = "#FF0000",
```

```
    "TR_J_gene" = "#800080",
    "TR_J_pseudogene" = "#FFA500",
    "IG_LV_gene" = "#FFC0CB",
    "IG_V_gene" = "#A52A2A",
    "IG_V_pseudogene" = "#808080",
    "IG_J_gene" = "#ADD8E6",
    "IG_C_gene" = "#90EE90",
    "sRNA" = "#00008B",
    "scRNA" = "#006400",
    "IG_C_pseudogene" = "#8B0000",
    "IG_D_gene" = "#DAA520",
    "IG_D_pseudogene" = "#2E8B57",
    "IG_pseudogene" = "#FF69B4",
    "Mt_tRNA" = "#D2B48C",
    "Mt_rRNA" = "#D3D3D3"
  ))  +
  theme_minimal()
```



Distribusció segons aligment

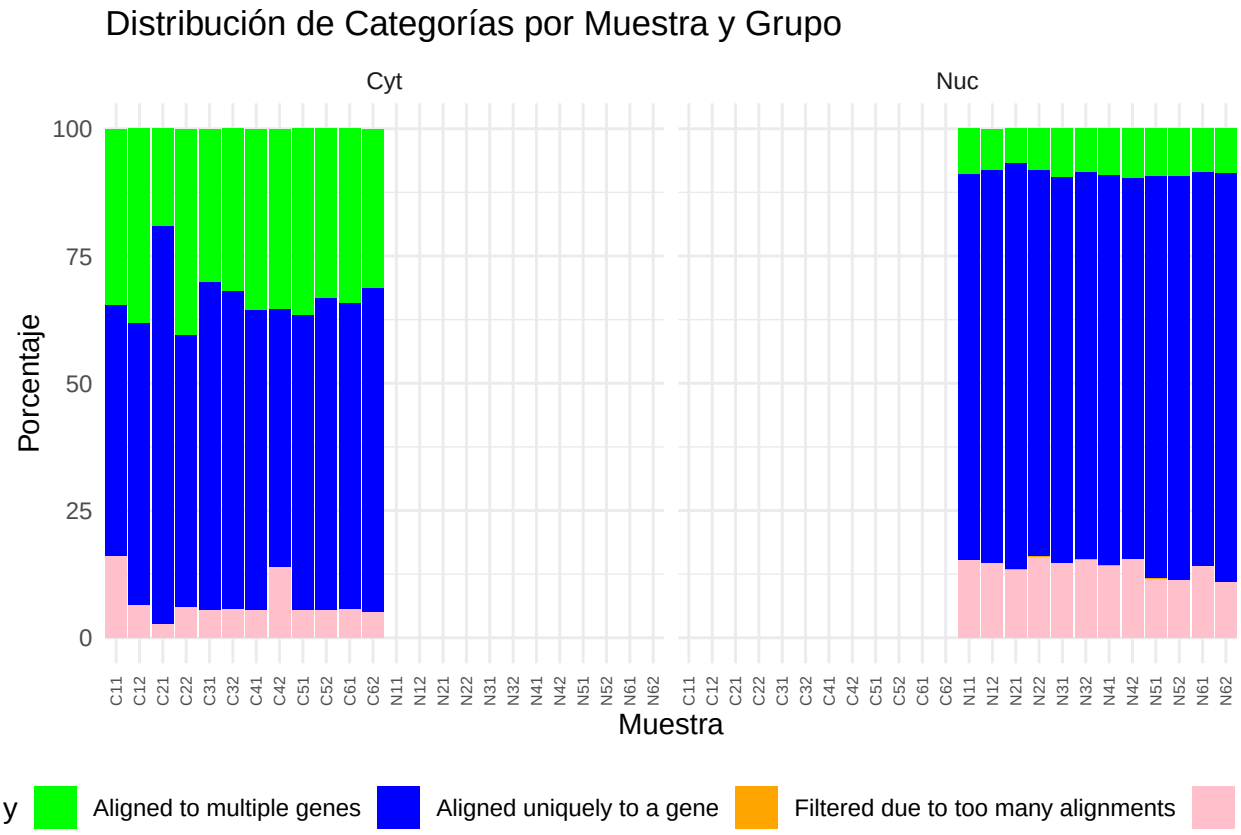
```
genomic_assigment<- read_delim("./mqc_rsem_assignment_plot_1.txt")
```

