Week2 GSE198256

Ana Cecilia González Álvarez

2024-January

Load Data

```
# Read data
urld <- "https://www.ncbi.nlm.nih.gov/geo/download/?format=file&type=rnaseq_counts"
path <- paste(urld, "acc=GSE198256", "file=GSE198256_raw_counts_GRCh38.p13_NCBI.tsv.gz", sep="&");
GSE198256_count <- as.matrix(data.table::fread(path, header=T, colClasses="integer"), rownames=1)
dim(GSE198256_count)
```

```
## [1] 39376    34
```

```
# Read Meta data
library(GEOquery)
gds <- getGEO("GSE198256")
Meta_GSE198256 <- pData(gds$GSE198256_series_matrix.txt.gz@phenoData)
Group <- Meta_GSE198256[,c("disease state:ch1")]
Group
```

```
##  [1] "Healthy"                  "Healthy"
##  [3] "Healthy"                  "Healthy"
##  [5] "Healthy"                  "Healthy"
##  [7] "Healthy"                  "Healthy"
##  [9] "Healthy"                  "Healthy"
## [11] "Healthy"                  "Covid19: Acute infection"
## [13] "Covid19: Acute infection" "Covid19: Acute infection"
## [15] "Covid19: Acute infection" "Covid19: Acute infection"
## [17] "Covid19: Acute infection" "Covid19: Acute infection"
## [19] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [21] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [23] "Covid19: Recovery 3Mo"    "Covid19: Recovery 3Mo"
## [25] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [27] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [29] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [31] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
## [33] "Covid19: Recovery 6Mo"    "Covid19: Recovery 6Mo"
```

Prepare Data

```r
# 1 prepare data ---------------------------------------------------------------------------
## Make sure row names in samples match column names in counts
### all columns in counts in varaibles rows?
all(colnames(GSE198256_count) %in% rownames(Meta_GSE198256)) ##T
```

```
## [1] TRUE
```

```r
### same order?
all(colnames(GSE198256_count) == rownames(Meta_GSE198256)) ##T
```

```
## [1] TRUE
```

```r
# Rename my conditions just so its easier to understand
Group[Group=="Healthy"] <- "Controls"
Group[Group=="Covid19: Acute infection"] <- "Acute"
Group[Group=="Covid19: Recovery 3Mo"] <- "EarlyRecovery"
Group[Group=="Covid19: Recovery 6Mo"] <- "LateRecovery"


### As dataframe for DESEQ2
Group_df <- data.frame(Group)
```

DESEQ2

Need: 1. Counts 2. Sample Info

```r
library(DESeq2)


# 2 Create DESeqDataSet object ------------------------------------------------------------
GSE198256_DESeq2 <- DESeqDataSetFromMatrix(countData = GSE198256_count,
                                           colData = Group_df,
                                           design = ~ Group) ## in colData what is the name of the col



# 3 QC and Filtering  ---------------------------------------------------------------------
# Keeping only rows with at least 10 reads
smallestGroupSize <- 6
keep <- rowSums(counts(GSE198256_DESeq2) >= 10) >= smallestGroupSize
GSE198256_DESeq2_F <- GSE198256_DESeq2[keep,]
nrow(GSE198256_DESeq2)##39376
```

```
## [1] 39376
```

```r
nrow(GSE198256_DESeq2_F) ###16737
```

```
## [1] 16737
```

```
# 4 Factors  ------------------------------------------------------------------------
GSE198256_DESeq2_F$Group ## Levels: Acute Controls EarlyRecovery LateRecovery
```

```
##  [1] Controls      Controls      Controls      Controls      Controls
##  [6] Controls      Controls      Controls      Controls      Controls
## [11] Controls      Acute         Acute         Acute         Acute
## [16] Acute         Acute         Acute         EarlyRecovery EarlyRecovery
## [21] EarlyRecovery EarlyRecovery EarlyRecovery EarlyRecovery LateRecovery
## [26] LateRecovery  LateRecovery  LateRecovery  LateRecovery  LateRecovery
## [31] LateRecovery  LateRecovery  LateRecovery  LateRecovery
## Levels: Acute Controls EarlyRecovery LateRecovery
```

```
GSE198256_DESeq2_F$Group <-  relevel(GSE198256_DESeq2_F$Group, ref = 'Controls') ## set healthy as ref

# 5 Run DESeq ------------------------------------------------------------------------
GSE198256_DESeq2_F<- DESeq(GSE198256_DESeq2_F)
resultsNames(GSE198256_DESeq2_F) ##[1] "Intercept" "Group_Acute_vs_Controls"   "Group_EarlyRecovery_vs_
```

```
## [1] "Intercept"                      "Group_Acute_vs_Controls"
## [3] "Group_EarlyRecovery_vs_Controls" "Group_LateRecovery_vs_Controls"
```

```
### 6.1 Contrasts ------------------------------------------------------------------------
res <- results(GSE198256_DESeq2_F, contrast=c('Group','Acute','Controls'))
filtered_res <- res[!is.na(res$log2FoldChange) &
                      !is.na(res$padj) &
                      (abs(res$log2FoldChange) > 1) &
                      res$padj < 0.05, ]
summary(filtered_res)
```

```
##
## out of 1273 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 418, 33%
## LFC < 0 (down)     : 855, 67%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# LFC > 0 (up)       : 418, 33%
# LFC < 0 (down)     : 855, 67%

res <- results(GSE198256_DESeq2_F, contrast=c('Group','EarlyRecovery','Controls'))
filtered_res <- res[!is.na(res$log2FoldChange) &
                      !is.na(res$padj) &
                      (abs(res$log2FoldChange) > 1) &
                      res$padj < 0.05, ]
summary(filtered_res)
```

```
##
```

```
## out of 965 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 506, 52%
## LFC < 0 (down)     : 459, 48%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# LFC > 0 (up)       : 506, 52%
# LFC < 0 (down)     : 459, 48%

res <- results(GSE198256_DESeq2_F, contrast=c('Group','LateRecovery','Controls'))
filtered_res <- res[!is.na(res$log2FoldChange) &
                      !is.na(res$padj)   &
                      (abs(res$log2FoldChange) > 1) &
                      res$padj < 0.05, ]
summary(filtered_res)
```

```
##
## out of 71 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 7, 9.9%
## LFC < 0 (down)     : 64, 90%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# LFC > 0 (up)       : 7, 9.9%
# LFC < 0 (down)     : 64, 90%
```

Limma: Normalize and set design

```
require(limma)
require(edgeR)

## Create DGEList
dge <- DGEList(counts=GSE198256_count)
nrow(dge)
```

```
## [1] 39376
```

```
# Design
design <- model.matrix(~ Group )

# Filter
keep <- filterByExpr(dge, design=design)
```

```
dge <- dge[keep,,keep.lib.sizes=FALSE]
nrow(dge)
```

```
## [1] 16409
```

```
dge$samples$norm.factors ### All 1
```

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

```
# Normalization
dge <- calcNormFactors(dge)
dge$samples$norm.factors ### norm.factors
```

```
##  [1] 0.9252685 0.8717521 0.8495083 1.1289978 0.9681553 1.2082121 1.0328488
##  [8] 1.0870069 1.0797373 0.7998169 1.0957984 1.1353119 1.1524320 1.0837039
## [15] 1.1263899 1.0172511 1.0527453 1.1563443 1.0714865 0.8093003 0.7114121
## [22] 0.7241662 0.8961464 1.0807998 1.0348874 0.9792803 1.1876698 0.8949173
## [29] 1.1641783 1.0762318 1.1731336 0.9427702 0.9089632 0.8958383
```

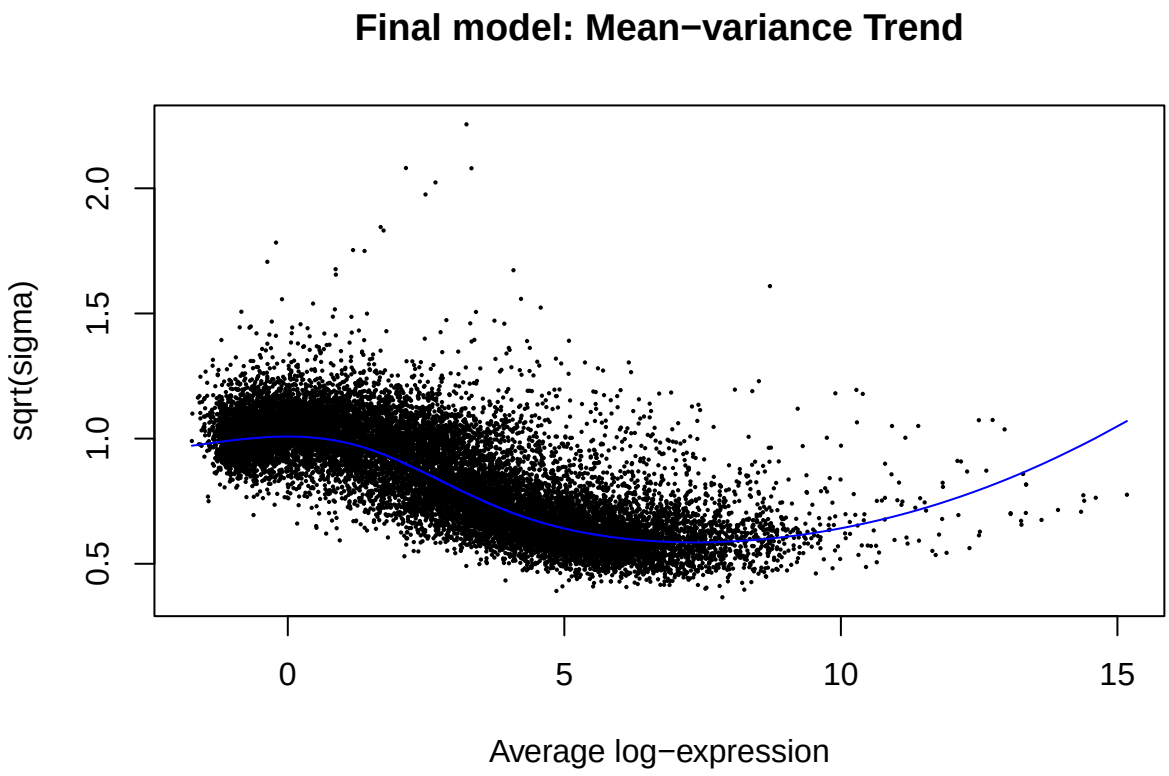
Limma: Voom vs Trend without contrasts

By default takes last one

```
## Trend

# If sequencing depth not too different between samples (less than 3 fold dif)
logCPM <- cpm(dge, log=TRUE, prior.count=3)
fit <- lmFit(logCPM, design)
fit <- eBayes(fit, trend=TRUE)
summary(decideTests(fit))
```

```
##        (Intercept) GroupControls GroupEarlyRecovery GroupLateRecovery
## Down           903           708                2852               656
## NotSig        3308         14993               11613             14998
## Up           12198           708                1944               755
```

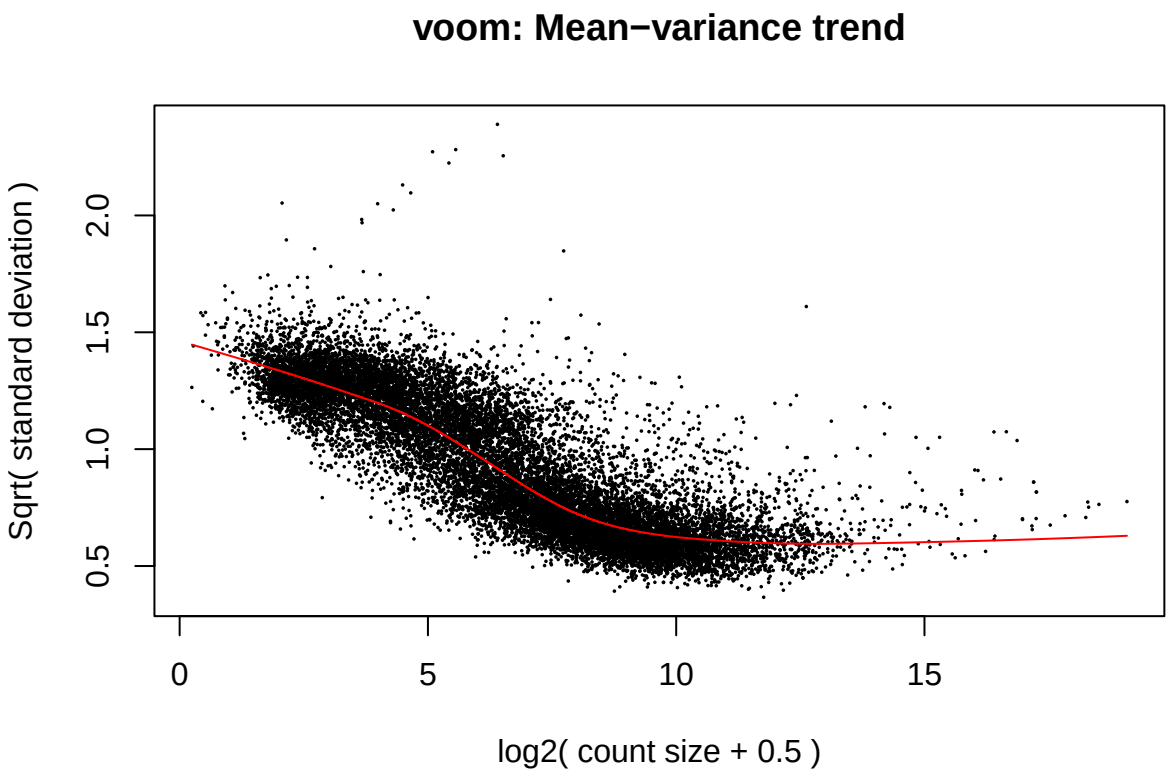
```
plotSA(fit, main = "Final model: Mean-variance Trend")
```




```
trend_topTable <- topTable(fit, coef=ncol(design))
trend_topTable
```

```
##           logFC    AveExpr         t      P.Value    adj.P.Val        B
## 9509  -8.351204  0.6515318 -9.937422 4.464258e-12 7.325401e-08 17.23697
## 51278  1.976317  5.3241986  8.978016 6.756457e-11 4.481291e-07 14.69031
## 57118  2.691818  4.8006409  8.911468 8.192988e-11 4.481291e-07 14.50878
## 8460  -4.981032  2.5000782 -8.631446 1.854889e-10 6.624582e-07 13.73824
## 7045   1.764092  8.2384423  8.602649 2.018582e-10 6.624582e-07 13.65840
## 11326 -2.875773  3.6231752 -8.450263 3.163114e-10 8.650589e-07 13.23403
## 22885 -4.183155 -0.7581040 -8.339297 4.394471e-10 1.030127e-06 12.92308
## 920    2.342202  7.2570299  8.258856 5.582168e-10 1.144972e-06 12.69668
## 55022  3.773515  6.5973933  8.174363 7.182497e-10 1.231474e-06 12.45799
## 7071   1.654727  8.1496958  8.130709 8.184120e-10 1.231474e-06 12.33431
```

```
trend_all <- topTable(fit, coef=ncol(design),nrow(GSE198256_count), sort.by = "none")


## Voom
# Library sizes are quite variable between samples
v <- voom(dge, design, plot=TRUE)
```




```
fit <- lmFit(v, design)
fit <- eBayes(fit)
summary(decideTests(fit))
```

```
##        (Intercept) GroupControls GroupEarlyRecovery GroupLateRecovery
## Down          1175          1011                2896               719
## NotSig        3384         14737               11585             14995
## Up           11850           661                1928               695
```

```
voom_topTable<-topTable(fit, coef=ncol(design))
voom_all<-topTable(fit, coef=ncol(design), nrow(GSE198256_count), sort.by = "none")
```

ACTIVITY 1:

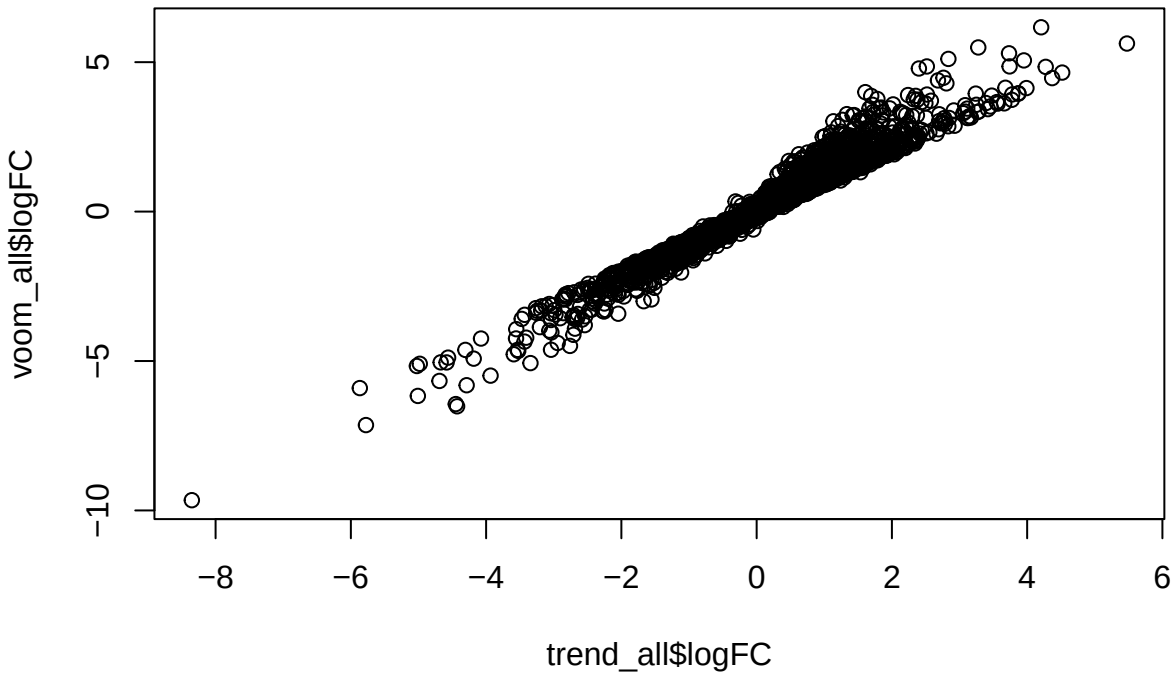
How would you compare the results between voom and trend?

For most of them the estimation i am getting is very similar between limma and voom Regarding the amount DEG we obtained a higher number for the downregulated using VOOM, while when using TREND we obtained a higher number for the upregulated

```
# How do we compare limma trend vs limma voom?
sum(rownames(trend_all)==rownames(voom_all)) ##16409
```

```
## [1] 16409
```

```
plot(trend_all$logFC,voom_all$logFC)
```



Is it required to run more than analysis? What exactly are we asking with this differential expression? Right now, because we have no contrast set. We are comparing them to the last Level alphabetically which is LateRecovery. And is not what the original paper was testing.

What we want to test is Acute,EarlyRecovery and LateRecovery vs Controls so we need to set contrasts

ACTIVITY 2:

Plan the next analysis:

We want to study each of our disease states Acute,EarlyRecovery and LateRecovery against our controls. So we need to: 1. Set design. Using control as the intercept `colnames(design) <- c("Intercept","Acute","EarlyRecovery","LateRecovery")` . 2. Set contrasts. Since the intercept is the control we are already testing each of the levels with the reference.makeContrasts(Acute, EarlyRecovery, LateRecovery,levels=design)‘

```
### LIMMA TREND  ADD  CONTRASTS ------------------------------------------------------------
###  If Contrasts need
###  1. colnames(design)
###  2. contrast.matrix
###  3. contrasts.fit

logCPM <- cpm(dge, log=TRUE, prior.count=3)
colnames(design) <- c("Intercept","Acute","EarlyRecovery","LateRecovery")
```

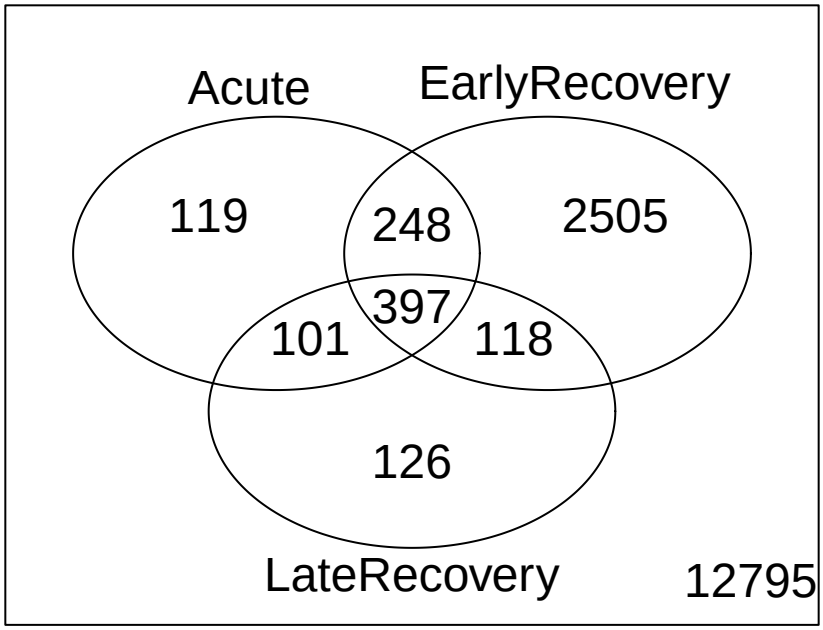
```
### intercept is Control is what is missing
### Controls Acute EarlyRecovery LateRecovery
fit <- lmFit(logCPM, design)
contrast.matrix <- makeContrasts(Acute, EarlyRecovery,
                                 LateRecovery,
                                 levels=design) ### Contrasts comparing everything to the intercept=Con
contrast.matrix
```

```
##                Contrasts
## Levels          Acute EarlyRecovery LateRecovery
##   Intercept         0             0            0
##   Acute             1             0            0
##   EarlyRecovery     0             1            0
##   LateRecovery      0             0            1
```

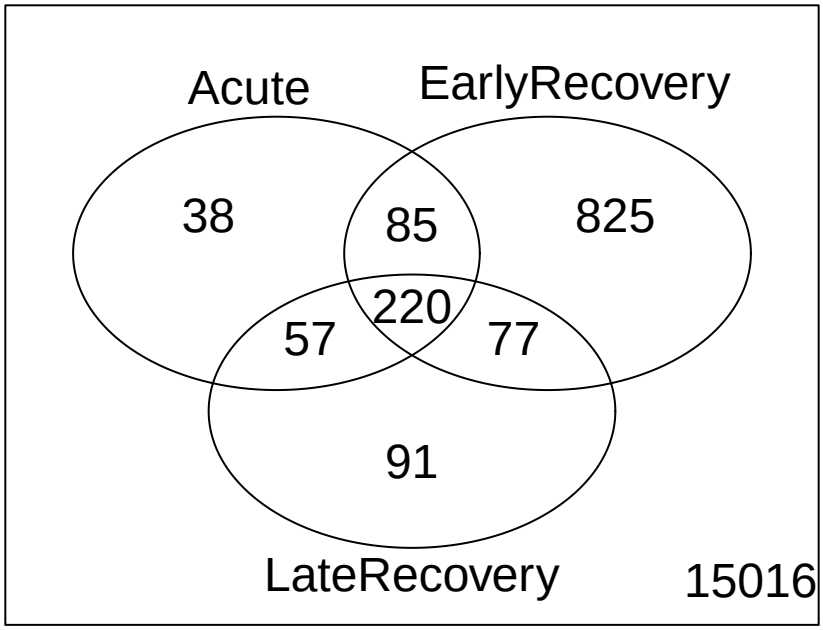
```
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
trend_topTable_CONTRASTS<-topTable(fit2)

results <- decideTests(fit2, adjust = "BH", p.value = 0.05, lfc = log2(2))
summary(decideTests(fit2))
```

```
##        Acute EarlyRecovery LateRecovery
## Down     669          2825          570
## NotSig 15089         11732        15165
## Up       651          1852          674
```