```
## Rows: 120 Columns: 5
```

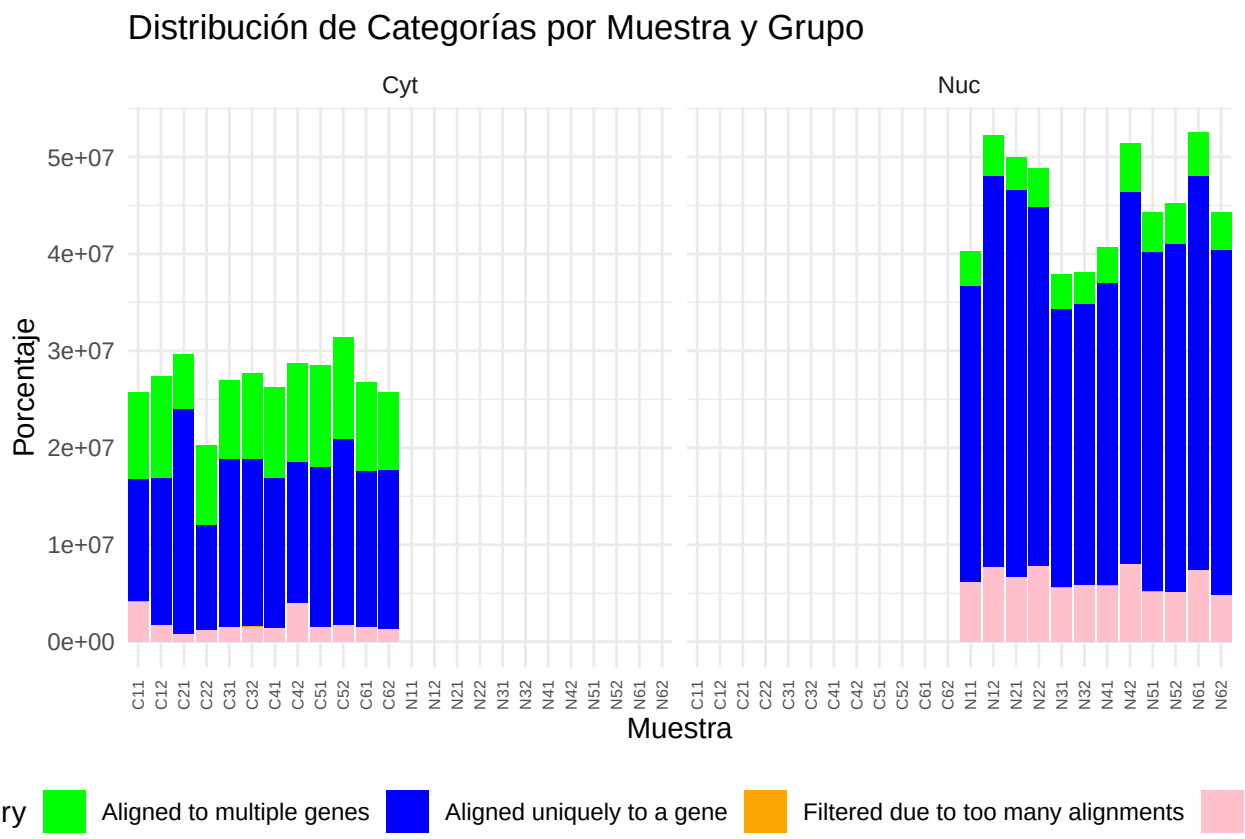
```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): Sample
## dbl (4): Aligned uniquely to a gene, Aligned to multiple genes, Filtered due...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
genomic_assigment<- as.data.frame(genomic_assigment)
genomic_assigment<-genomic_assigment[-c(1:36, 61:120),]

merge_assigment<-merge(x=info, y=genomic_assigment, by.x = "Run", by.y = "Sample")
merge_assigment<-merge_assigment[,-c(2:11)]
```

```
genomic_assigment_long  <- gather(merge_assigment, Category, Fraction, "Aligned uniquely to a gene", "Al

genomic_assigment_long <- genomic_assigment_long %>%
  group_by(Sample) %>%
  mutate(Percentage = (Fraction / sum(Fraction)) * 100)

# Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nucle
ggplot(genomic_assigment_long, aes(x = Sample, y = Percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Group, nrow = 1) +  # Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c("Aligned uniquely to a gene" = "blue", "Aligned to multiple genes" = "gr
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```




```
  # Crear un gráfico de barras apiladas con porcentajes y secciones separadas para "Cytoplasmic" y "Nuc
ggplot(genomic_assigment_long, aes(x = Sample, y = Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_grid(. ~ Group) +# Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c("Aligned uniquely to a gene" = "blue", "Aligned to multiple genes" = "gr
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```

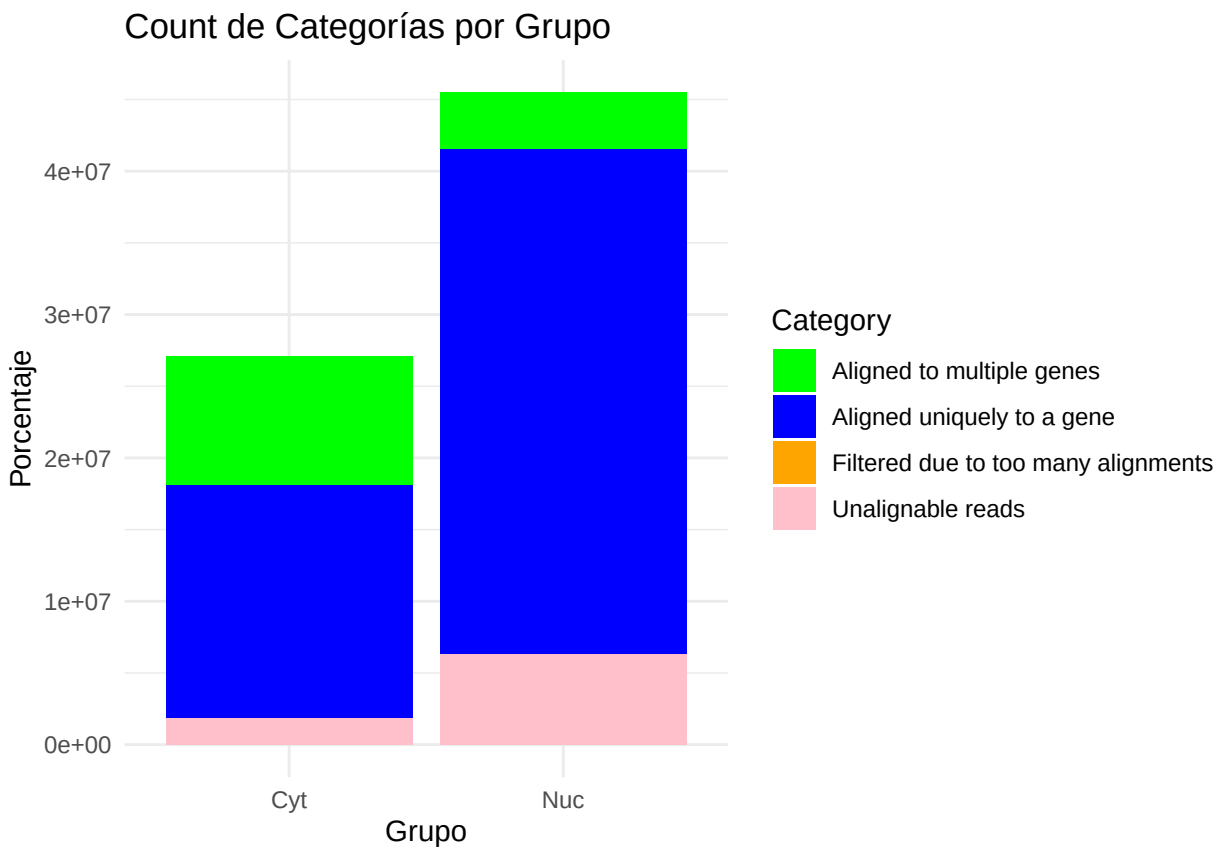