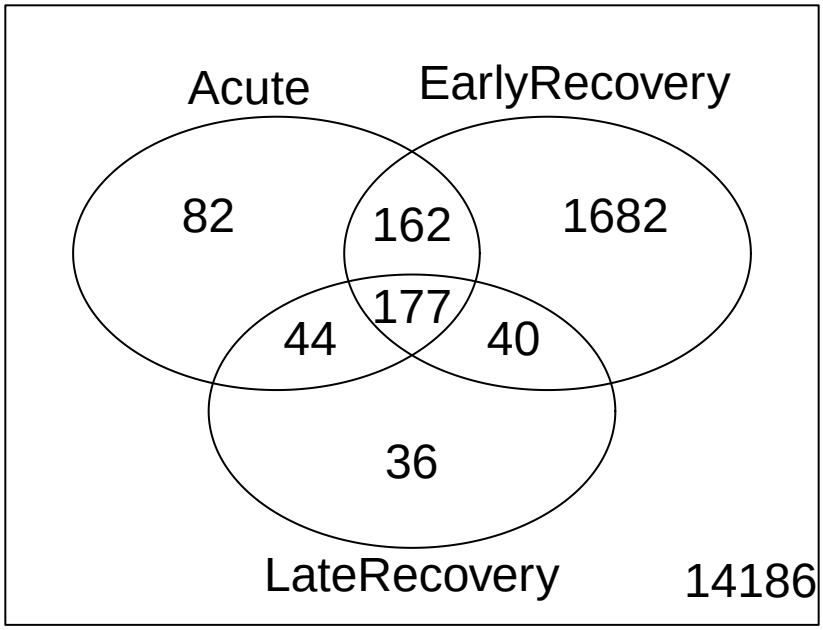
```
vennDiagram(results)
```




```
vennDiagram(results, include = "up")  # Only upregulated
```




```
vennDiagram(results, include = "down")  # Or downregulated
```




```
### https://mdozmorov.github.io/BIOS567/assets/presentation_diffexpression/DiffExpr_Limma.html

#write.table(topTable(fit2, coef = "B - L", number = 1000, adjust.method = "BH", p.value = 0.05, lfc =


topTable(fit2)
```

```
##               Acute EarlyRecovery LateRecovery    AveExpr        F      P.Value
## 22885     -4.808056     -4.465491    -4.183155 -0.7581040 38.95412 6.094036e-11
## 9509      -7.329539     -6.729428    -8.351204  0.6515318 35.23218 2.159979e-10
## 100534589 -2.750211     -3.052547    -2.468746 -1.1426183 34.22888 3.090911e-10
## 273       -4.806237     -5.106730    -4.306841  0.1405092 33.50655 4.020357e-10
## 2681       3.071050      2.574567     2.778411  2.8405740 32.59970 5.626413e-10
## 55022      4.018296      4.820864     3.773515  6.5973933 32.54590 5.740979e-10
## 7079      -3.636336     -3.262812    -3.344162 -1.5073937 31.59131 8.243420e-10
## 932       -3.013523     -3.332429    -2.581037 -1.3220467 31.00953 1.031763e-09
## 79071     -3.277514     -3.971939    -3.204220 -0.8003025 29.71975 1.715974e-09
## 3205      -2.862676     -3.105113    -2.423542 -0.8857215 29.63060 1.778430e-09
##               adj.P.Val
## 22885     9.999703e-07
## 9509      1.570062e-06
## 100534589 1.570062e-06
## 273       1.570062e-06
## 2681      1.570062e-06
## 55022     1.570062e-06
```

```
## 7079      1.932376e-06
## 932       2.116275e-06
## 79071     2.918225e-06
## 3205      2.918225e-06
```

```
topTable(fit2,coef=1)
```

```
##               logFC    AveExpr          t      P.Value    adj.P.Val        B
## 22885     -4.808056 -0.7581040 -10.040090 1.490162e-11 2.445207e-07 16.05362
## 2681       3.071050  2.8405740   9.234571 1.161084e-10 9.526111e-07 14.15501
## 7079      -3.636336 -1.5073937  -8.966412 2.343503e-10 9.904763e-07 13.50142
## 273       -4.806237  0.1405092  -8.920336 2.646552e-10 9.904763e-07 13.38804
## 100534589 -2.750211 -1.1426183  -8.870681 3.018088e-10 9.904763e-07 13.26551
## 9509      -7.329539  0.6515318  -8.580607 6.542818e-10 1.789352e-06 12.54249
## 932       -3.013523 -1.3220467  -8.518875 7.724655e-10 1.810770e-06 12.38704
## 3205      -2.862676 -0.8857215  -8.404179 1.052980e-09 1.928076e-06 12.09678
## 55076     -4.612932 -0.8918986  -8.377009 1.133436e-09 1.928076e-06 12.02775
## 1718      -5.048316  1.4274992  -8.363728 1.175011e-09 1.928076e-06 11.99396
```

```
topTable(fit2,coef=2)
```

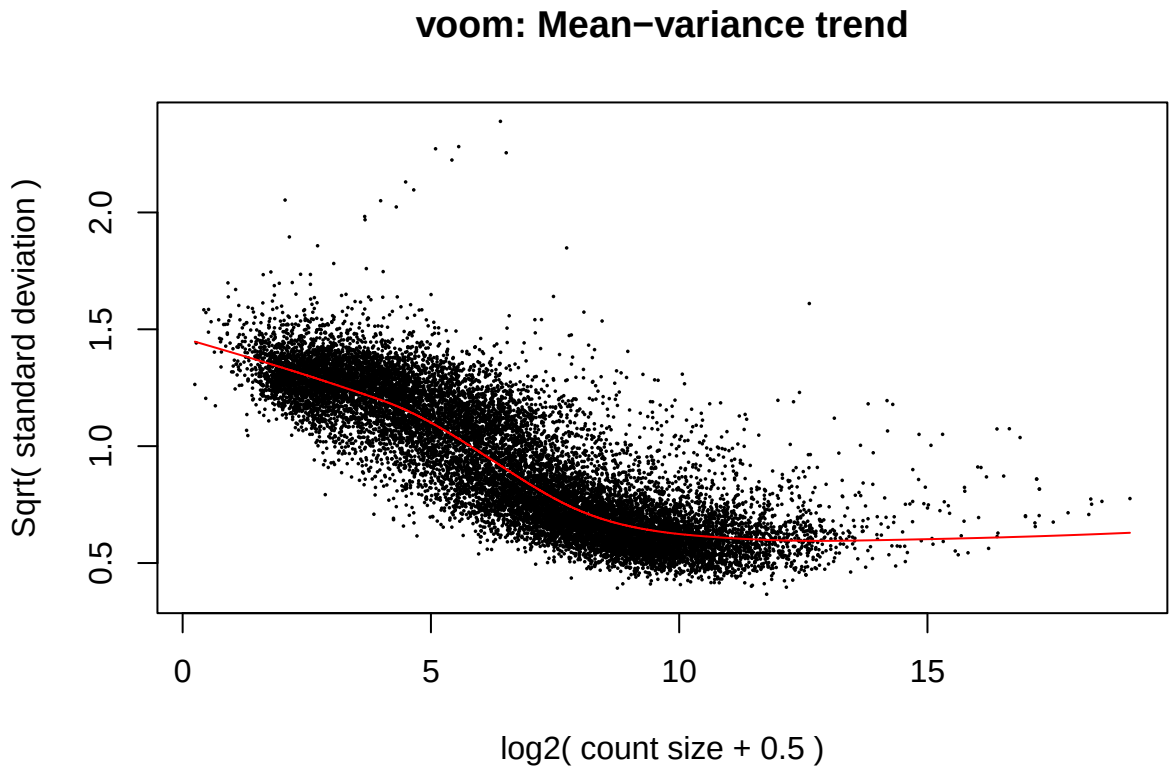
```
##               logFC    AveExpr          t      P.Value    adj.P.Val        B
## 55022      4.820864  6.5973933  8.650340 5.426950e-10 4.529639e-06 12.75729
## 100534589 -3.052547 -1.1426183 -8.556520 6.980357e-10 4.529639e-06 12.51998
## 9308       4.885358  6.0040253  8.450975 9.277710e-10 4.529639e-06 12.25151
## 79071     -3.971939 -0.8003025 -8.238937 1.650103e-09 4.529639e-06 11.70742
## 273       -5.106730  0.1405092 -8.236880 1.659391e-09 4.529639e-06 11.70212
## 932       -3.332429 -1.3220467 -8.186769 1.902865e-09 4.529639e-06 11.57261
## 22885     -4.465491 -0.7581040 -8.103660 2.389544e-09 4.529639e-06 11.35706
## 151963     4.438264  1.8103093  8.092085 2.466711e-09 4.529639e-06 11.32696
## 481        3.720613  1.9467443  8.089482 2.484414e-09 4.529639e-06 11.32019
## 23308      4.533395  2.3947701  8.031701 2.912368e-09 4.778904e-06 11.16967
```

```
topTable(fit2,coef=3)
```

```
##               logFC    AveExpr          t      P.Value    adj.P.Val        B
## 9509      -8.351204  0.6515318 -9.591928 4.621137e-11 7.582823e-07 15.02372
## 22885     -4.183155 -0.7581040 -8.570133 6.729534e-10 3.623140e-06 12.52569
## 51278      1.976317  5.3241986  8.533558 7.425183e-10 3.623140e-06 12.43344
## 57118      2.691818  4.8006409  8.469198 8.832081e-10 3.623140e-06 12.27063
## 8460      -4.981032  2.5000782 -8.339996 1.253205e-09 4.112769e-06 11.94203
## 11326     -2.875773  3.6231752 -8.229210 1.694498e-09 4.340466e-06 11.65840
## 2681       2.778411  2.8405740  8.196752 1.851622e-09 4.340466e-06 11.57497
## 102724008 -4.584507  0.3407294 -8.132166 2.209781e-09 4.521120e-06 11.40854
## 7079      -3.344162 -1.5073937 -8.090167 2.479741e-09 4.521120e-06 11.30000
## 7045       1.764092  8.2384423  8.048791 2.778521e-09 4.559274e-06 11.19284
```

```
### LIMMA VOOM  ADD  CONTRASTS ------------------------------------------------------
###  If Contrasts need
###  1. colnames(design)
###  2. contrast.matrix
```

```
###  3. contrasts.fit

v <- voom(dge, design, plot=TRUE)
```




```
colnames(design) <- c("Intercept","Acute","EarlyRecovery","LateRecovery")


### intercept is Control is what is missing
### Controls Acute EarlyRecovery LateRecovery
fit <- lmFit(v, design)
contrast.matrix <- makeContrasts(Acute, EarlyRecovery,
                                 LateRecovery,
                                 levels=design) ### Contrasts comparing everything to the intercept=Con
contrast.matrix
```

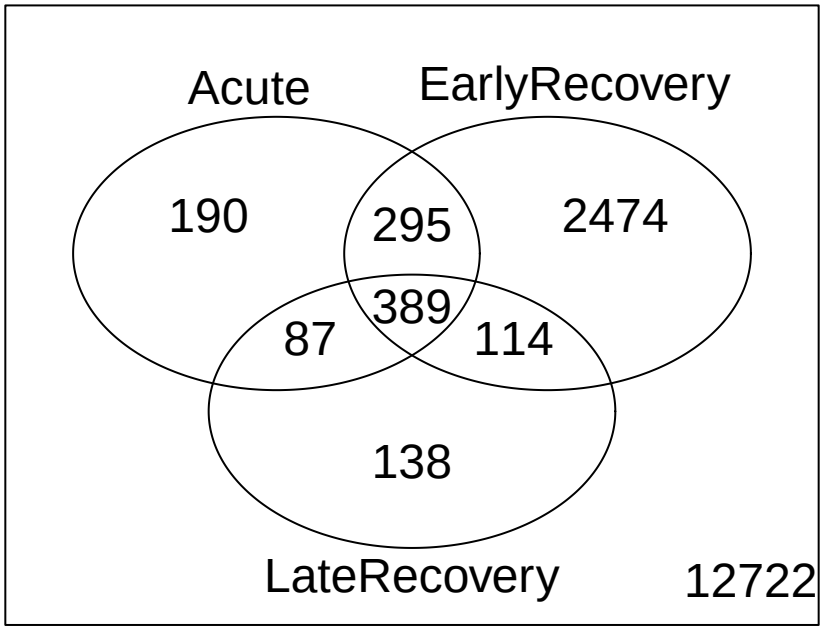
```
##                Contrasts
## Levels          Acute EarlyRecovery LateRecovery
##   Intercept         0             0            0
##   Acute             1             0            0
##   EarlyRecovery     0             1            0
##   LateRecovery      0             0            1
```

```
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
```

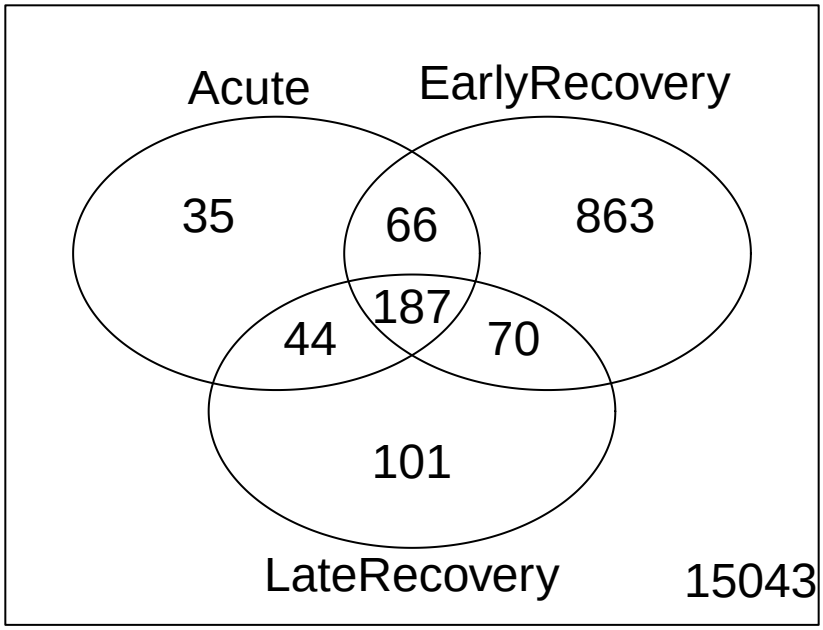
```r
voom_topTable_CONTRASTS<-topTable(fit2)

results <- decideTests(fit2, adjust = "BH", p.value = 0.05, lfc = log2(2))
summary(decideTests(fit2, adjust = "BH", p.value = 0.05, lfc = log2(2)))
```

```
##        Acute EarlyRecovery LateRecovery
## Down     629          2086          326
## NotSig 15448         13137        15681
## Up       332          1186          402
```

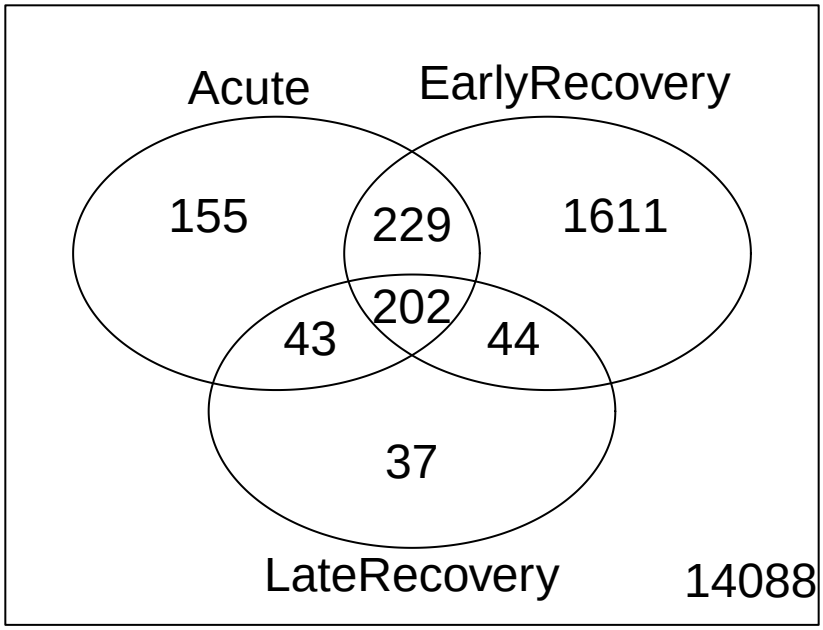
```r
vennDiagram(results)
```




```r
vennDiagram(results, include = "up")  # Only upregulated
```




```
vennDiagram(results, include = "down")  # Or downregulated
```




```
### https://mdozmorov.github.io/BIOS567/assets/presentation_diffexpression/DiffExpr_Limma.html

#write.table(topTable(fit2, coef = "B - L", number = 1000, adjust.method = "BH", p.value = 0.05, lfc =


topTable(fit2)
```

```
##           Acute EarlyRecovery LateRecovery    AveExpr        F      P.Value
## 9509  -8.105772     -7.668233    -9.656670 -0.2386102 35.61236 5.037560e-11
## 7045   1.612767      1.694176     1.766524  8.2375143 31.33206 2.812885e-10
## 920    2.126645      2.447447     2.351524  7.2546704 29.43713 6.355782e-10
## 55022  3.974775      4.776597     3.740634  6.5885428 27.98600 1.216535e-09
## 2322  -3.128272     -3.067106    -3.453648  4.3439994 27.49685 1.522023e-09
## 57118  2.453259      2.660830     2.754724  4.7869364 27.29268 1.672549e-09
## 11326 -2.611574     -2.922505    -2.942943  3.5989265 27.12694 1.806266e-09
## 7079  -5.852175     -4.856194    -5.069654 -3.2470195 26.77400 2.129954e-09
## 4082   2.051550      2.500927     2.032964  6.9588584 26.65350 2.254018e-09
## 55076 -6.573700     -5.411600    -5.810065 -2.3285550 26.59844 2.313224e-09
##          adj.P.Val
## 9509  8.266132e-07
## 7045  2.307831e-06
## 920   3.476401e-06
## 55022 3.795769e-06
## 2322  3.795769e-06
## 57118 3.795769e-06
```

```
## 11326 3.795769e-06
## 7079  3.795769e-06
## 4082  3.795769e-06
## 55076 3.795769e-06
```

```
topTable(fit2,coef=1)
```

```
##               logFC    AveExpr         t      P.Value    adj.P.Val        B
## 9509      -8.105772 -0.2386102 -9.151225 4.648245e-11 7.627305e-07 14.43235
## 55076     -6.573700 -2.3285550 -8.482649 3.201407e-10 1.751063e-06 12.11434
## 7045       1.612767  8.2375143  8.083755 1.038505e-09 3.716100e-06 11.96217
## 7079      -5.852175 -3.2470195 -8.568670 2.489933e-10 1.751063e-06 11.92837
## 11326     -2.611574  3.5989265 -7.982045 1.405949e-09 3.716100e-06 11.84199
## 2322      -3.128272  4.3439994 -7.951867 1.538509e-09 3.716100e-06 11.71199
## 102724008 -5.482236 -0.1681813 -7.854528 2.058741e-09 4.222736e-06 11.07869
## 4128      -5.933255 -1.2630316 -7.941846 1.585270e-09 3.716100e-06 11.05461
## 241       -2.033411  5.2271701 -7.567324 4.890415e-09 8.024682e-06 10.50831
## 920        2.126645  7.2546704  7.573392 4.801412e-09 8.024682e-06 10.47407
```

```
topTable(fit2,coef=2)
```

```
##            logFC   AveExpr         t      P.Value    adj.P.Val        B
## 55022   4.776597  6.588543  8.286939 5.689743e-10 6.713918e-06 12.75393
## 7439    2.060775 10.082736  8.076834 1.060096e-09 6.713918e-06 12.10370
## 64651   3.529106  6.521076  7.991889 1.365259e-09 6.713918e-06 11.88221
## 4082    2.500927  6.958858  7.875720 1.932074e-09 6.713918e-06 11.53467
## 2495    1.976997 10.260166  7.805896 2.382148e-09 6.713918e-06 11.30564
## 490     4.068117  8.330494  7.795870 2.454964e-09 6.713918e-06 11.28253
## 3727    1.612274  7.317048  7.727227 3.017845e-09 7.012510e-06 11.08185
## 920     2.447447  7.254670  7.685818 3.418860e-09 7.012510e-06 10.96994
## 124599  2.284304  5.724244  7.600989 4.416885e-09 7.362972e-06 10.74424
## 3148   -2.469250  6.192001 -7.595769 4.487155e-09 7.362972e-06 10.72666
```

```
topTable(fit2,coef=3)
```

```
##            logFC    AveExpr         t      P.Value    adj.P.Val        B
## 7045    1.766524  8.2375143  8.698624 1.706155e-10 1.399815e-06 13.89220
## 51278   1.967276  5.3175377  8.503181 3.014760e-10 1.648973e-06 13.37317
## 9509   -9.656670 -0.2386102 -9.272701 3.293194e-11 5.403802e-07 13.04765
## 920     2.351524  7.2546704  8.248931 6.365313e-10 2.313710e-06 12.61137
## 2322   -3.453648  4.3439994 -8.150728 8.512373e-10 2.313710e-06 12.36401
## 7071    1.656693  8.1487916  8.147832 8.585780e-10 2.313710e-06 12.29394
## 57118   2.754724  4.7869364  8.100865 9.870176e-10 2.313710e-06 12.18114
## 11326  -2.942943  3.5989265 -8.023540 1.242330e-09 2.548174e-06 11.97123
## 81545  -1.415826  5.2946801 -7.863748 2.002629e-09 3.651238e-06 11.50869
## 8682    1.797785  5.6954372  7.648659 3.824445e-09 6.275532e-06 10.87859
```

Biological Interpretation

Right now we have DE genes with ENTREZ ids. We will keep the results from the LIMMA voom analysis `fit2`

1. Retrieve ENSEMBL ids of the ENTREZ ids i have
2. Create DF with the ENTREZIDS genes from GSE198256_count and the retrieved ENSEMBL ids

```
library(clusterProfiler)
library(msigdbr)
library(org.Hs.eg.db)
library(magrittr)

keytypes(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GENETYPE"     "GO"           "GOALL"        "IPI"          "MAP"
## [16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
## [21] "PMID"         "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [26] "UNIPROT"
```

```
### We want to retrieve ENSEMBL id
ENSEMBL_vector <- mapIds(
  org.Hs.eg.db,
  keys = rownames(GSE198256_count),
  keytype = "ENTREZID",
  column = "ENSEMBL",
  multiVals = "first"
)

# We would like a data frame we can join to the differential expression stats
gene_key_df <- data.frame(
  ensembl_id = ENSEMBL_vector,
  entrez_id = names(ENSEMBL_vector),
  stringsAsFactors = FALSE
) %>%
  # If an Ensembl gene identifier doesn't map to a gene symbol, drop that
  # from the data frame
  dplyr::filter(!is.na(ensembl_id))
head(gene_key_df)
```

```
##                ensembl_id entrez_id
## 100287102 ENSG00000290825 100287102
## 102466751 ENSG00000278267 102466751
## 100302278 ENSG00000284332 100302278
## 645520    ENSG00000237613    645520
## 79501     ENSG00000186092     79501
## 102725121 ENSG00000290825 102725121
```

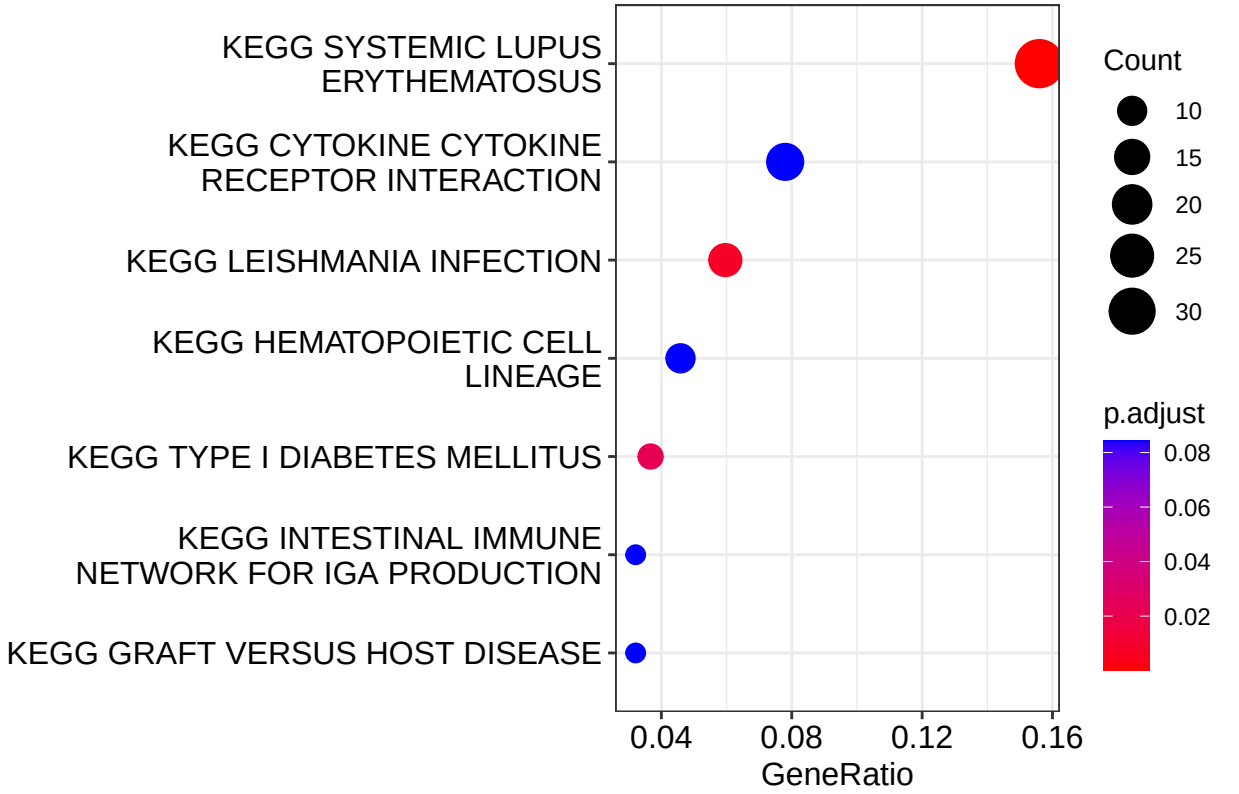
ORA

We need to determine a treshold for the genes we want to keep

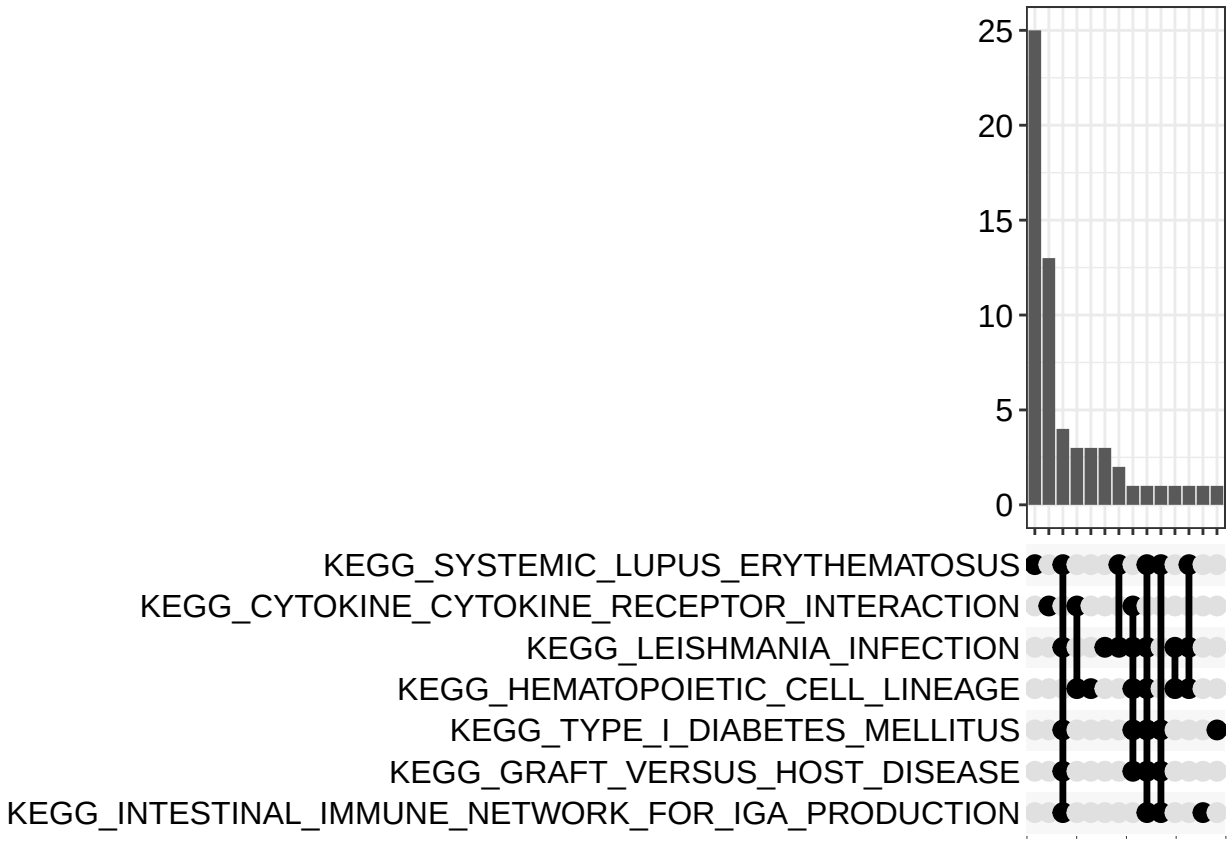
- What do we need to do the analysis?
- What are the tools required?

ORA CONTRAST1 ACUTE-HEALTHY

```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=1,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))

# Step 3: Determine gene sets.
#msigdbr_species()
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
#head(hs_msigdb_df)

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```



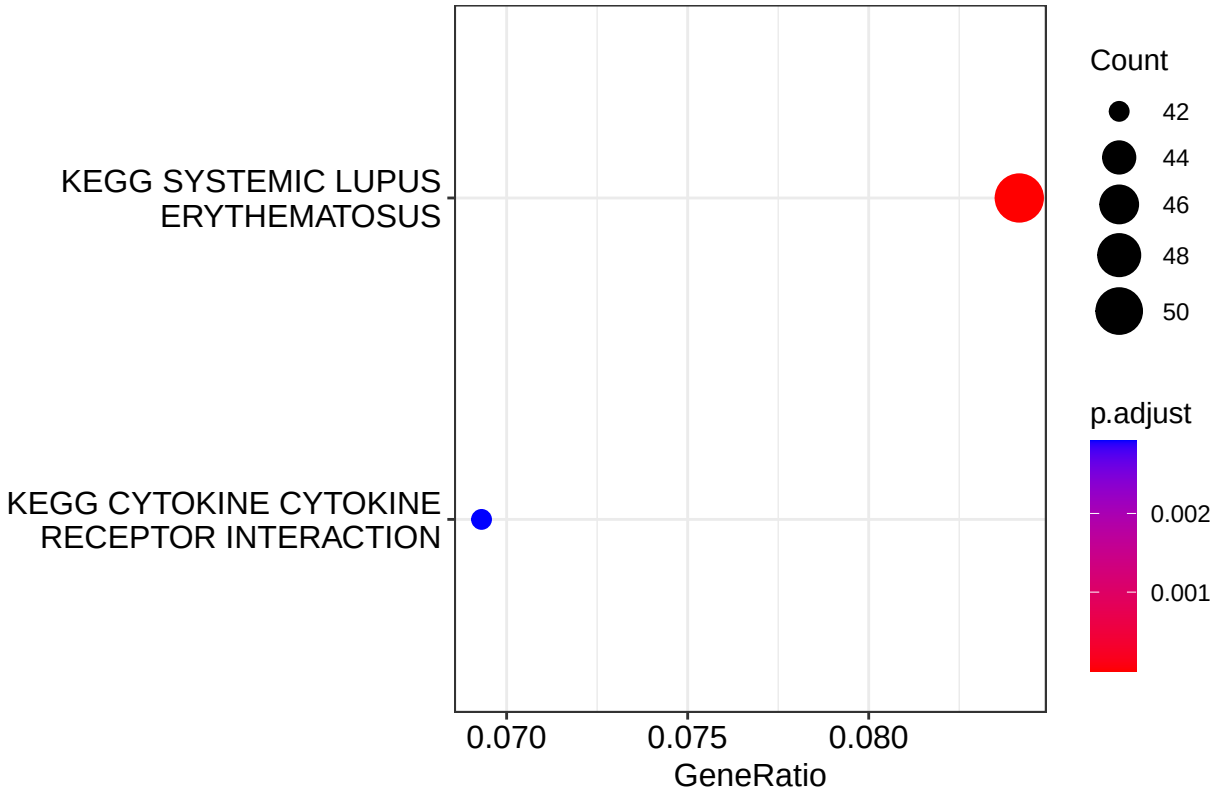

```
upset_plot <- enrichplot::upsetplot(kegg_ora_results)
upset_plot
```



ORA CONTRAST2 EARLY-HEALTHY

```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=2,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))

# Step 3: Determine gene sets.
#msigdbr_species()
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
#head(hs_msigdb_df)

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
```

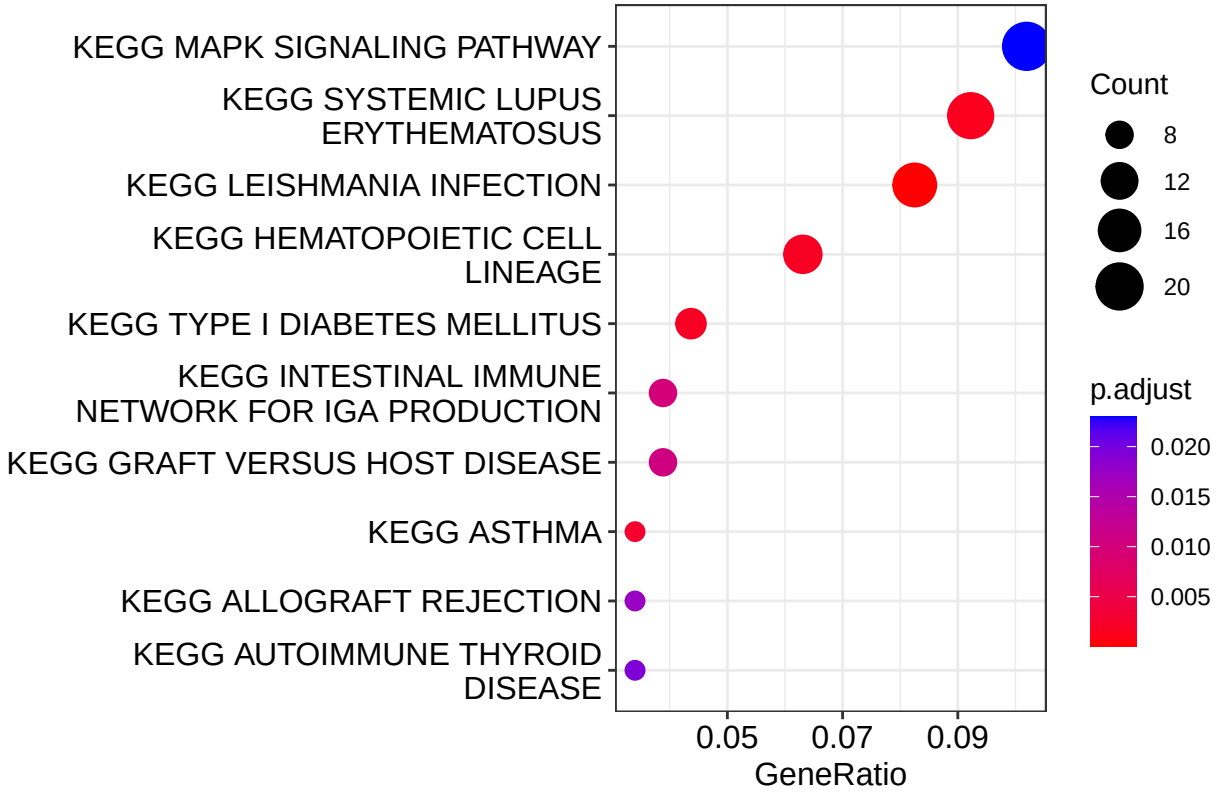
```
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```



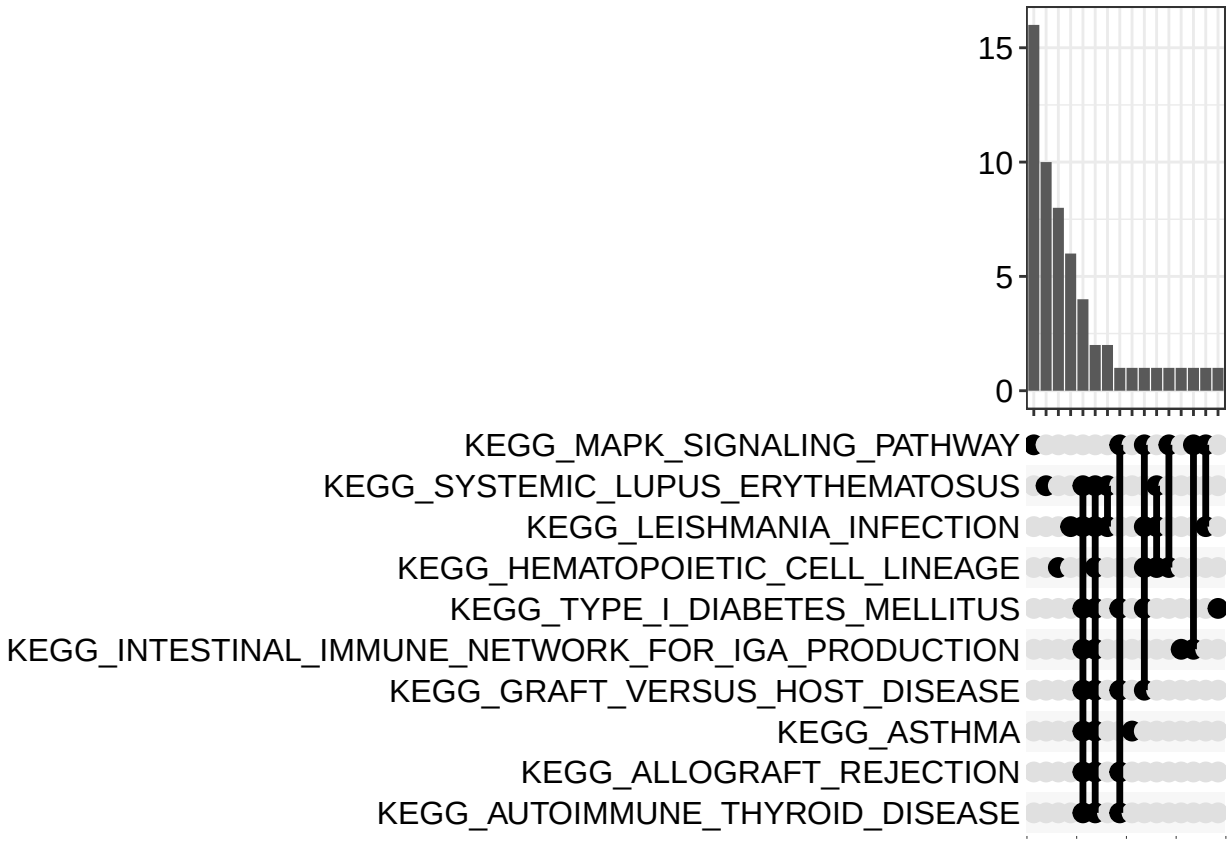
ORA CONTRAST3 LATE-HEALTHY

```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=3,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))

# Step 3: Determine gene sets.
#msigdbr_species()
```

```r
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
#head(hs_msigdb_df)

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```




```
upset_plot <- enrichplot::upsetplot(kegg_ora_results)
upset_plot
```

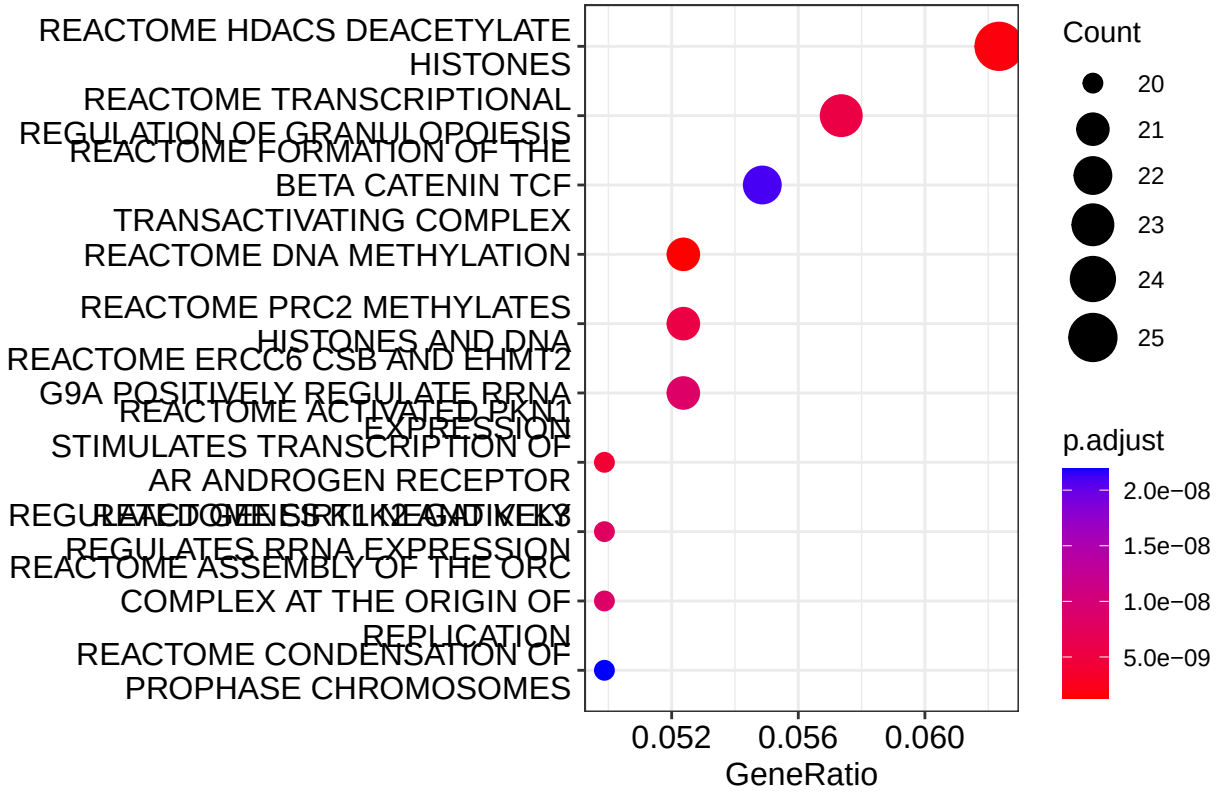



```
# Step 6: EXERCISE: alternatives to KEGG?
msigdbr_collections()
```

```
## # A tibble: 23 x 3
##    gs_cat gs_subcat         num_genesets
##    <chr>  <chr>                    <int>
##  1 C1     ""                         299
##  2 C2     "CGP"                     3384
##  3 C2     "CP"                        29
##  4 C2     "CP:BIOCARTA"              292
##  5 C2     "CP:KEGG"                  186
##  6 C2     "CP:PID"                   196
##  7 C2     "CP:REACTOME"             1615
##  8 C2     "CP:WIKIPATHWAYS"          664
##  9 C3     "MIR:MIRDB"               2377
## 10 C3     "MIR:MIR_Legacy"           221
## # i 13 more rows
```

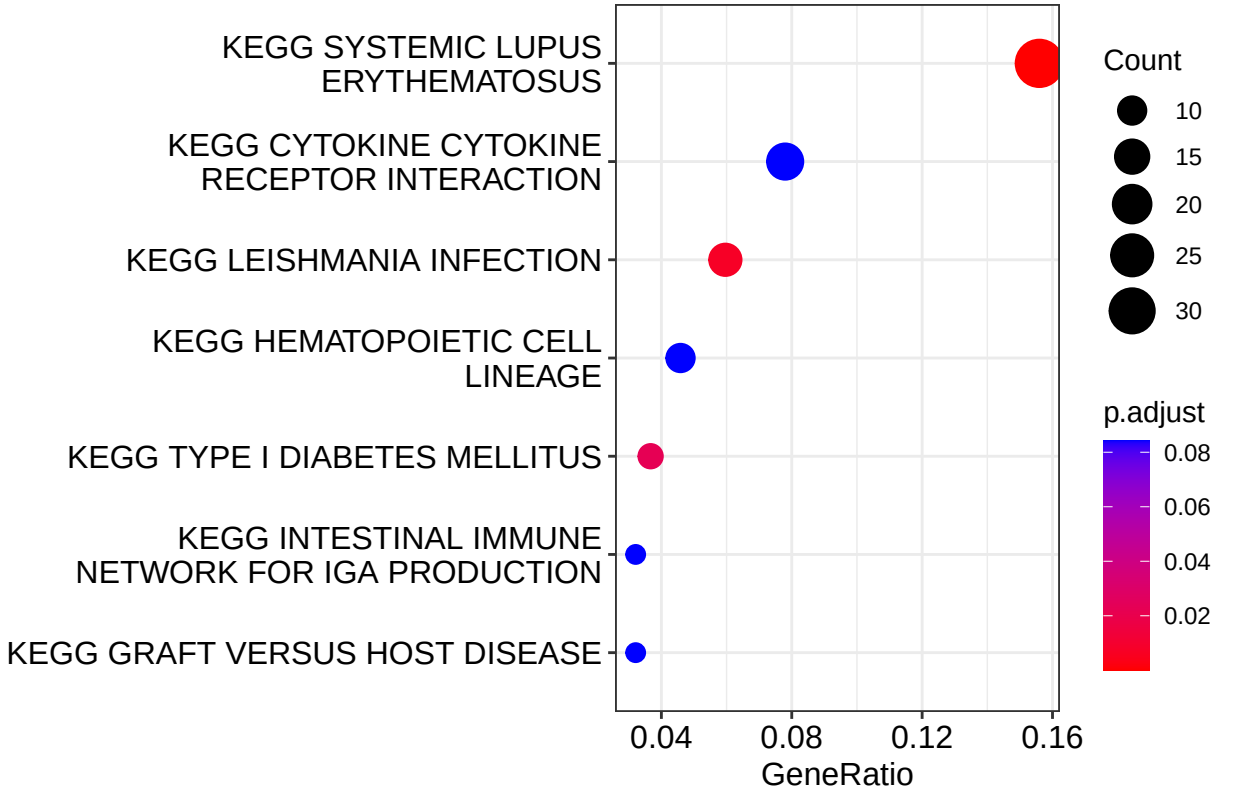
```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=1,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))
```

```r
# Step 3: Determine gene sets.
hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:REACTOME" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```

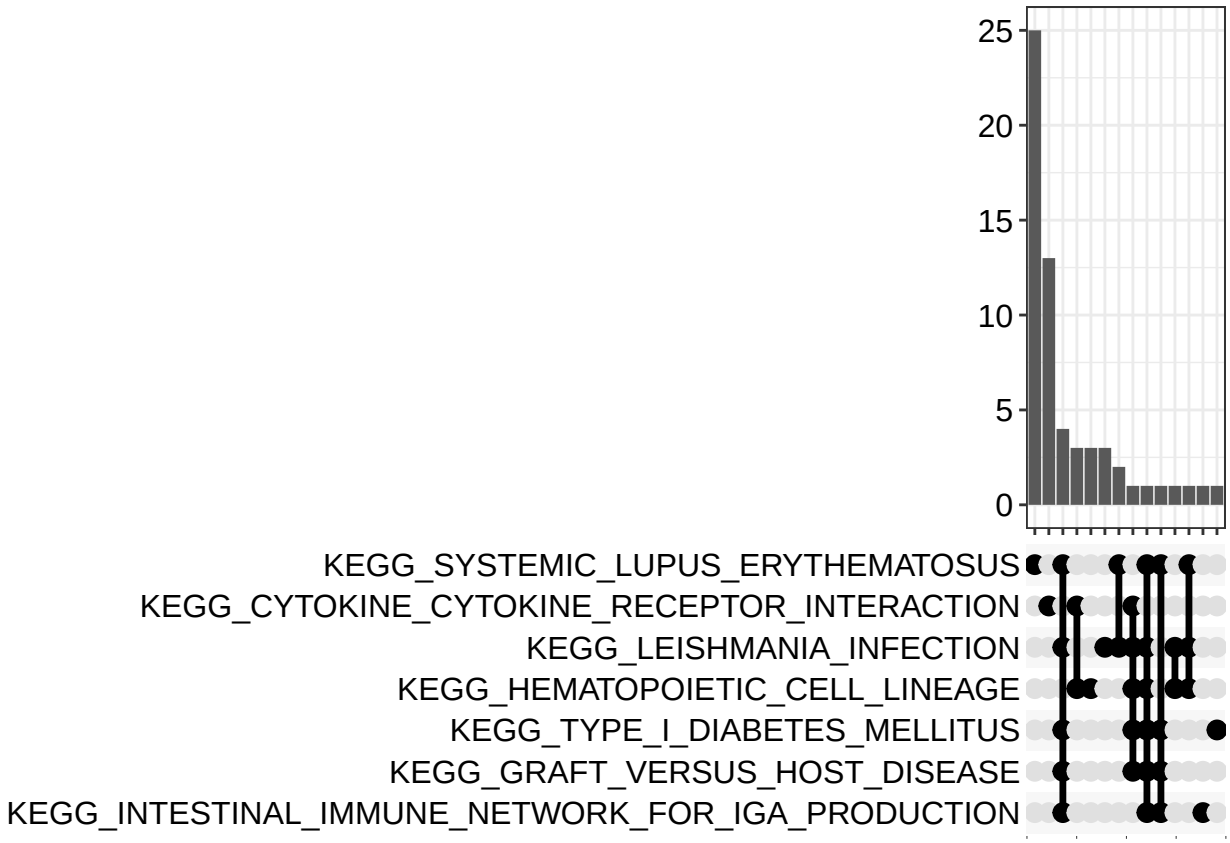



```r
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=1,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))

# Step 3: Determine gene sets.
#msigdbr_species()
hs_msigdb_df <- msigdbr(species = "Homo sapiens")
#head(hs_msigdb_df)

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
```

```
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```