```r
groups <- split(genomic_assigment_long, genomic_assigment_long$Group)

# Calcular la media para cada grupo
mean_data <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_percentage = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Percentage[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data <- do.call(rbind, mean_data)

pdf(paste("Grafic4.pdf",sep=""))
# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data, aes(x = Group, y = mean_percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c("Aligned uniquely to a gene" = "blue", "Aligned to multiple genes" = "gr
  theme_minimal() +
  theme(
    legend.text = element_text(size = 18),      # Tamaño del texto de la leyenda
    legend.title = element_text(size = 20)      # Tamaño del título de la leyenda
  )
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
# Calcular la media para cada grupo
mean_data2 <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_fraction = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Fraction[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data2 <- do.call(rbind, mean_data2)

# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data2, aes(x = Group, y = mean_fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Count de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c("Aligned uniquely to a gene" = "blue", "Aligned to multiple genes" = "gr
  theme_minimal()
```



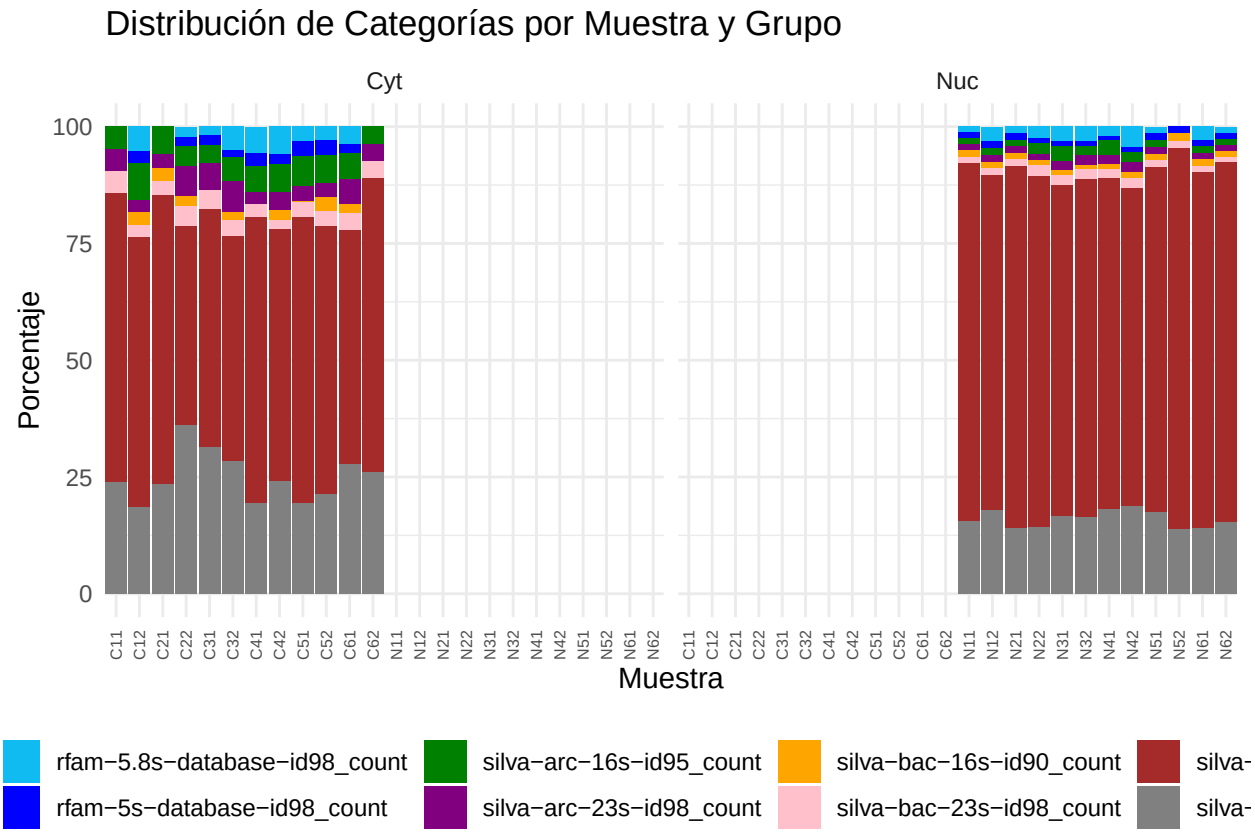
Distribució segons SortmeRNA