```
upset_plot <- enrichplot::upsetplot(kegg_ora_results)
upset_plot
```

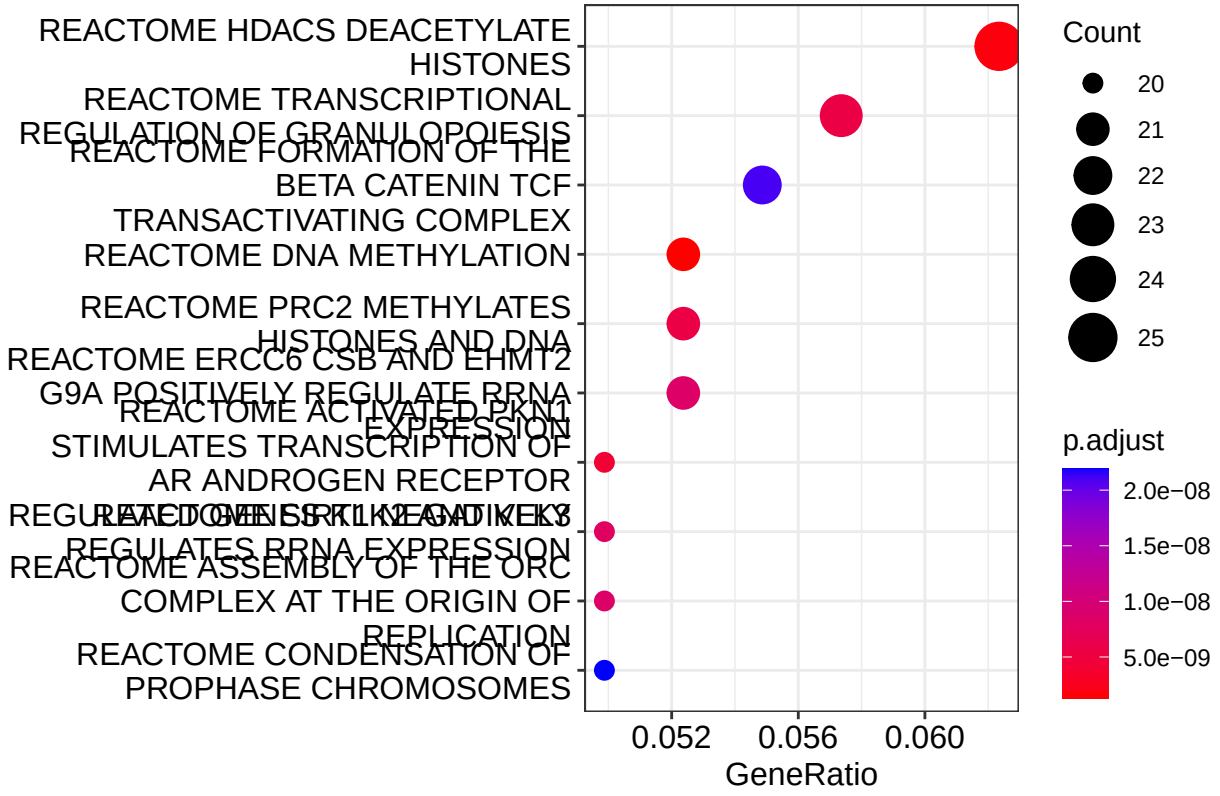



```
# Step 6: EXERCISE: alternatives to KEGG?
msigdbr_collections()
```

```
## # A tibble: 23 x 3
##    gs_cat gs_subcat         num_genesets
##    <chr>  <chr>                    <int>
##  1 C1     ""                         299
##  2 C2     "CGP"                     3384
##  3 C2     "CP"                        29
##  4 C2     "CP:BIOCARTA"              292
##  5 C2     "CP:KEGG"                  186
##  6 C2     "CP:PID"                   196
##  7 C2     "CP:REACTOME"             1615
##  8 C2     "CP:WIKIPATHWAYS"          664
##  9 C3     "MIR:MIRDB"               2377
## 10 C3     "MIR:MIR_Legacy"           221
## # i 13 more rows
```

```
# Step 1: determine genes of interest.
diff_table <- topTable(fit2,coef=1,p.value=0.01,number=10000)
genes_dif<- rownames(diff_table)

# Step 2: determine background.
background_set <- unique(rownames(v))
```

```r
# Step 3: Determine gene sets.
hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:REACTOME" # This is because we only want KEGG pathways
  )

# Step 4: conduct ORA.
kegg_ora_results <- enricher(
  gene = genes_dif, # A vector of your genes of interest
  pvalueCutoff = 0.1, # Can choose a FDR cutoff
  pAdjustMethod = "BH", # Method to be used for multiple testing correction
  universe = background_set, # A vector containing your background set genes
  # The pathway information should be a data frame with a term name or
  # identifier and the gene identifiers
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(kegg_ora_results)
enrich_plot
```



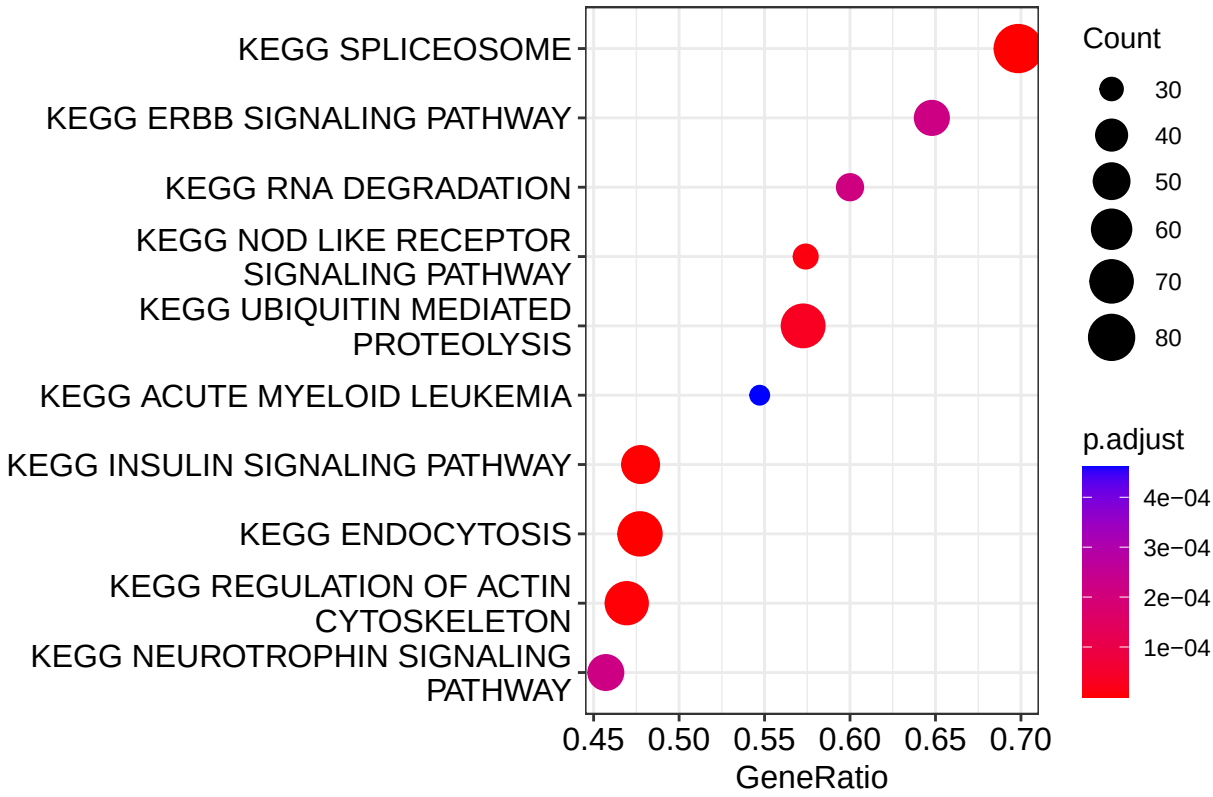

```
upset_plot <- enrichplot::upsetplot(kegg_ora_results)
upset_plot
```

REACTOME_HDACS
REACTOME_TRANSCRIPTIONAL_REGULAT
REACTOME_FORMATION_OF_THE_BETA_CATENIN_TCF_TR
REACTOME_ERCC6_CSB_AND_EHMT2_G9A_POSITIVELY_REGU
REAC
REACTOME_PRC2_METHYLA
N1_STIMULATES_TRANSCRIPTION_OF_AR_ANDROGEN_RECEPTOR_REGULATEI
REACTOME_ASSEMBLY_OF_THE_ORC_COMPLEX_AT_THE
REACTOME_CONDENSATION_OF_PF
REACTOME_SIRT1_NEGATIVELY_REGUI

GSEA

```
# Step 1: determine genes of interest.
diff_table_all <- topTable(fit2,coef=1,p.value=1,number=nrow(logCPM))

# Step 2: determine background.

# Step 3: Determine gene sets.
hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct GSEA

list_ordered <- diff_table_all[,"B"]
names(list_ordered) <- rownames(diff_table_all)


gsea_results <- GSEA(
  geneList = list_ordered, # Ordered ranked gene list
  minGSSize = 25, # Minimum gene set size
  maxGSSize = 500, # Maximum gene set set
  pvalueCutoff = 0.05, # p-value cutoff
```

```
  eps = 0, # Boundary for calculating the p value
  seed = TRUE, # Set seed to make results reproducible
  pAdjustMethod = "BH", # Benjamini-Hochberg correction
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(gsea_results)
enrich_plot
```




```
# Step 5: Visualize / explore
head(gsea_results@result)
```

```
##                                                                                    ID
## KEGG_SPLICEOSOME                                                     KEGG_SPLICEOSOME
## KEGG_ENDOCYTOSIS                                                     KEGG_ENDOCYTOSIS
## KEGG_INSULIN_SIGNALING_PATHWAY                         KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON           KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS           KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
##                                                                            Description
```

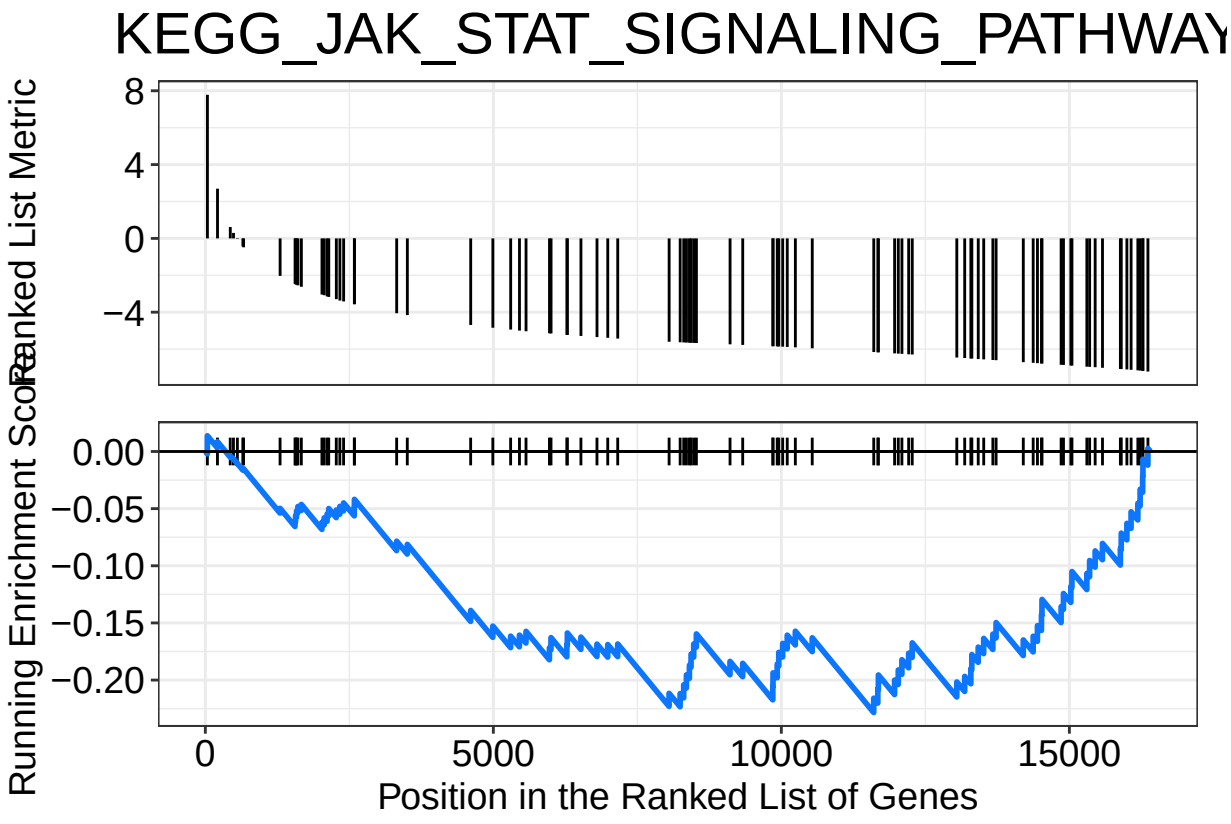
```
## KEGG_SPLICEOSOME                                          KEGG_SPLICEOSOME
## KEGG_ENDOCYTOSIS                                          KEGG_ENDOCYTOSIS
## KEGG_INSULIN_SIGNALING_PATHWAY                    KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON       KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS          KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
##                                          setSize enrichmentScore       NES
## KEGG_SPLICEOSOME                             126      -0.4447394 -2.921911
## KEGG_ENDOCYTOSIS                             153      -0.3323859 -2.245047
## KEGG_INSULIN_SIGNALING_PATHWAY               111      -0.3653569 -2.353010
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON        147      -0.3283742 -2.211618
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY      54      -0.4463884 -2.472979
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS          124      -0.3246401 -2.126791
##                                                pvalue     p.adjust       qvalue
## KEGG_SPLICEOSOME                          3.867506e-15 4.834383e-13 2.320504e-13
## KEGG_ENDOCYTOSIS                          8.597844e-09 5.373652e-07 2.579353e-07
## KEGG_INSULIN_SIGNALING_PATHWAY            5.831834e-08 2.429931e-06 1.166367e-06
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON     1.235483e-07 3.860884e-06 1.853225e-06
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY 4.441754e-07 1.110438e-05 5.330104e-06
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS       1.637529e-06 3.411518e-05 1.637529e-05
##                                          rank                  leading_edge
## KEGG_SPLICEOSOME                         5285 tags=70%, list=32%, signal=48%
## KEGG_ENDOCYTOSIS                         4334 tags=48%, list=26%, signal=35%
## KEGG_INSULIN_SIGNALING_PATHWAY           3730 tags=48%, list=23%, signal=37%
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON    4763 tags=47%, list=29%, signal=34%
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY 4748 tags=57%, list=29%, signal=41%
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS      2820 tags=57%, list=17%, signal=48%
##
## KEGG_SPLICEOSOME                         10286/6625/5093/22916/84991/3304/84321/8559/9410/51691/5150
## KEGG_ENDOCYTOSIS
## KEGG_INSULIN_SIGNALING_PATHWAY
## KEGG_REGULATION_OF_ACTIN_CYTOSKELETON
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_UBIQUITIN_MEDIATED_PROTEOLYSIS
```

```
gsea_result_df <- data.frame(gsea_results@result)
gsea_result_df %>%
  # This returns the 3 rows with the largest NES values
  dplyr::slice_max(NES, n = 3)
```

```
##                                                                    ID
## KEGG_ALZHEIMERS_DISEASE                           KEGG_ALZHEIMERS_DISEASE
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY
## KEGG_LEISHMANIA_INFECTION                       KEGG_LEISHMANIA_INFECTION
##                                                           Description
## KEGG_ALZHEIMERS_DISEASE                           KEGG_ALZHEIMERS_DISEASE
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY
## KEGG_LEISHMANIA_INFECTION                       KEGG_LEISHMANIA_INFECTION
##                                        setSize enrichmentScore       NES
## KEGG_ALZHEIMERS_DISEASE                    139      -0.2168379 -1.442677
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY      91      -0.2536014 -1.569666
## KEGG_LEISHMANIA_INFECTION                   63      -0.2735039 -1.572445
##                                           pvalue   p.adjust     qvalue rank
## KEGG_ALZHEIMERS_DISEASE                0.018511178 0.04059469 0.01948545 5666
```

```
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY 0.008769623 0.02192406 0.01052355 3791
## KEGG_LEISHMANIA_INFECTION              0.014215722 0.03173152 0.01523113 4382
##                                                                    leading_edge
## KEGG_ALZHEIMERS_DISEASE                tags=43%, list=35%, signal=29%
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY tags=41%, list=23%, signal=31%
## KEGG_LEISHMANIA_INFECTION              tags=38%, list=27%, signal=28%
##
## KEGG_ALZHEIMERS_DISEASE                7385/1347/5664/23621/1329/102/4512/572/4694/4698/23385/513/153
## KEGG_T_CELL_RECEPTOR_SIGNALING_PATHWAY
## KEGG_LEISHMANIA_INFECTION
```

```
most_positive_nes_plot <- enrichplot::gseaplot(
  gsea_results,
  geneSetID = "KEGG_JAK_STAT_SIGNALING_PATHWAY",
  title = "KEGG_JAK_STAT_SIGNALING_PATHWAY",
  color.line = "#0d76ff"
)
most_positive_nes_plot
```

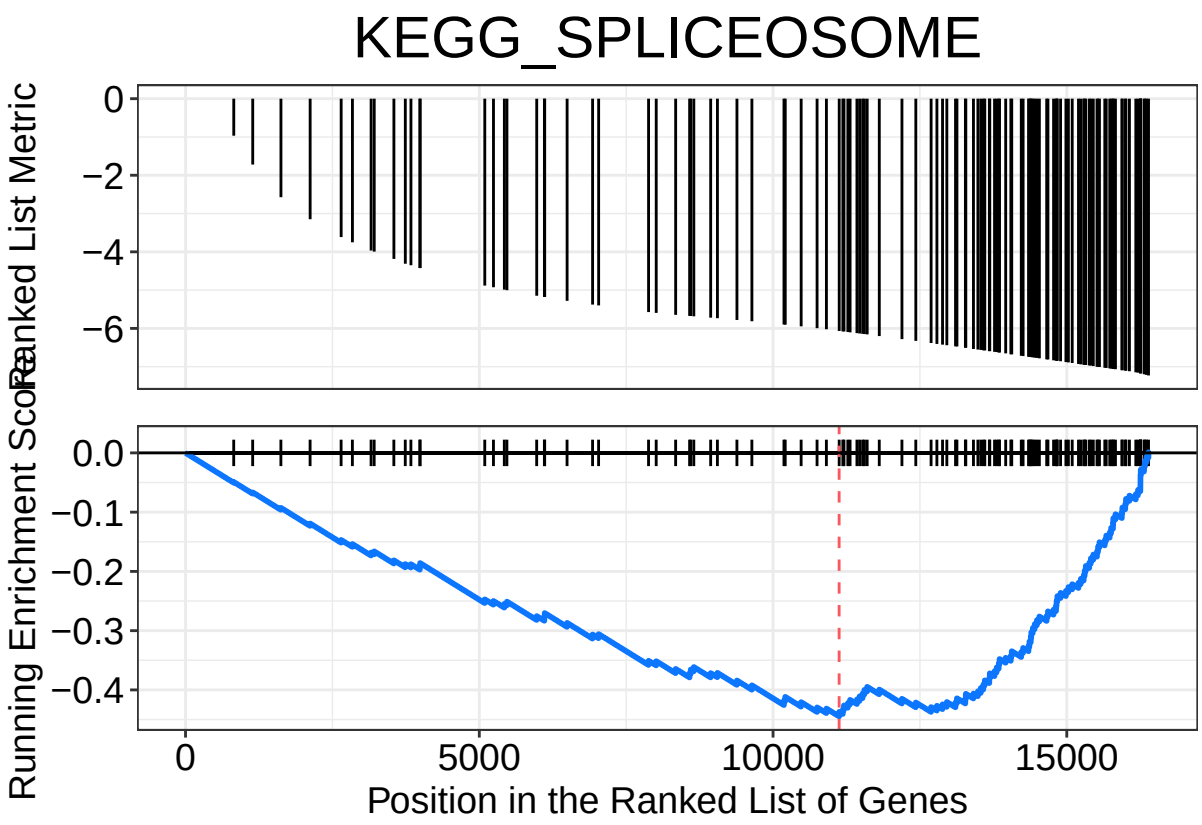



```
gsea_result_df %>%
  # Return the 3 rows with the smallest (most negative) NES values
  dplyr::slice_min(NES, n = 3)
```

```
##                                                                    ID
## KEGG_SPLICEOSOME                                     KEGG_SPLICEOSOME
```

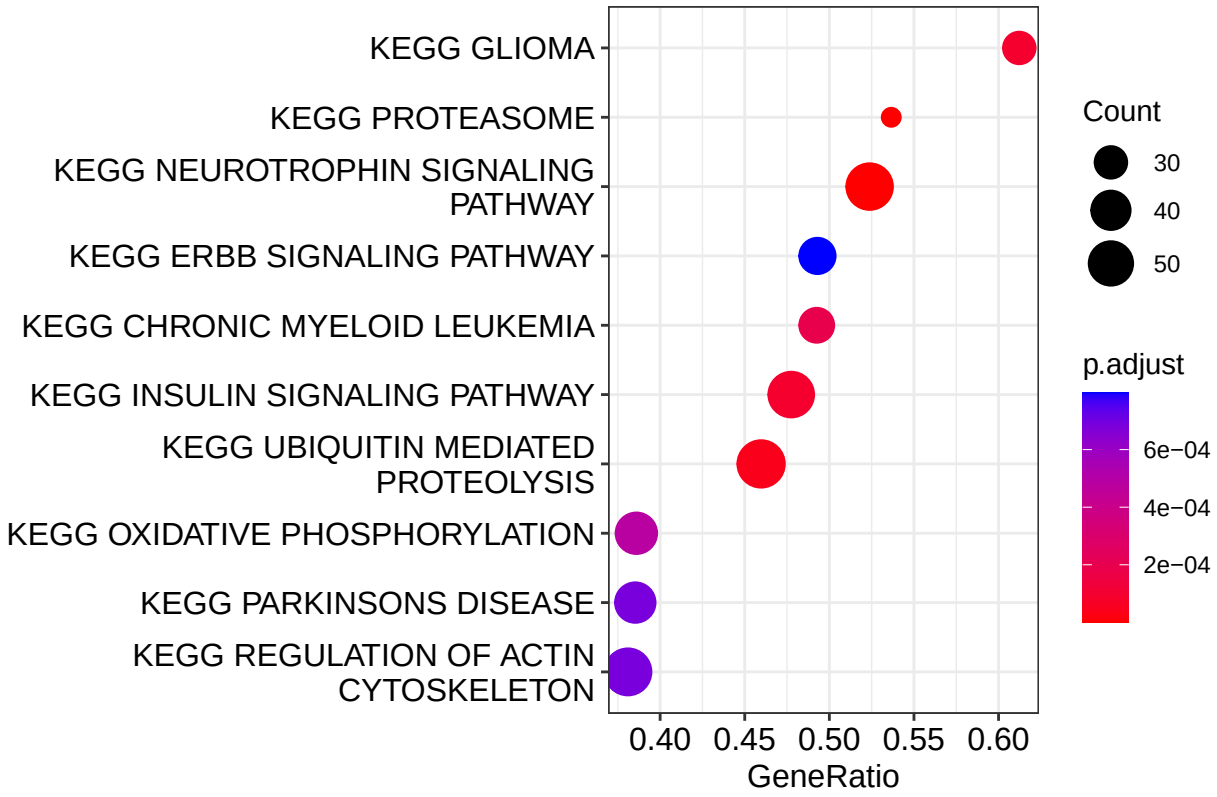
```
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_INSULIN_SIGNALING_PATHWAY                     KEGG_INSULIN_SIGNALING_PATHWAY
##                                                                        Description
## KEGG_SPLICEOSOME                                                    KEGG_SPLICEOSOME
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_INSULIN_SIGNALING_PATHWAY                     KEGG_INSULIN_SIGNALING_PATHWAY
##                                        setSize enrichmentScore       NES
## KEGG_SPLICEOSOME                           126      -0.4447394 -2.921911
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY    54      -0.4463884 -2.472979
## KEGG_INSULIN_SIGNALING_PATHWAY             111      -0.3653569 -2.353010
##                                              pvalue     p.adjust       qvalue
## KEGG_SPLICEOSOME                       3.867506e-15 4.834383e-13 2.320504e-13
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY 4.441754e-07 1.110438e-05 5.330104e-06
## KEGG_INSULIN_SIGNALING_PATHWAY           5.831834e-08 2.429931e-06 1.166367e-06
##                                        rank                  leading_edge
## KEGG_SPLICEOSOME                       5285 tags=70%, list=32%, signal=48%
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY 4748 tags=57%, list=29%, signal=41%
## KEGG_INSULIN_SIGNALING_PATHWAY           3730 tags=48%, list=23%, signal=37%
##
## KEGG_SPLICEOSOME                       10286/6625/5093/22916/84991/3304/84321/8559/9410/51691/5150
## KEGG_NOD_LIKE_RECEPTOR_SIGNALING_PATHWAY
## KEGG_INSULIN_SIGNALING_PATHWAY
```

```
most_negative_nes_plot <- enrichplot::gseaplot(
  gsea_results,
  geneSetID = "KEGG_SPLICEOSOME",
  title = "KEGG_SPLICEOSOME",
  color.line = "#0d76ff"
)
most_negative_nes_plot
```



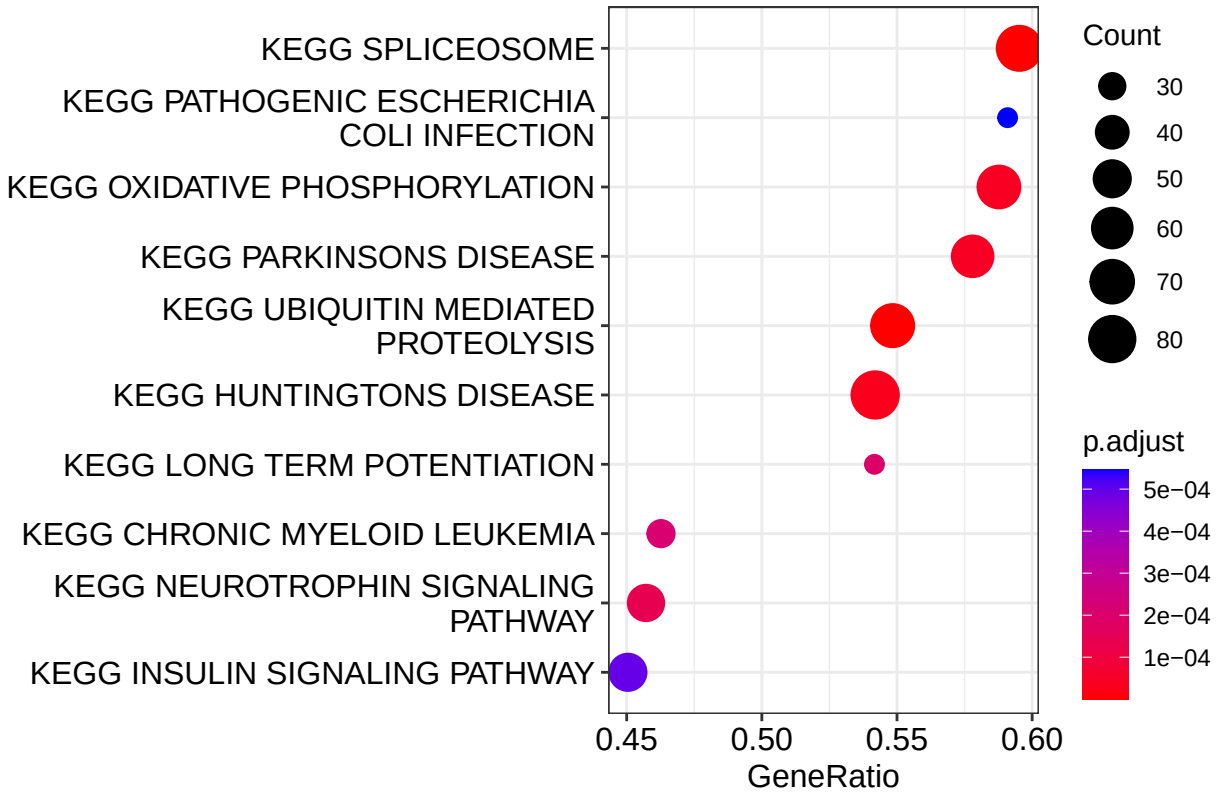

```r
# Step 1: determine genes of interest.
diff_table_all <- topTable(fit2,coef=2,p.value=1,number=nrow(logCPM))

# Step 2: determine background.

# Step 3: Determine gene sets.
hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )

# Step 4: conduct GSEA

list_ordered <- diff_table_all[,"B"]
names(list_ordered) <- rownames(diff_table_all)


gsea_results <- GSEA(
  geneList = list_ordered, # Ordered ranked gene list
  minGSSize = 25, # Minimum gene set size
  maxGSSize = 500, # Maximum gene set set
  pvalueCutoff = 0.05, # p-value cutoff
  eps = 0, # Boundary for calculating the p value
```

```
  seed = TRUE, # Set seed to make results reproducible
  pAdjustMethod = "BH", # Benjamini-Hochberg correction
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(gsea_results)
enrich_plot
```



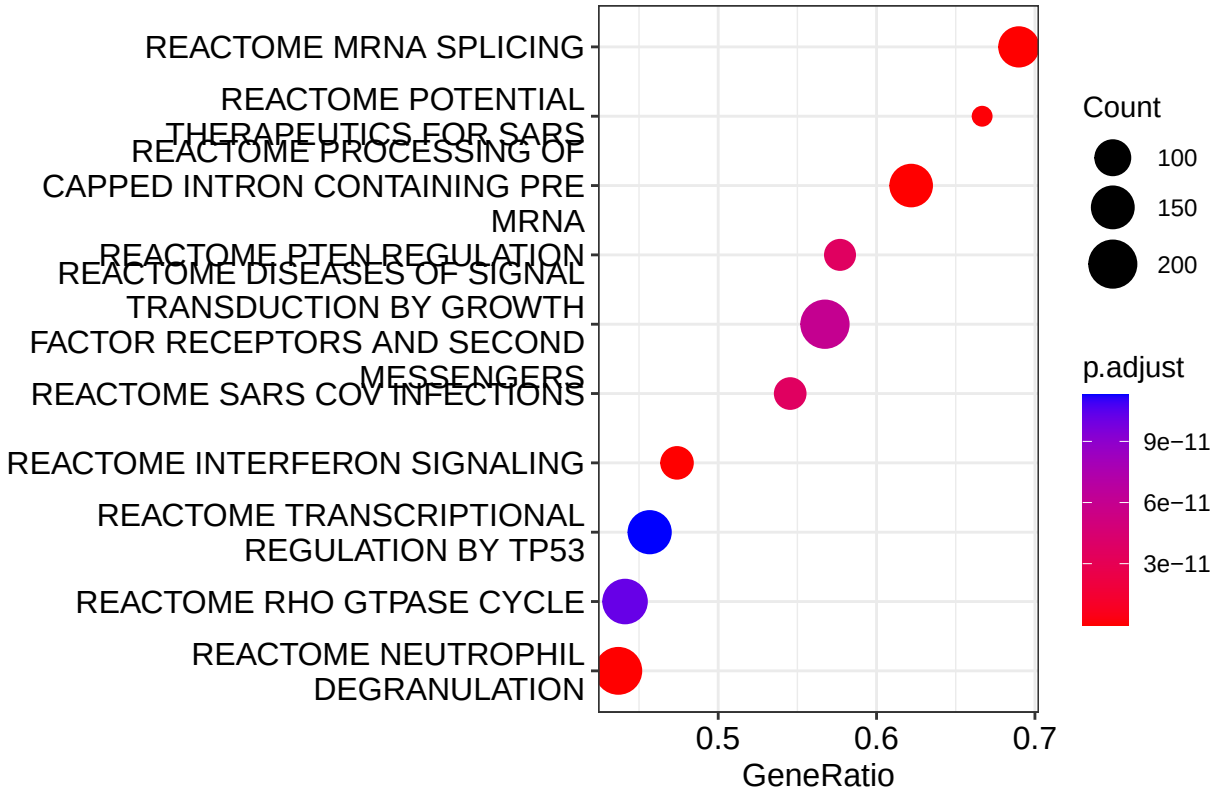

```
# Step 1: determine genes of interest.
diff_table_all <- topTable(fit2,coef=3,p.value=1,number=nrow(logCPM))

# Step 2: determine background.

# Step 3: Determine gene sets.
hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
```

```
  )

# Step 4: conduct GSEA

list_ordered <- diff_table_all[,"B"]
names(list_ordered) <- rownames(diff_table_all)


gsea_results <- GSEA(
  geneList = list_ordered, # Ordered ranked gene list
  minGSSize = 25, # Minimum gene set size
  maxGSSize = 500, # Maximum gene set set
  pvalueCutoff = 0.05, # p-value cutoff
  eps = 0, # Boundary for calculating the p value
  seed = TRUE, # Set seed to make results reproducible
  pAdjustMethod = "BH", # Benjamini-Hochberg correction
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(gsea_results)
enrich_plot
```




```
# Step 6: EXERCISE: alternatives to KEGG?

# Step 1: determine genes of interest.
diff_table_all <- topTable(fit2,coef=1,p.value=1,number=nrow(logCPM))

# Step 2: determine background.

# Step 3: Determine gene sets.

hs_msigdb_df <- msigdbr(species = "Homo sapiens")

hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:REACTOME" # This is because we only want KEGG pathways
  )

# Step 4: conduct GSEA

list_ordered <- diff_table_all[,"B"]
names(list_ordered) <- rownames(diff_table_all)


gsea_results <- GSEA(
  geneList = list_ordered, # Ordered ranked gene list
  minGSSize = 25, # Minimum gene set size
```

```
  maxGSSize = 500, # Maximum gene set set
  pvalueCutoff = 0.05, # p-value cutoff
  eps = 0, # Boundary for calculating the p value
  seed = TRUE, # Set seed to make results reproducible
  pAdjustMethod = "BH", # Benjamini-Hochberg correction
  TERM2GENE = dplyr::select(
    hs_kegg_df,
    gs_name,
    human_entrez_gene
  )
)

# Step 5: Visualize / explore
enrich_plot <- enrichplot::dotplot(gsea_results)
enrich_plot
```



GSEA vs ORA

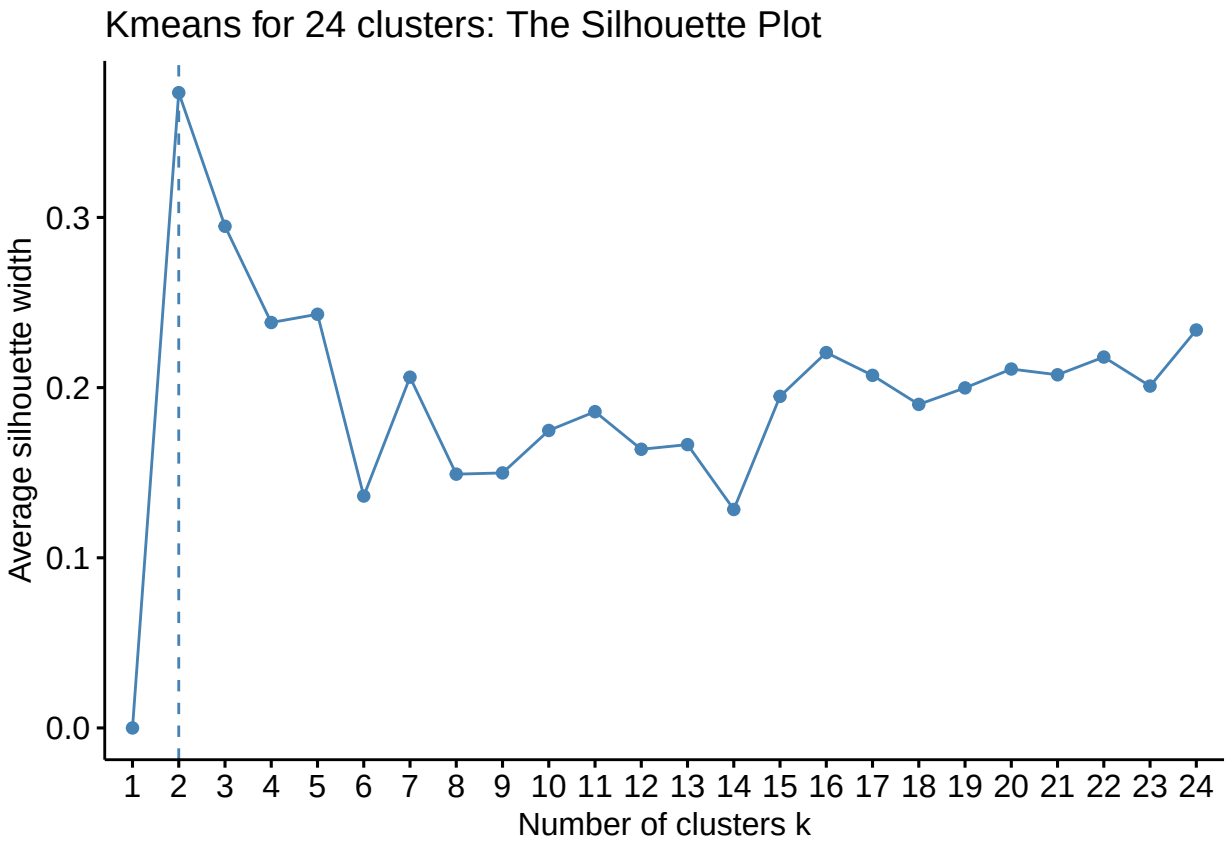
ORA	GSEA
Which pathways are overrepresented in my list of significant genes? Background genes	These changes I see between treatment and control are similar to which known list of changes? No background genes.

ORA	GSEA
Is the gene in the dataset?Is the gene DE?	Is the gene in the dataset?What position it has?

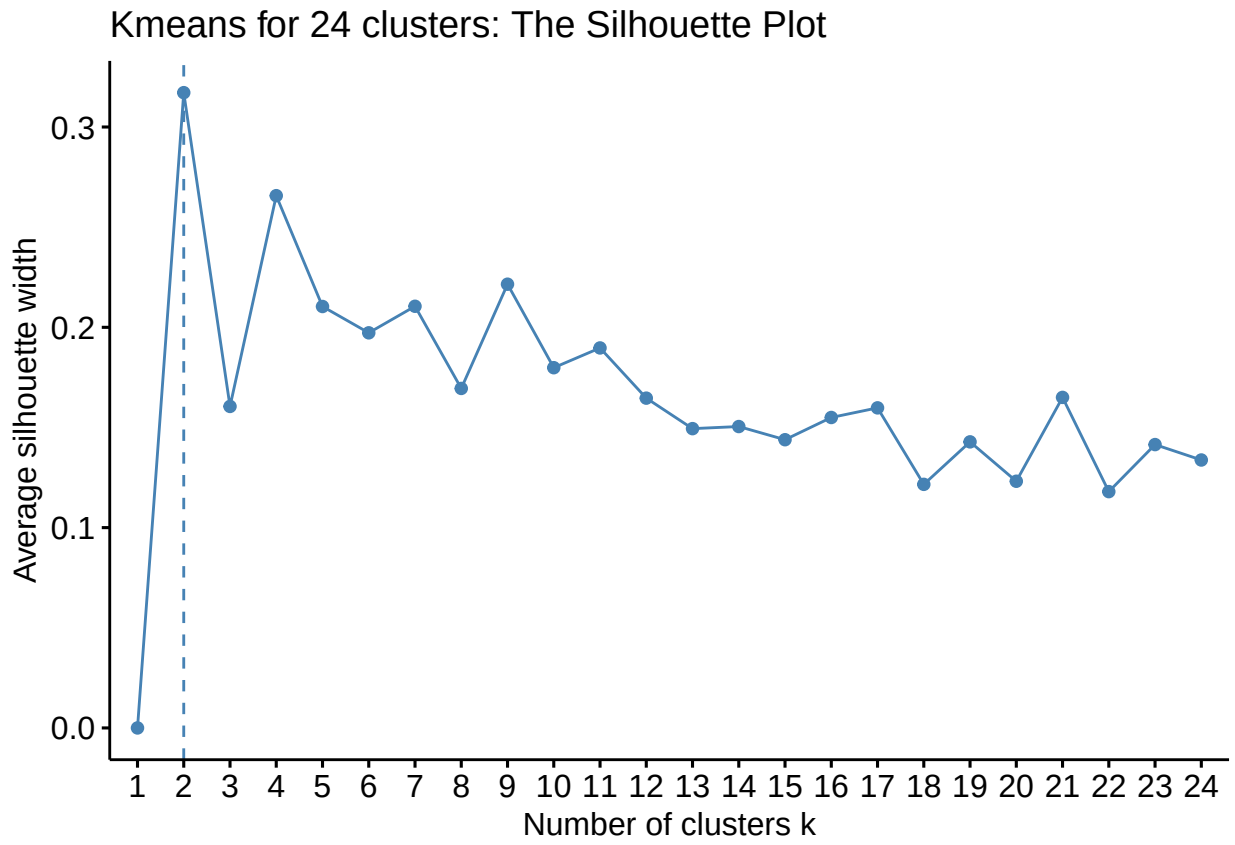
Lets conduct GeneSetCluster.

```
# Healthy vs Group Covid19
# We prepare a function from the previous analysis

# Healthy vs Covid19AI
Diff_HvsAI <- topTable(fit2,coef=1,p.value=1,number=nrow(logCPM))
# Healthy vs Covid196Mo
Diff_Hvs6Mo <- topTable(fit2,coef=3,p.value=1,number=nrow(logCPM))


hs_msigdb_df <- msigdbr(species = "Homo sapiens")
hs_kegg_df <- hs_msigdb_df %>%
  dplyr::filter(
    gs_cat == "C2", # This is to filter only to the C2 curated gene sets
    gs_subcat == "CP:KEGG" # This is because we only want KEGG pathways
  )


doGSEA <- function(diff_table) {
  list_ordered <- diff_table[,"B"]
  names(list_ordered) <- rownames(diff_table)

  return(GSEA(
    geneList = list_ordered, # Ordered ranked gene list
    minGSSize = 25, # Minimum gene set size
    maxGSSize = 500, # Maximum gene set set
    pvalueCutoff = 0.05, # p-value cutoff
    eps = 0, # Boundary for calculating the p value
    seed = TRUE, # Set seed to make results reproducible
    pAdjustMethod = "BH", # Benjamini-Hochberg correction
    TERM2GENE = dplyr::select(
      hs_kegg_df,
      gs_name,
      human_entrez_gene
    )
  ))
}

GSEA_HvsAI <- doGSEA(Diff_HvsAI)
GSEA_Hvs6Mo <- doGSEA(Diff_Hvs6Mo)

path <- "/Users/gonzalac/Desktop/PhD_2nd/BESE398_Pipelines/Bioinformatics_Pipelines/W2"

write.csv(GSEA_HvsAI, file = paste0(path, "/GSEA_HvsAI.csv"), row.names = FALSE)
write.csv(GSEA_Hvs6Mo, file = paste0(path, "/GSEA_Hvs6Mo.csv"), row.names = FALSE)
```

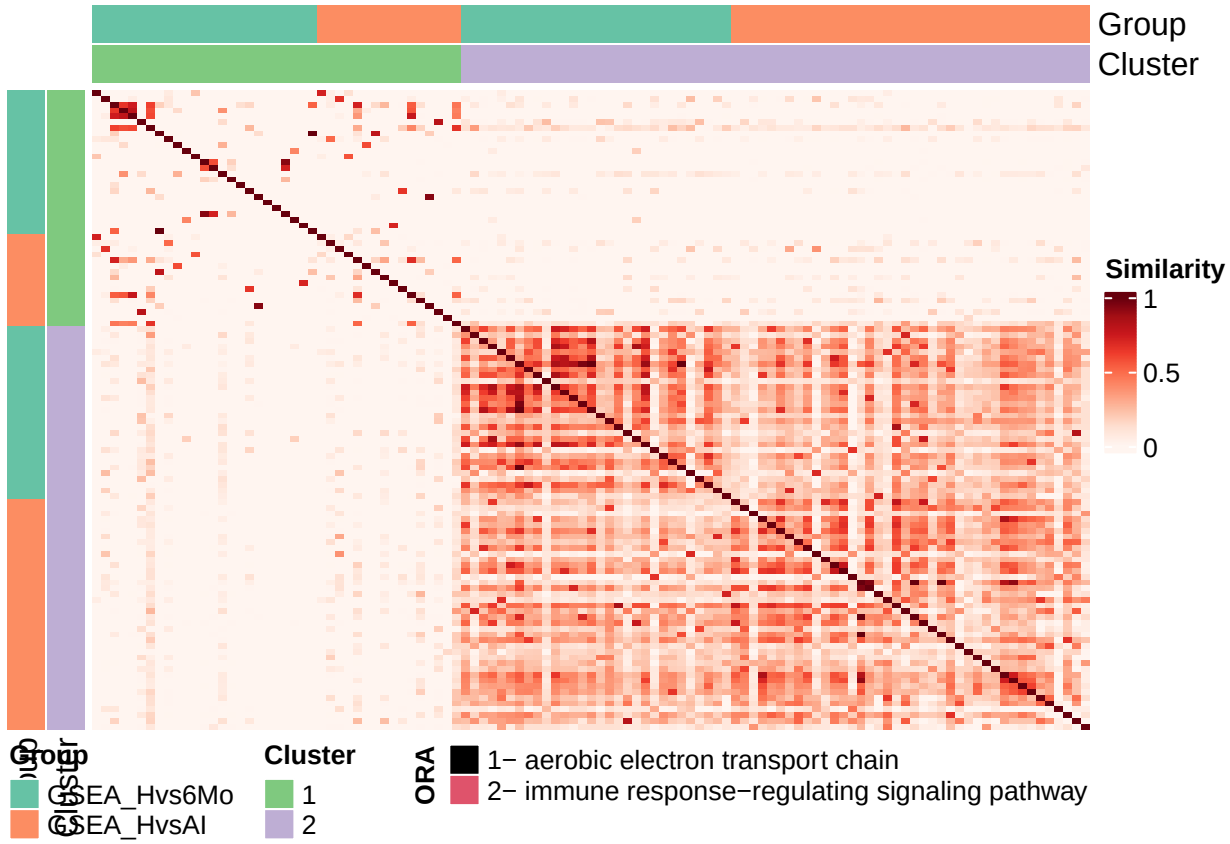
```r
library(GeneSetCluster)
path <- "/Users/gonzalac/Desktop/PhD_2nd/BESE398_Pipelines/Bioinformatics_Pipelines/W2"
GSEA.files <- list.files("./", pattern = ".csv")


# Load the data and create Pathway object
# Automatically for GSEA, GREAT or IPA
GSEA.Object1 <- LoadGeneSets(file_location = GSEA.files,
                             groupnames= c("GSEA_Hvs6Mo", "GSEA_HvsAI"), # names of the groups
                             P.cutoff = 0.05, # cut off the p.adjust
                             Mol.cutoff = 15, # minimum number of genes per pathway
                             Source = "GSEA", # the analysis (GSEA, GREAT or IPA)
                             structure = "ENTREZID", # Gene type (SYMBOL, ENTREZID, ENSEMBLID)
                             Organism = "org.Hs.eg.db", # database: Homo Sapiens or Mus musculus
                             seperator = "/") # the separator used for listing genes

# IMPORTANT when created manually, it is assumed that the pathways have been filtered by p-value and mi
# Make sure you have filtered your data
GSEA.Object1Manual <- ObjectCreator(Pathways = c(GSEA_HvsAI@result$ID,
                                                 GSEA_Hvs6Mo@result$ID),
                                    Molecules = c(GSEA_HvsAI@result$core_enrichment,
                                                  GSEA_Hvs6Mo@result$core_enrichment),
                                    Groups = c(rep("GSEA_HvsAI", times=nrow(GSEA_HvsAI@result)),
                                               rep("GSEA_Hvs6Mo", times=nrow(GSEA_Hvs6Mo@result))),
                                    Pvalues = c(GSEA_HvsAI@result$p.adjust,  # optional
                                                GSEA_Hvs6Mo@result$p.adjust),
                                    enrichmentScore = c(GSEA_HvsAI@result$NES, # optional
                                                        GSEA_Hvs6Mo@result$NES),
                                    structure = "ENTREZID", Type = "", sep = "/",
                                    Source = "GSEA", organism = "org.Hs.eg.db")

GSEA.Object2 <- CombineGeneSets(Object = GSEA.Object1,
                                combineMethod = "Standard", threads = 8)


OptimalGeneSets(Object = GSEA.Object2,
                uniquePathway = FALSE, # consider all the pathways (also repeated) or the unique pathwa
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