```
genomic_sortmerna<- read_delim("./mqc_sortmerna-detailed-plot_1.txt")
```

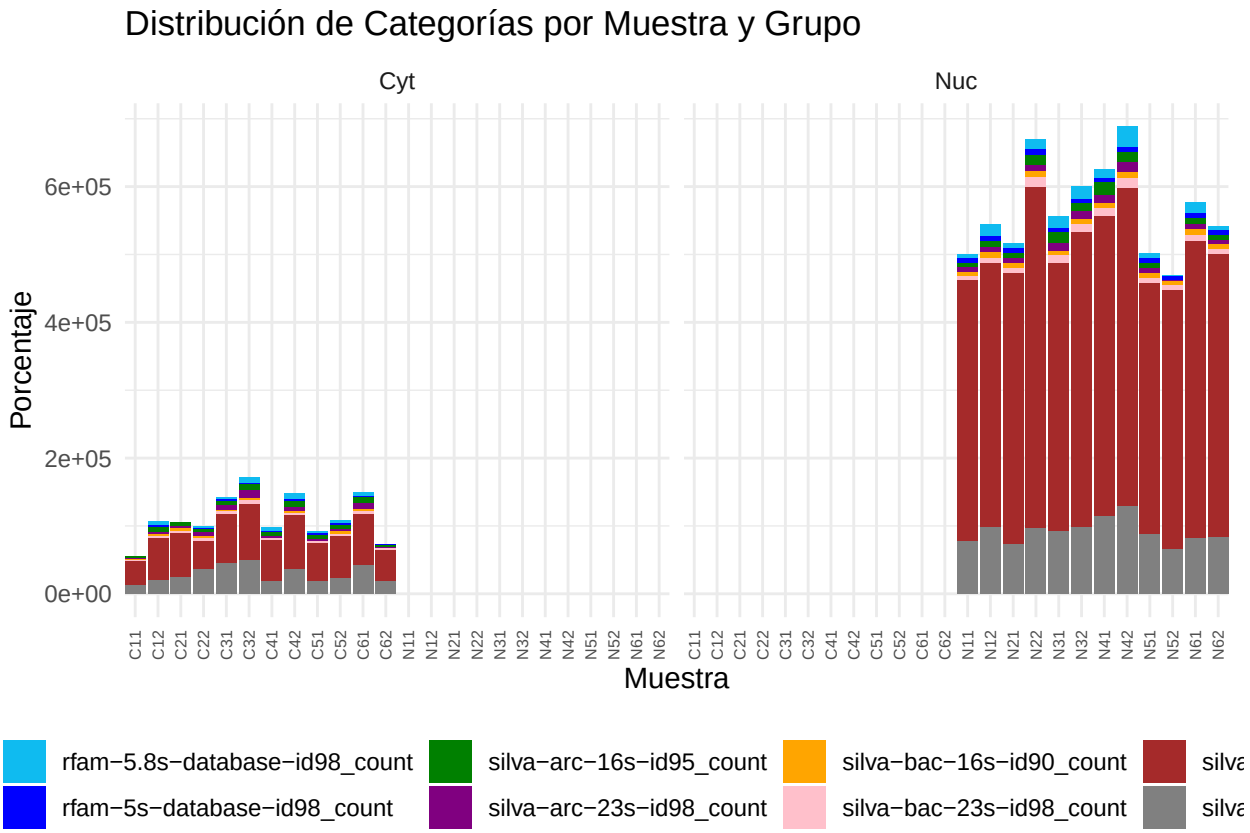
```
## Rows: 120 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): Sample
## dbl (8): rfam-5.8s-database-id98_count, rfam-5s-database-id98_count, silva-a...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
genomic_sortmerna<- as.data.frame(genomic_sortmerna)
genomic_sortmerna<-genomic_sortmerna[-c(1:36, 61:120),]

merge_sortmerna<-merge(x=info, y=genomic_sortmerna, by.x = "Run", by.y = "Sample")
merge_sortmerna<-merge_sortmerna[,-c(2:11)]
```

```
genomic_sortmerna_long  <- gather(merge_sortmerna, Category, Fraction, "rfam-5.8s-database-id98_count"

genomic_sortmerna_long <- genomic_sortmerna_long %>%
  group_by(Sample) %>%
  mutate(Percentage = (Fraction / sum(Fraction)) * 100)

ggplot(genomic_sortmerna_long, aes(x = Sample, y = Percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Group, nrow = 1) +
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "rfam-5.8s-database-id98_count" = "#0FBBF0",
    "rfam-5s-database-id98_count" = "#0000FF",
    "silva-arc-16s-id95_count" = "#008000",
    "silva-arc-23s-id98_count" = "#800080",
    "silva-bac-16s-id90_count" = "#FFA500",
    "silva-bac-23s-id98_count" = "#FFC0CB",
    "silva-euk-18s-id95_count" = "#A52A2A",
    "silva-euk-28s-id98_count" = "#808080"
  )) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```




```r
ggplot(genomic_sortmerna_long, aes(x = Sample, y = Fraction, fill = Category)) +
  geom_bar(stat = "identity") +
 facet_grid(. ~ Group) +# Usar facet_wrap para separar las secciones
  labs(title = "Distribución de Categorías por Muestra y Grupo", x = "Muestra", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "rfam-5.8s-database-id98_count" = "#0FBBF0",
    "rfam-5s-database-id98_count" = "#0000FF",
    "silva-arc-16s-id95_count" = "#008000",
    "silva-arc-23s-id98_count" = "#800080",
    "silva-bac-16s-id90_count" = "#FFA500",
    "silva-bac-23s-id98_count" = "#FFC0CB",
    "silva-euk-18s-id95_count" = "#A52A2A",
    "silva-euk-28s-id98_count" = "#808080"
  )) +
  theme_minimal() +
  scale_x_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1, size = 6)) +

  # Ajustar la posición de la leyenda
  theme(legend.position = "bottom",  # Colocar la leyenda debajo del gráfico
        legend.direction = "horizontal",  # Mostrar la leyenda en una fila
        legend.title = element_text(hjust = 0.5))  # Centrar el título de la leyenda
```