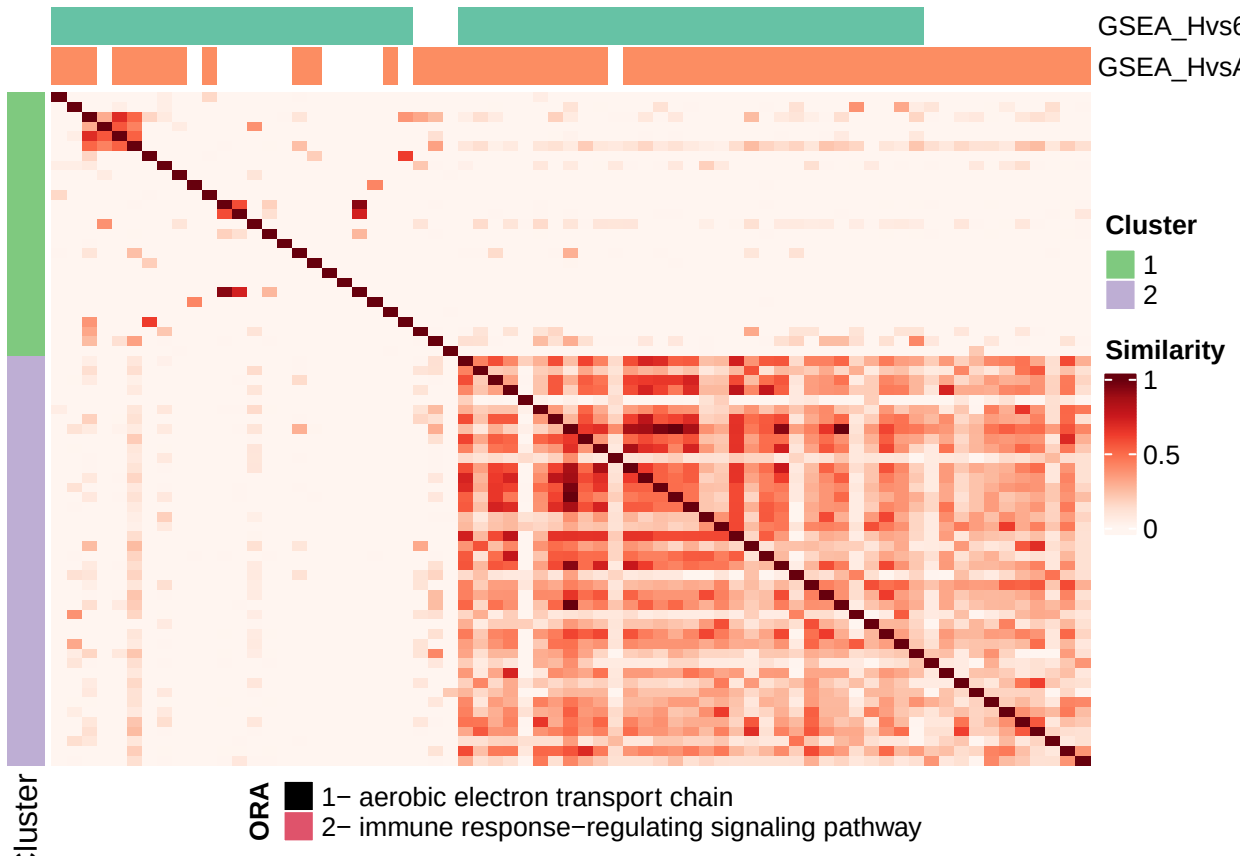

```
OptimalGeneSets(Object = GSEA.Object2,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



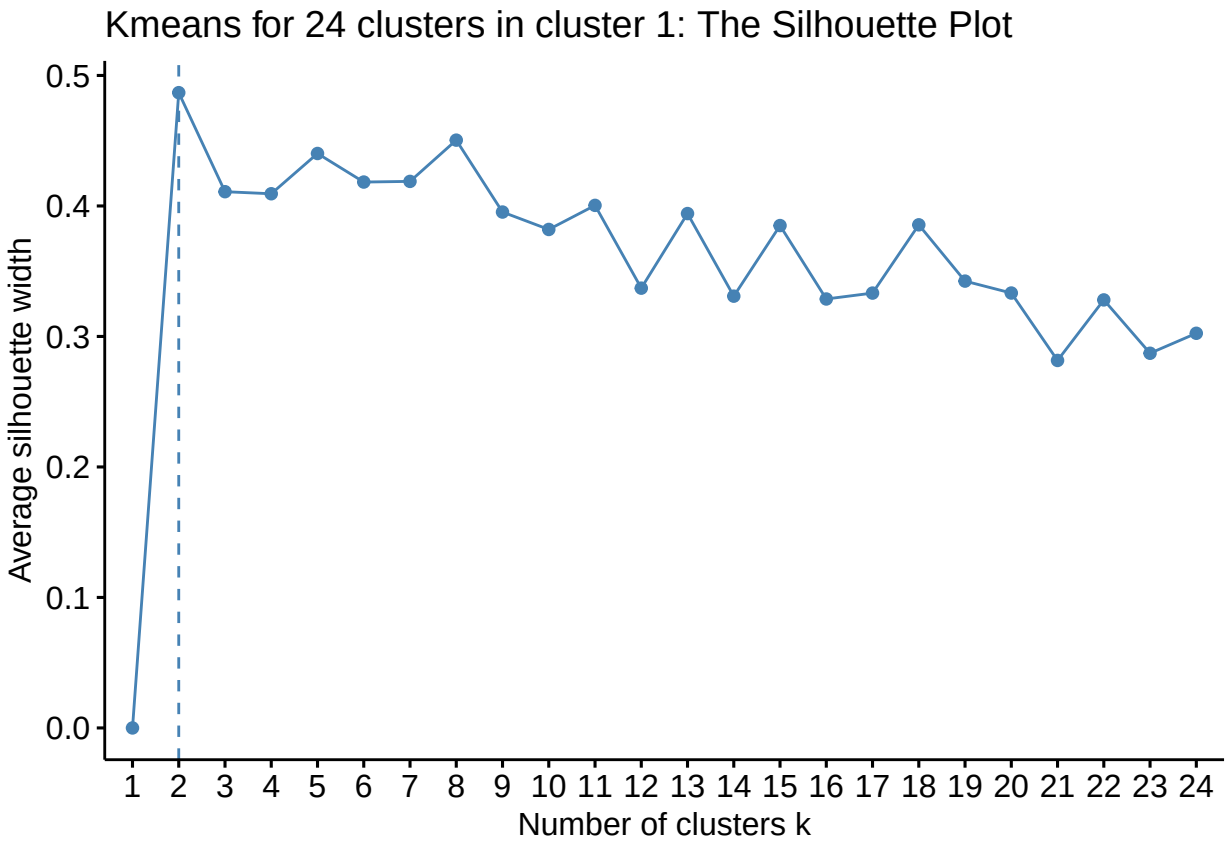

```
# in both cases the optimal cluster is 2

GSEA.Object3 <- ClusterGeneSets(Object = GSEA.Object2,
                                clusters = 2, # consider all the pathways (also repeated) or the unique
                                method = "Hierarchical", # Hierarchical clustering or kmeans
                                order = "cluster",
                                molecular.signature = "All")

# plot results for both all pathways and unique pathways
plotnounique <- PlotGeneSets(GSEA.Object3,
                             uniquePathways = FALSE,
                             wordcloud = FALSE, # wordcloud only supported for GO terms
                             doORA = T) # do ora per cluster
```



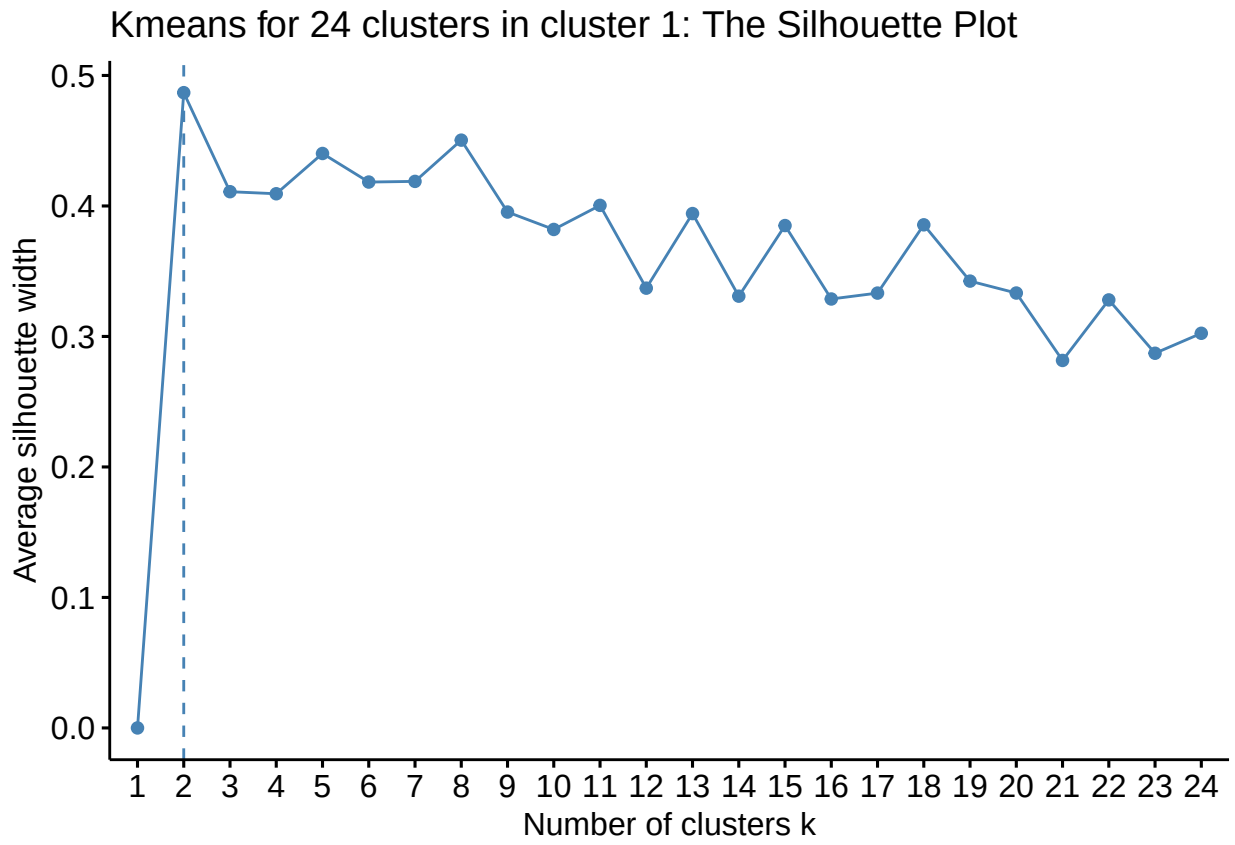

```
plotunique <- PlotGeneSets(GSEA.Object3,
                           uniquePathways = TRUE,
                           wordcloud = FALSE, # wordcloud only supported for GO terms
                           doORA = T) # do ora per cluster
```



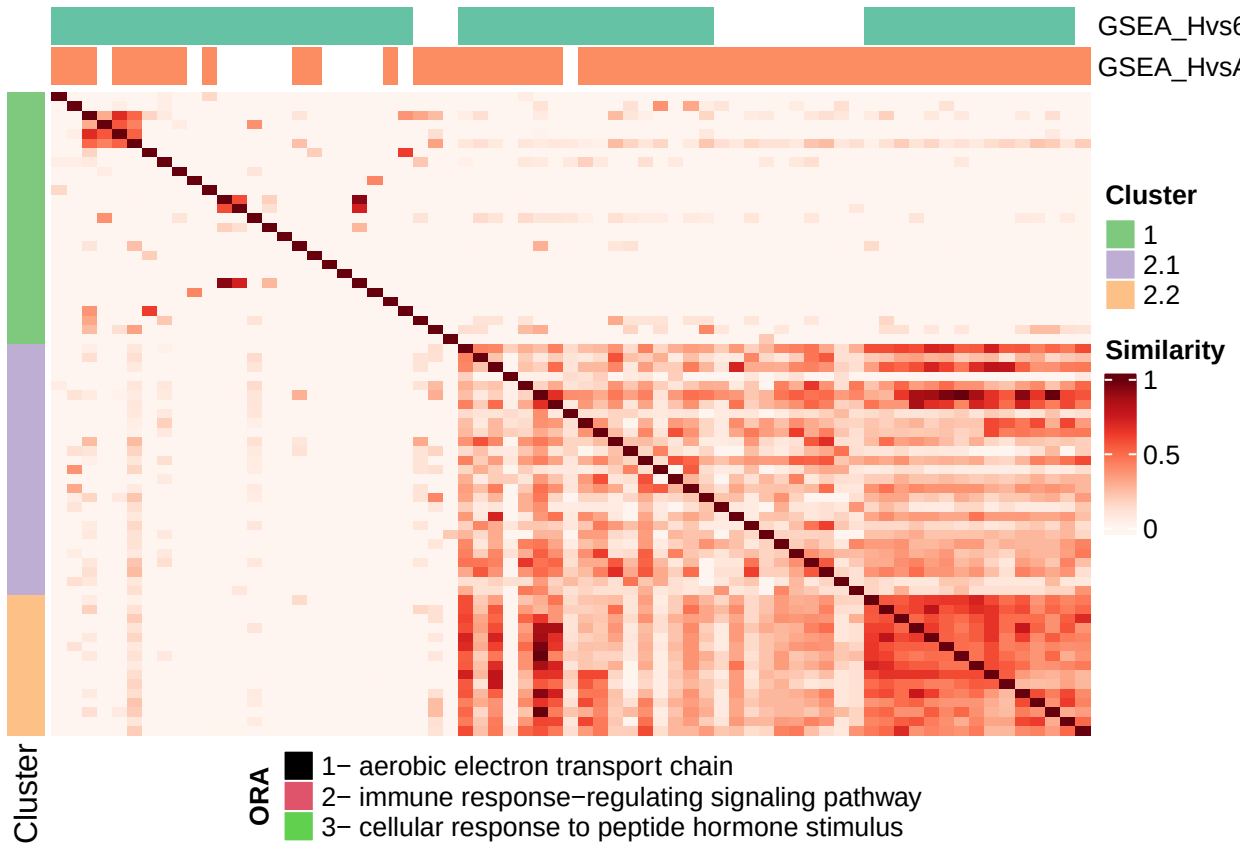

```
# let's say we are interested in exploring cluster 2 in plotunique. Lets break up this cluste for furth

plotoptimalcluster2 <- OptimalGeneSets(Object = GSEA.Object3,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                cluster = 2, # which cluster
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



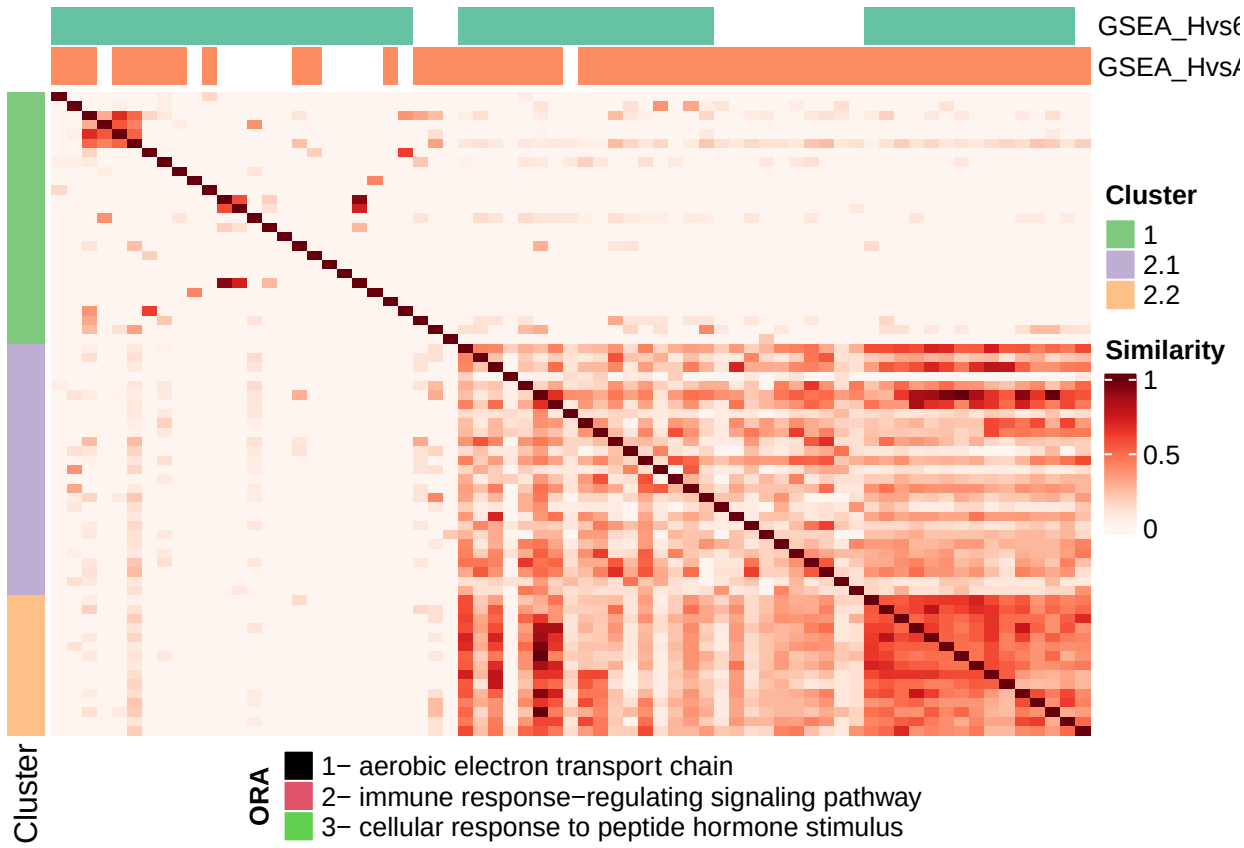

```
plotoptimalcluster2 # optimal 2 break up cluster 2 in 2 clusters
```



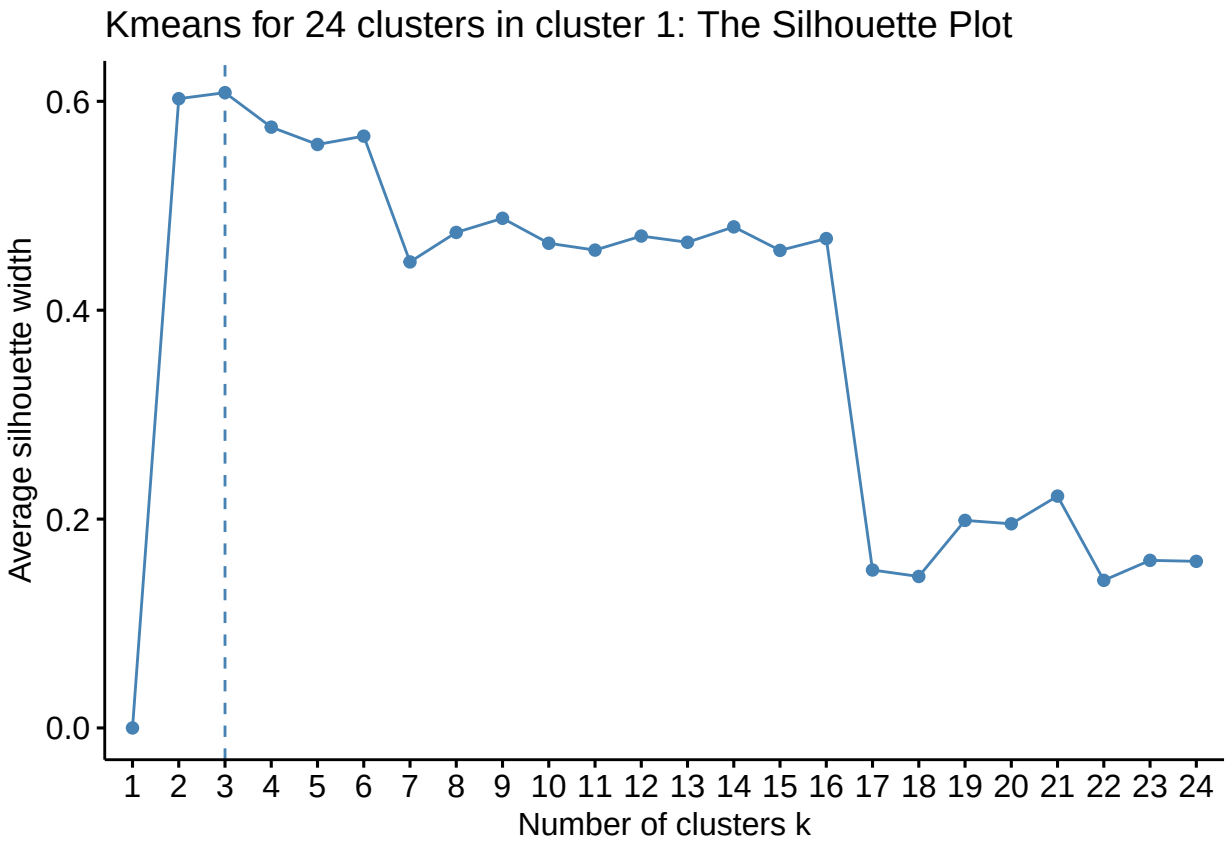

```
GSEA.Object3breakup <- BreakUpCluster(GSEA.Object3,
                                      breakup.cluster = 2, # which cluster
                                      sub.cluster = 2, # in how many cluster split up
                                      uniquePathways = TRUE) # conside unique pathways

plotuniquebreakup <- PlotGeneSets(GSEA.Object3breakup,
                                  uniquePathways = TRUE,
                                  wordcloud = FALSE, # wordcloud only supported for GO terms
                                  doORA = T) # do ora per cluster
```



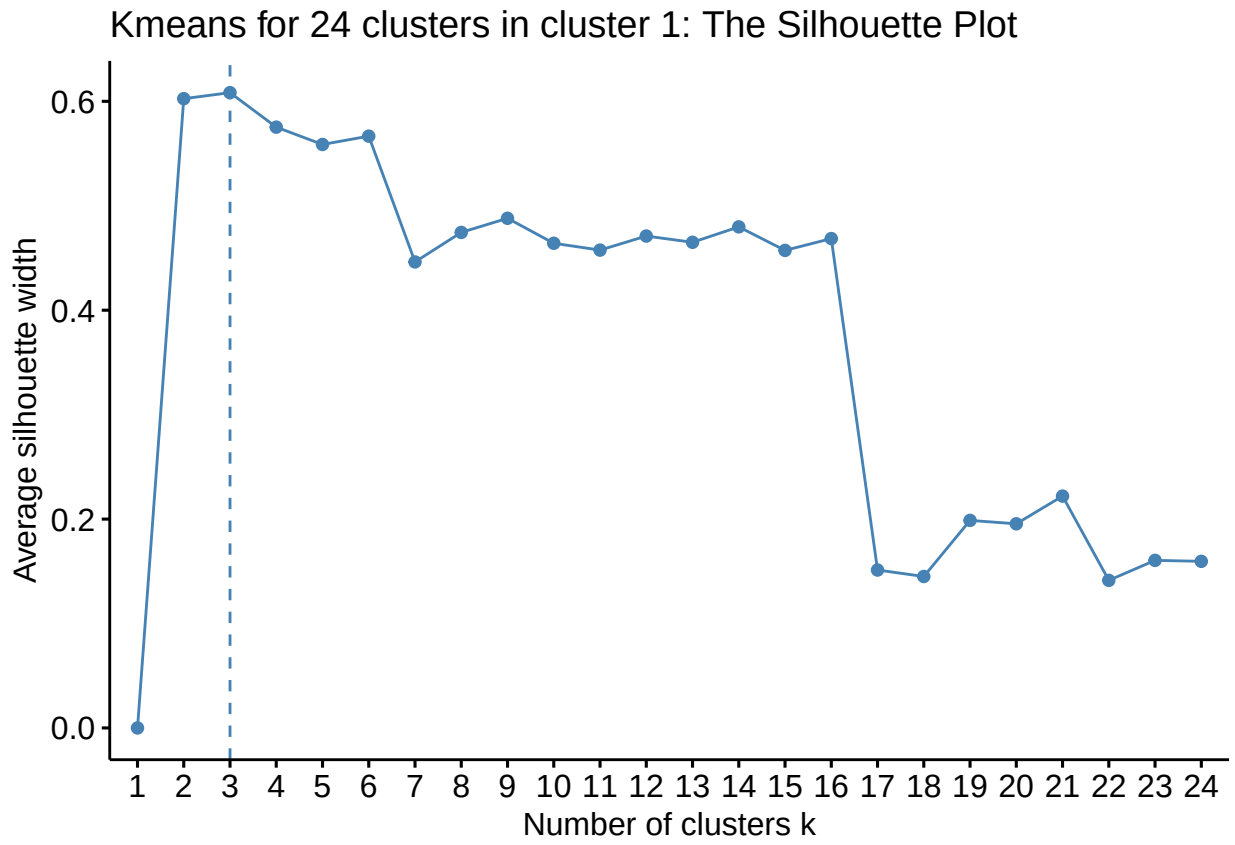

```
plotuniquebreakup
```



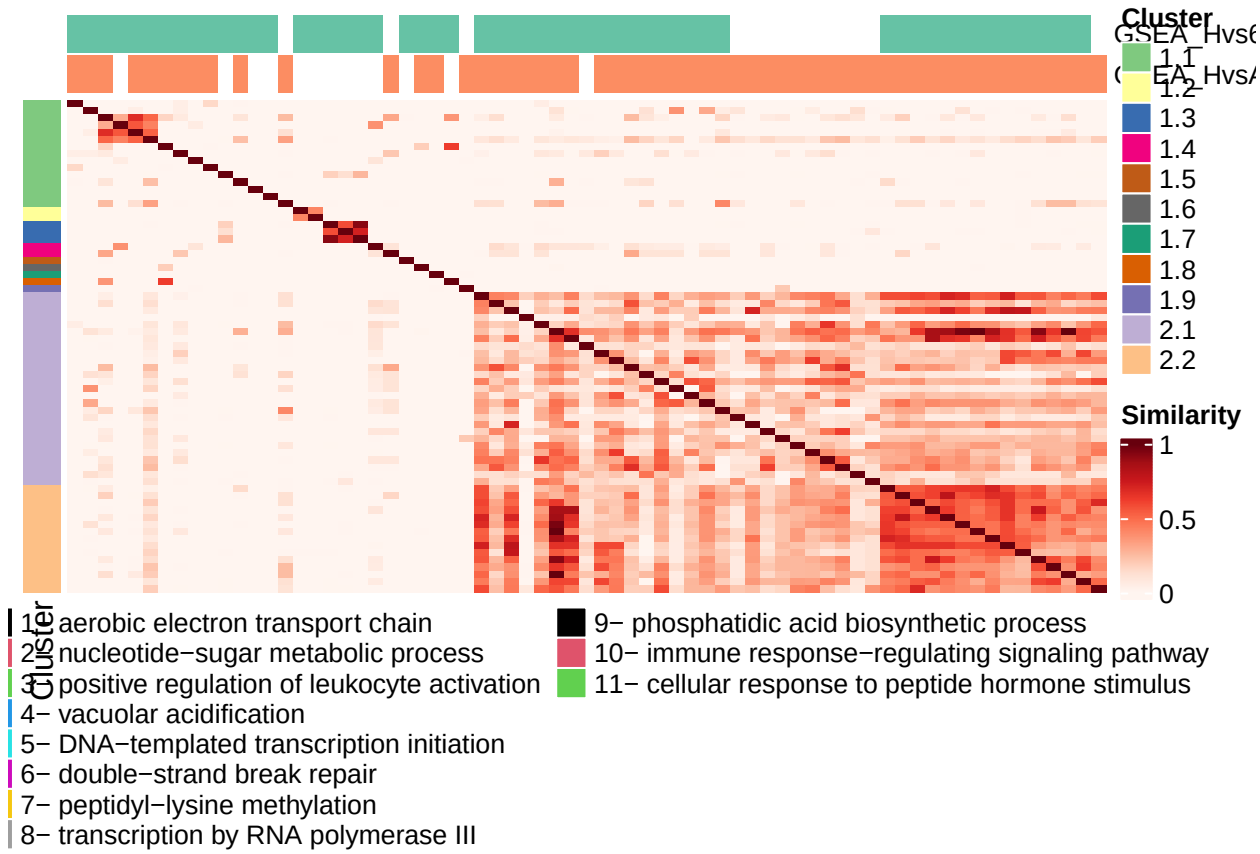

```
# Now break up the cluster 1
plotoptimalcluster1 <- OptimalGeneSets(Object = GSEA.Object3,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                cluster = 1, # which cluster
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



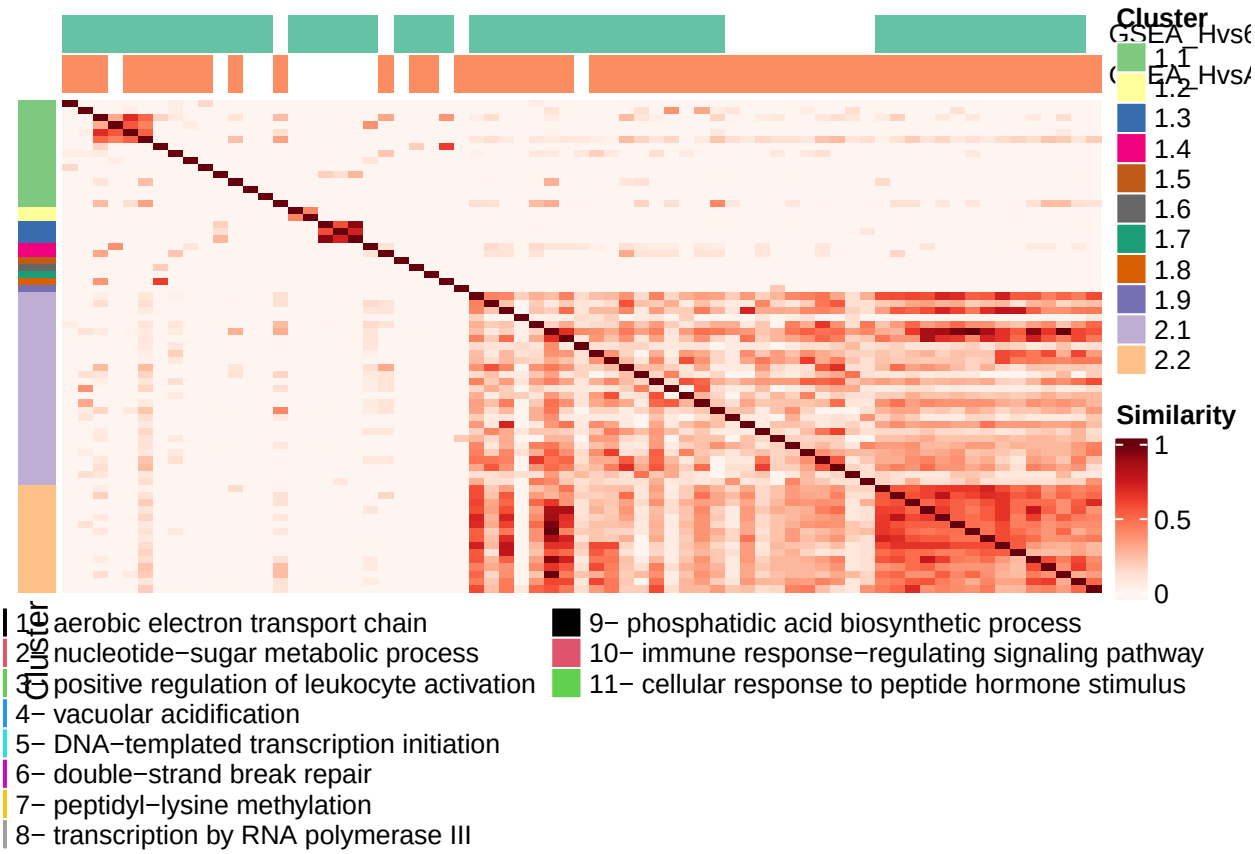

```
plotoptimalcluster1 # optimal 1 break up cluster 1 in 9 clusters
```



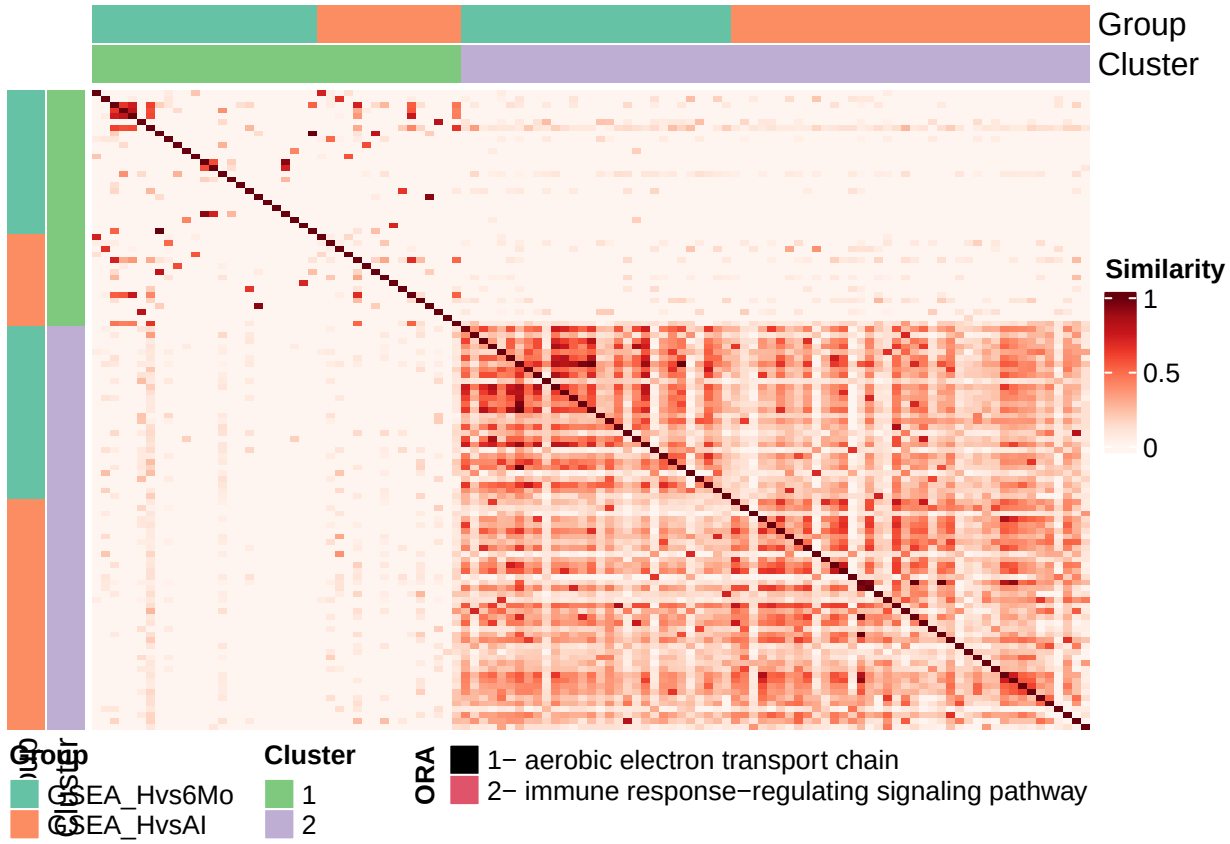

```
GSEA.Object3breakup2 <- BreakUpCluster(GSEA.Object3breakup,
                                       breakup.cluster = 1, # which cluster
                                       sub.cluster = 9, # in how many cluster split up
                                       uniquePathways = TRUE) # conside unique pathways

plotuniquebreakup2 <- PlotGeneSets(GSEA.Object3breakup2,
                                   uniquePathways = TRUE,
                                   wordcloud = FALSE, # wordcloud only supported for GO terms
                                   doORA = T) # do ora per cluster
```



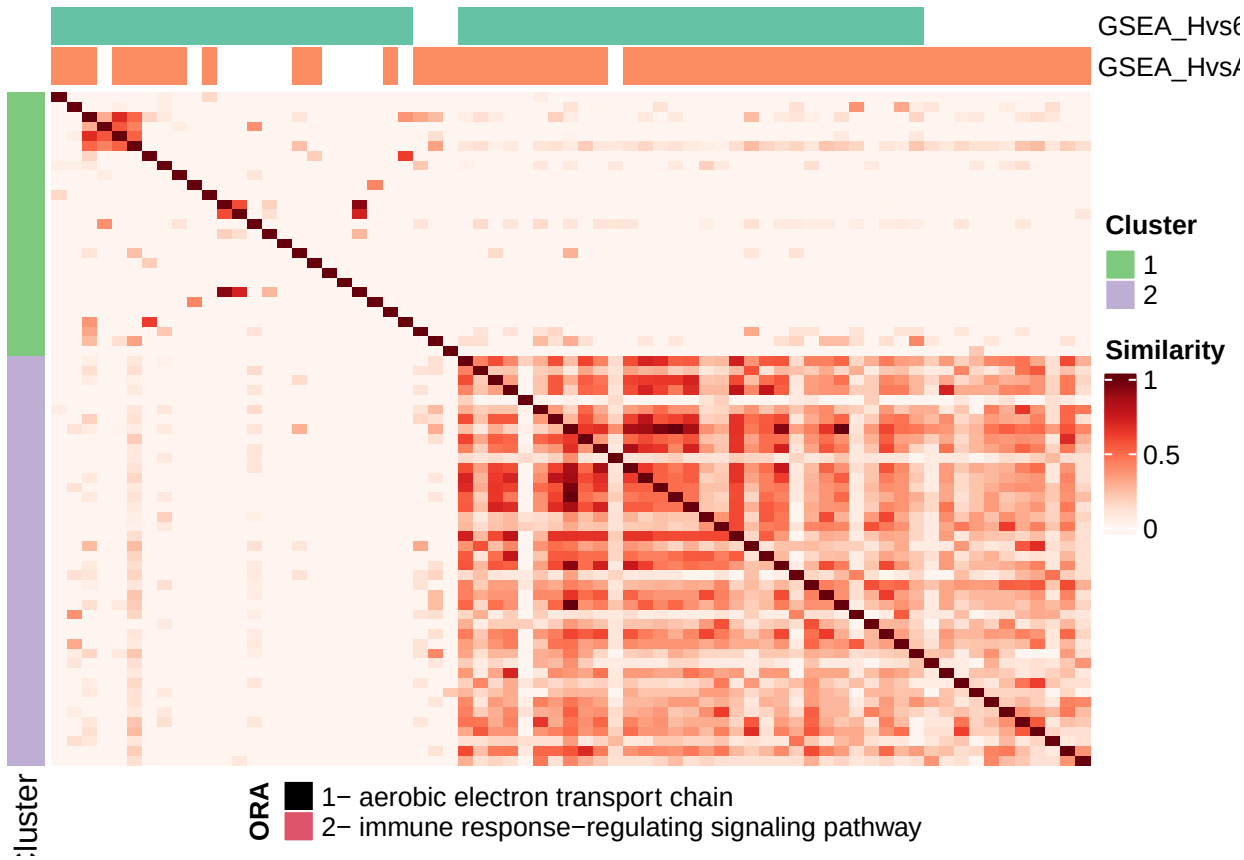

```
plotuniquebreakup2
```



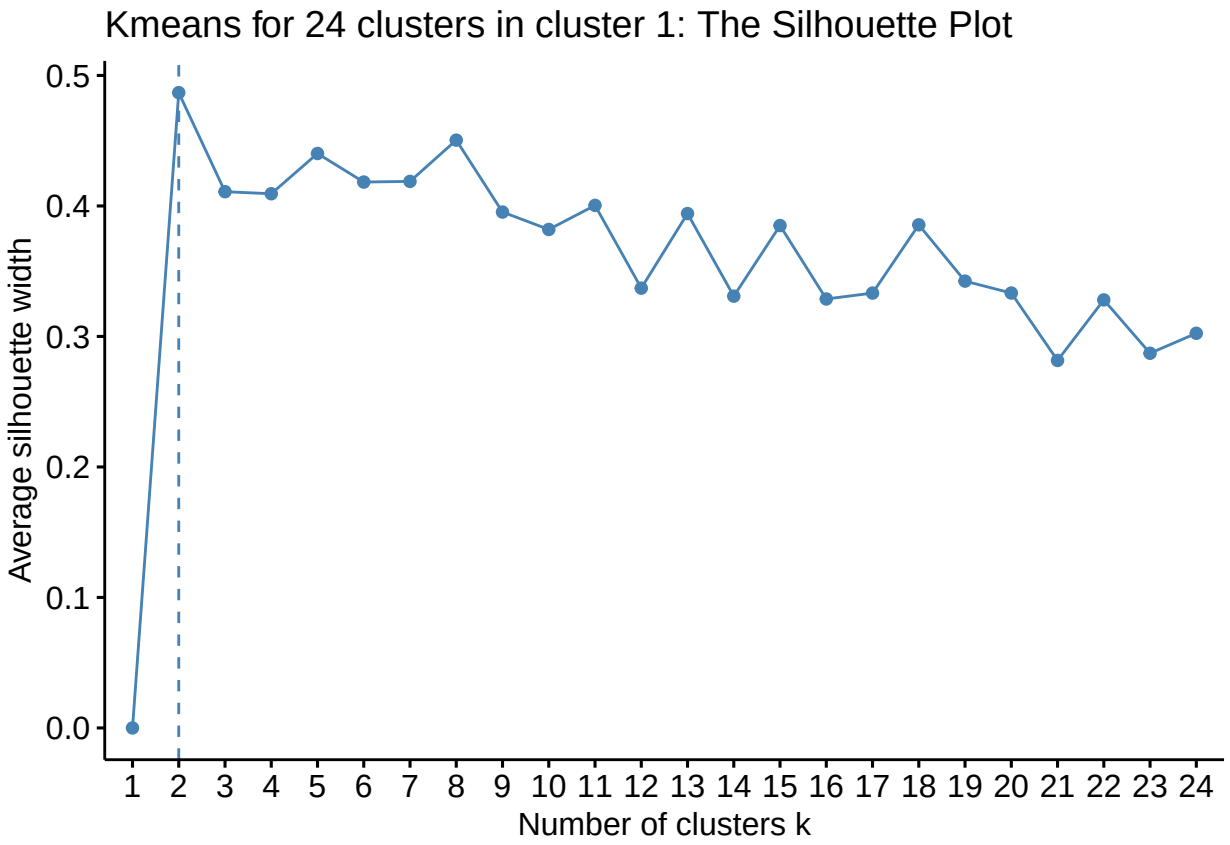

```
# plot results for both all pathways and unique pathways
plotnounique <- PlotGeneSets(GSEA.Object3,
                             uniquePathways = FALSE,
                             wordcloud = FALSE, # wordcloud only supported for GO terms
                             doORA = T) # do ora per cluster
```



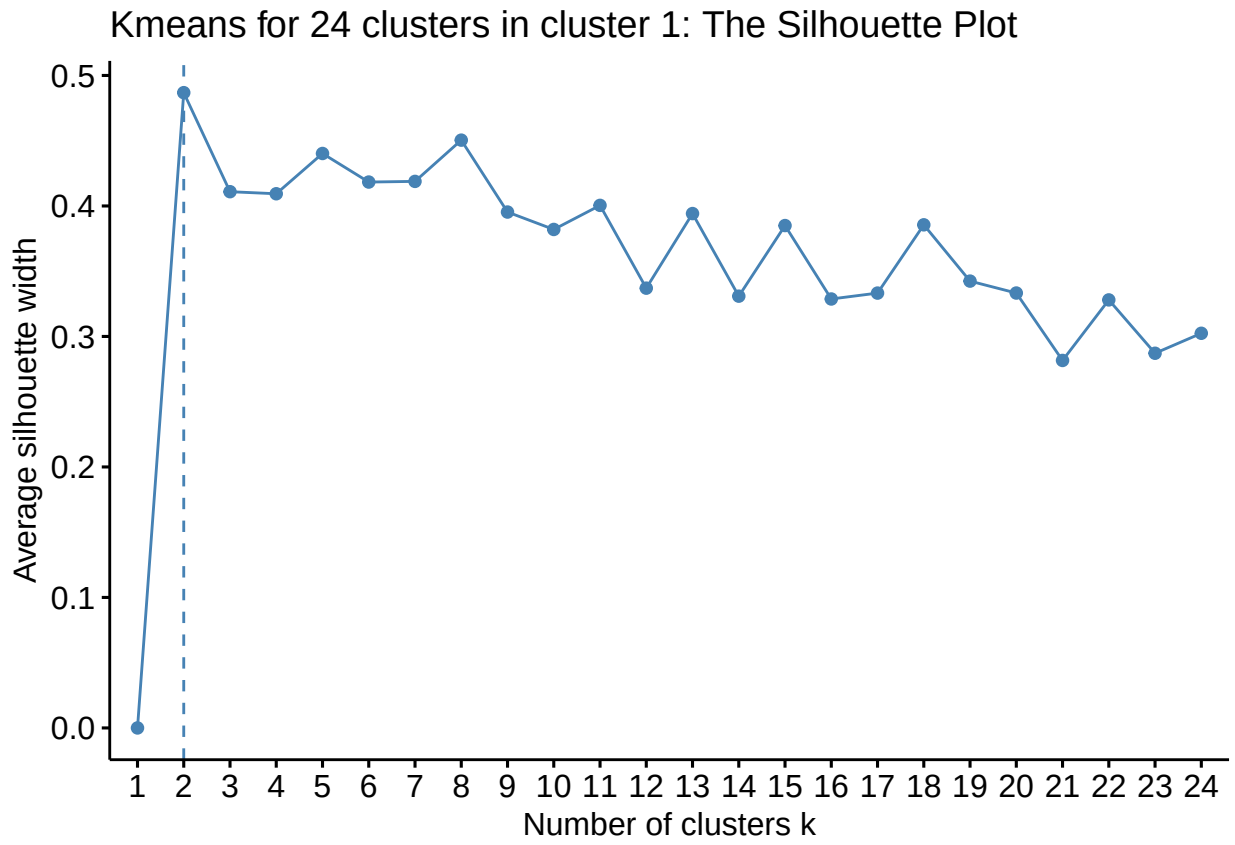

```
plotunique <- PlotGeneSets(GSEA.Object3,
                           uniquePathways = TRUE,
                           wordcloud = FALSE, # wordcloud only supported for GO terms
                           doORA = T) # do ora per cluster
```



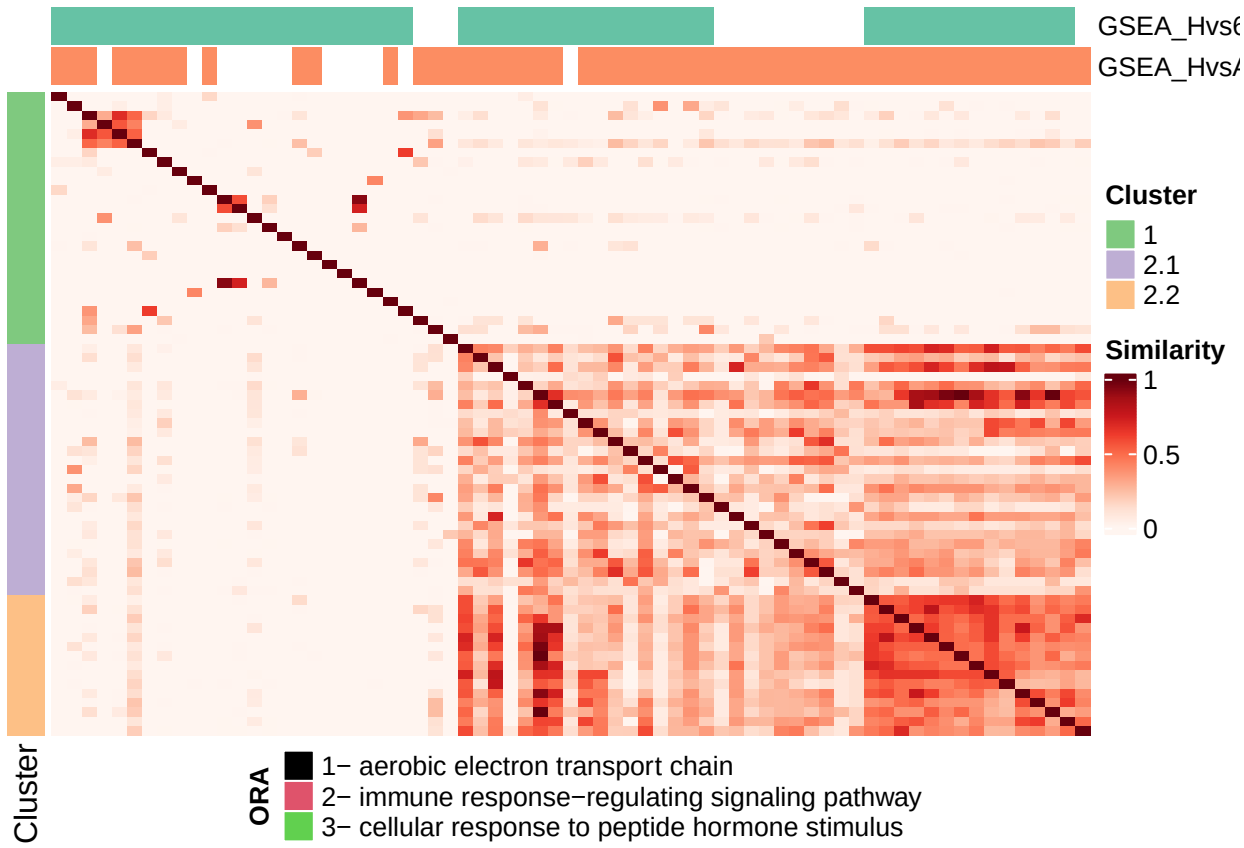

```
# let's say we are interested in exploring cluster 2 in plotunique. Lets break up this cluste for furth

plotoptimalcluster2 <- OptimalGeneSets(Object = GSEA.Object3,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                cluster = 2, # which cluster
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



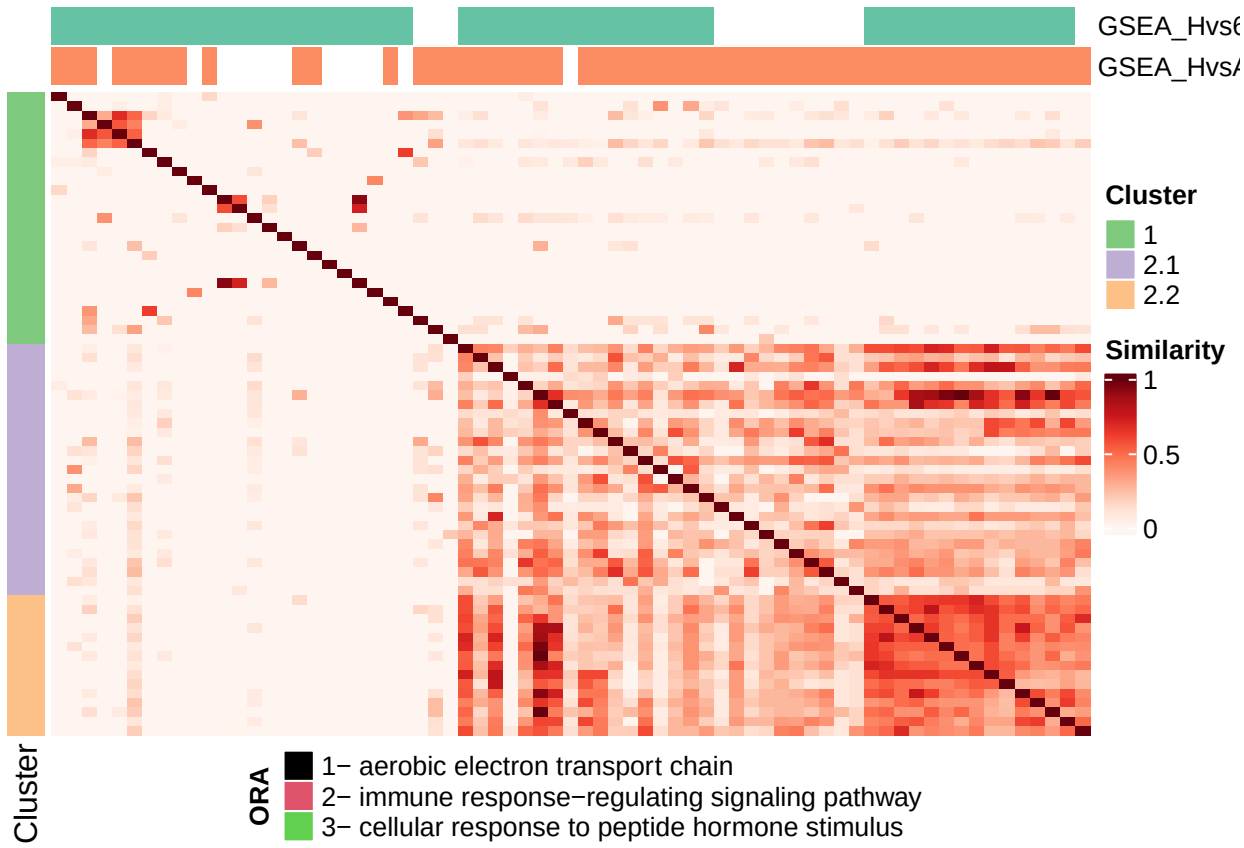

```
plotoptimalcluster2 # optimal 2 break up cluster 2 in 2 clusters
```