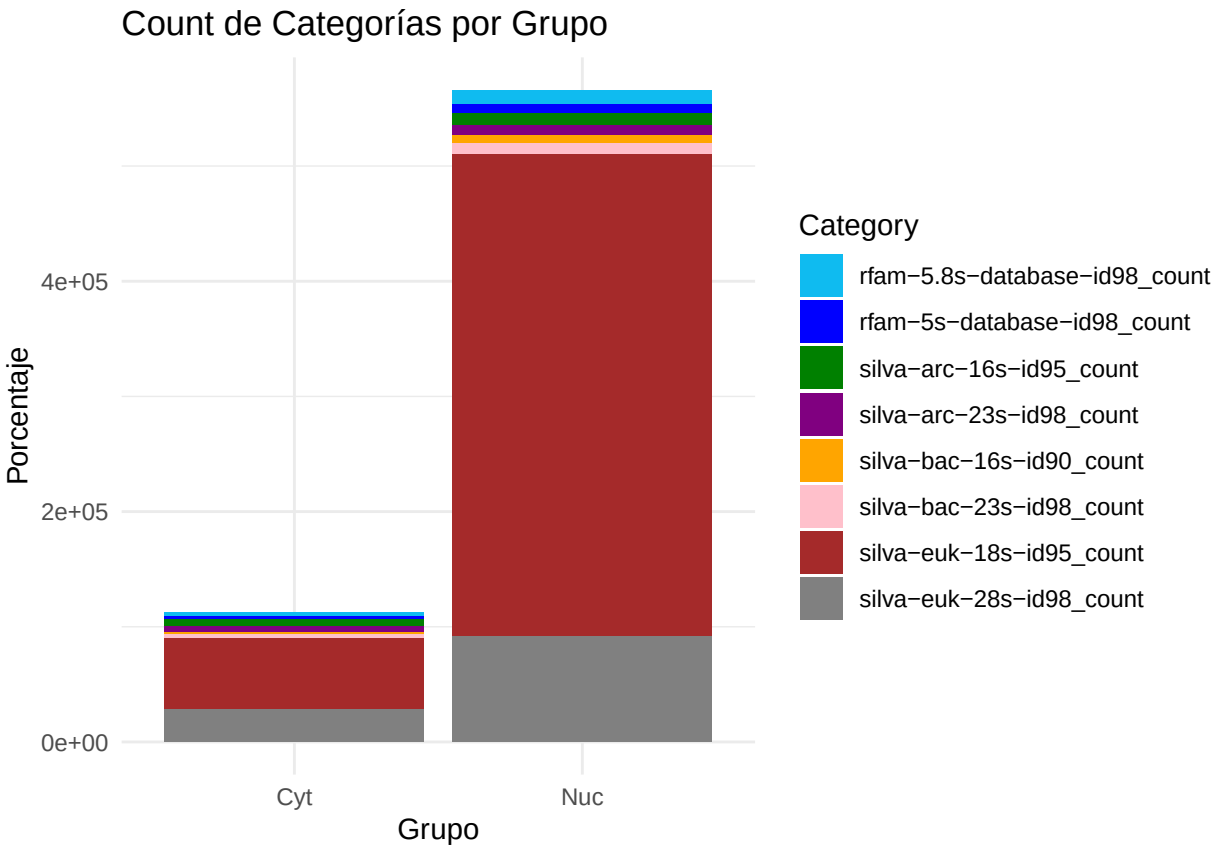



```r
groups <- split(genomic_sortmerna_long, genomic_sortmerna_long$Group)

# Calcular la media para cada grupo
mean_data <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_percentage = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Percentage[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data <- do.call(rbind, mean_data)
pdf(paste("Grafic5.pdf",sep=""))
# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data, aes(x = Group, y = mean_percentage, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Media de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "rfam-5.8s-database-id98_count" = "#0FBBF0",
    "rfam-5s-database-id98_count" = "#0000FF",
    "silva-arc-16s-id95_count" = "#008000",
    "silva-arc-23s-id98_count" = "#800080",
    "silva-bac-16s-id90_count" = "#FFA500",
    "silva-bac-23s-id98_count" = "#FFC0CB",
```

```
    "silva-euk-18s-id95_count" = "#A52A2A",
    "silva-euk-28s-id98_count" = "#808080"
  )) +
  theme_minimal()
dev.off()
```

```
## pdf
##   2
```

```
# Calcular la media para cada grupo
mean_data2 <- lapply(groups, function(group_data) {
  return(data.frame(
    Category = unique(group_data$Category),
    mean_fraction = sapply(unique(group_data$Category), function(cat) {
      mean(group_data$Fraction[group_data$Category == cat])
    }),
    Group = unique(group_data$Group)
  ))
})

# Convertir la lista de data frames en un único data frame
mean_data2 <- do.call(rbind, mean_data2)

# Crear un gráfico de barras apiladas en porcentaje con grupos en el eje X
ggplot(mean_data2, aes(x = Group, y = mean_fraction, fill = Category)) +
  geom_bar(stat = "identity") +
  labs(title = "Count de Categorías por Grupo", x = "Grupo", y = "Porcentaje") +
  scale_fill_manual(values = c(
    "rfam-5.8s-database-id98_count" = "#0FBBF0",
    "rfam-5s-database-id98_count" = "#0000FF",
    "silva-arc-16s-id95_count" = "#008000",
    "silva-arc-23s-id98_count" = "#800080",
    "silva-bac-16s-id90_count" = "#FFA500",
    "silva-bac-23s-id98_count" = "#FFC0CB",
    "silva-euk-18s-id95_count" = "#A52A2A",
    "silva-euk-28s-id98_count" = "#808080"
  )) +
  theme_minimal()
```


Count de Categorías por Grupo
Porcentaje
4e+05
2e+05
0e+00
Cyt
Nuc
Grupo
Category
rfam−5.8s−database−id98_count
rfam−5s−database−id98_count
silva−arc−16s−id95_count
silva−arc−23s−id98_count
silva−bac−16s−id90_count
silva−bac−23s−id98_count
silva−euk−18s−id95_count
silva−euk−28s−id98_count