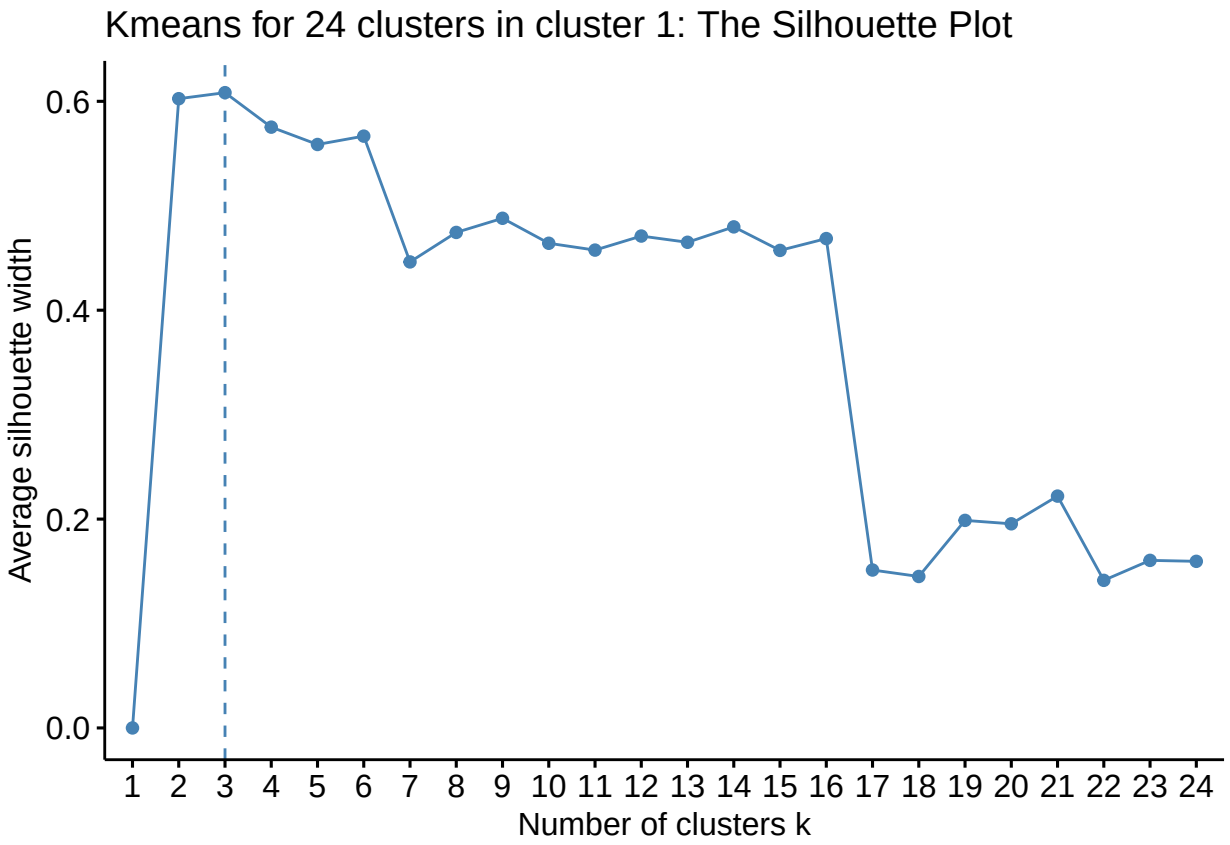

```
GSEA.Object3breakup <- BreakUpCluster(GSEA.Object3,
                                      breakup.cluster = 2, # which cluster
                                      sub.cluster = 2, # in how many cluster split up
                                      uniquePathways = TRUE) # conside unique pathways

plotuniquebreakup <- PlotGeneSets(GSEA.Object3breakup,
                                  uniquePathways = TRUE,
                                  wordcloud = FALSE, # wordcloud only supported for GO terms
                                  doORA = T) # do ora per cluster
```



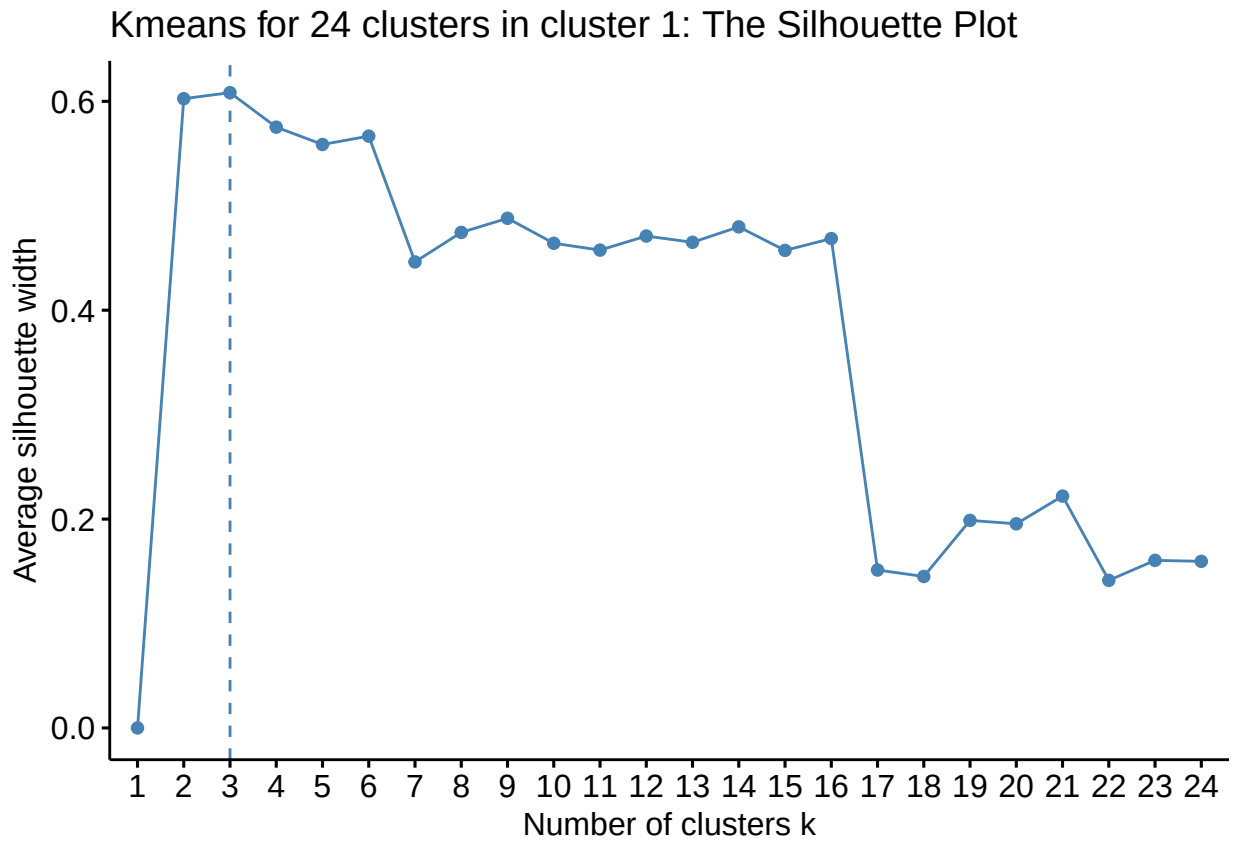

```
plotuniquebreakup
```



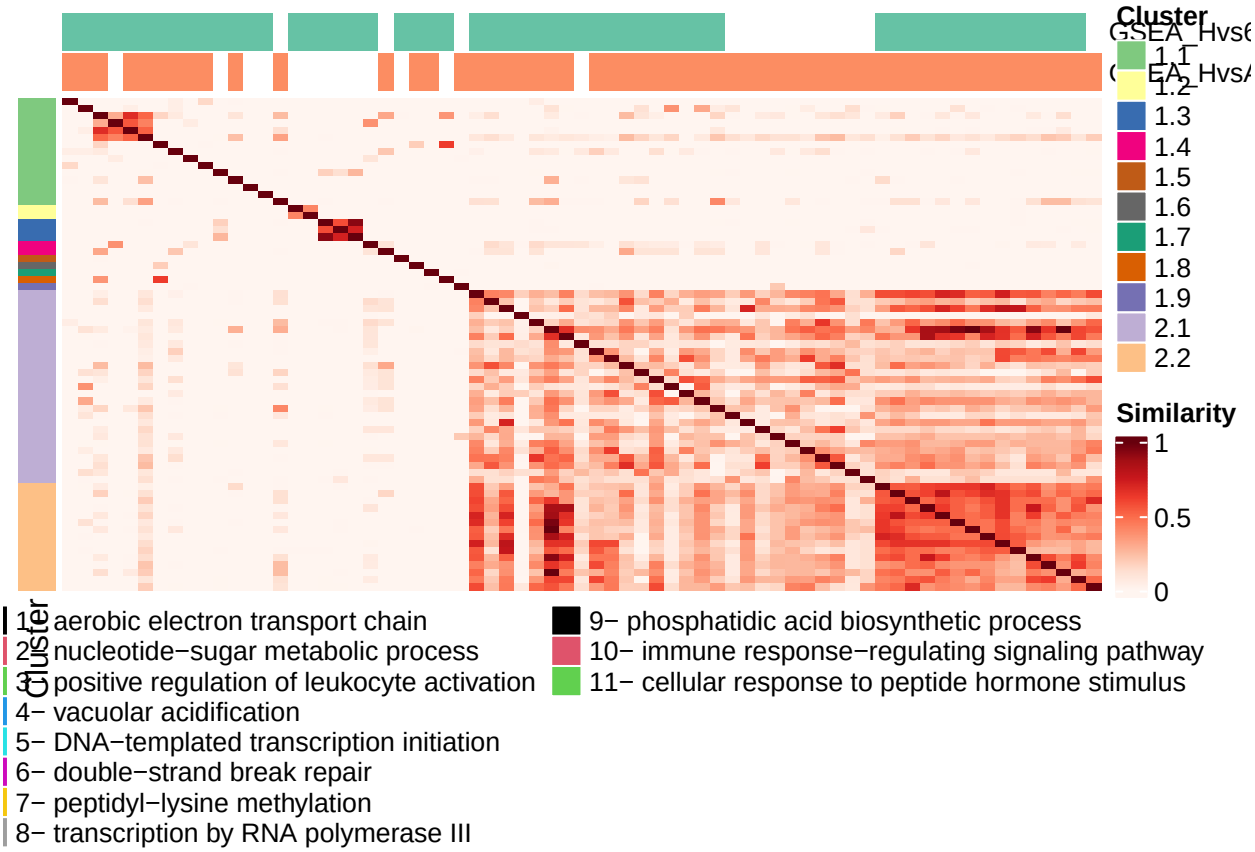

```
# Now break up the cluster 1
plotoptimalcluster1 <- OptimalGeneSets(Object = GSEA.Object3,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                cluster = 1, # which cluster
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



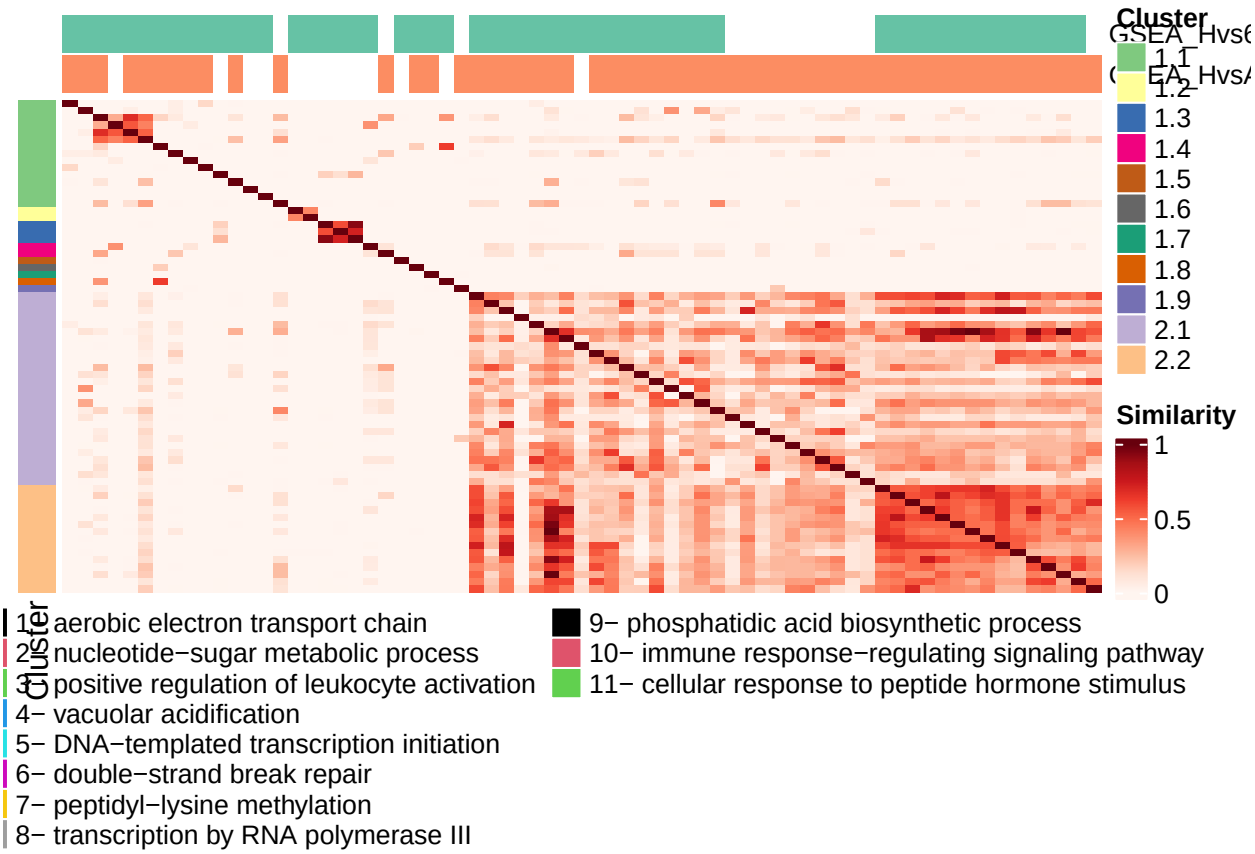

```
plotoptimalcluster1 # optimal 1 break up cluster 1 in 9 clusters
```



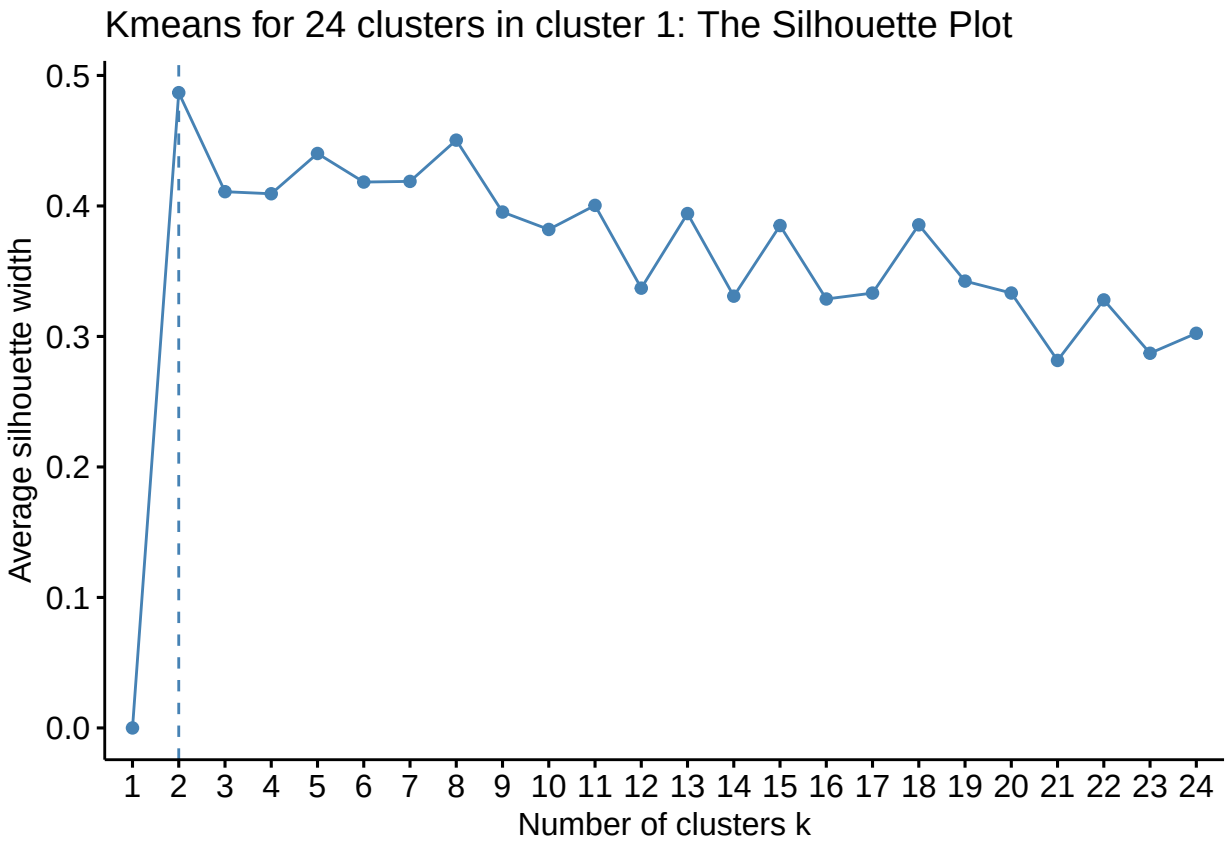

```
GSEA.Object3breakup2 <- BreakUpCluster(GSEA.Object3breakup,
                                       breakup.cluster = 1, # which cluster
                                       sub.cluster = 9, # in how many cluster split up
                                       uniquePathways = TRUE) # conside unique pathways

plotuniquebreakup2 <- PlotGeneSets(GSEA.Object3breakup2,
                                   uniquePathways = TRUE,
                                   wordcloud = FALSE, # wordcloud only supported for GO terms
                                   doORA = T) # do ora per cluster
```



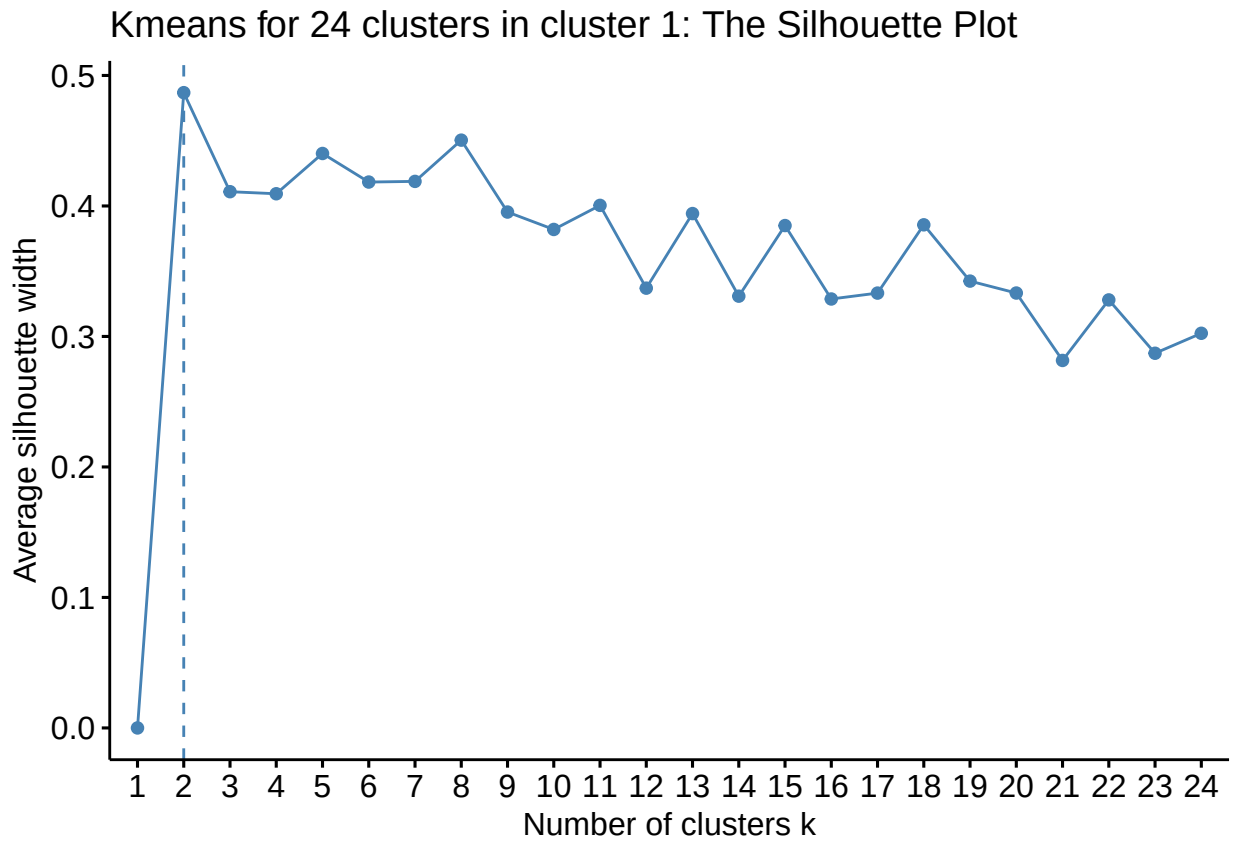

```
plotuniquebreakup2
```



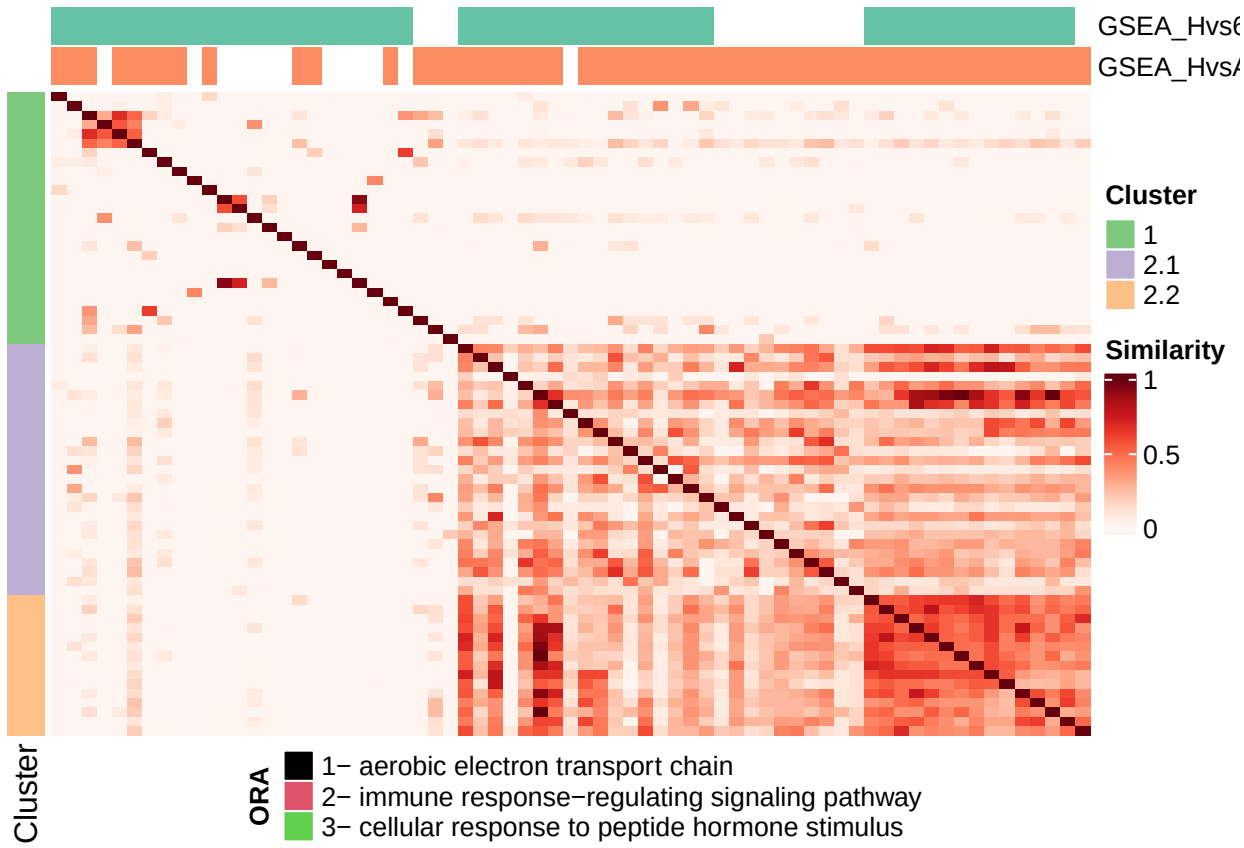

```
# let's say we are interested in exploring cluster 2 in plotunique. Lets break up this cluste for furth

plotoptimalcluster2 <- OptimalGeneSets(Object = GSEA.Object3,
                uniquePathway = TRUE, # consider all the pathways (also repeated) or the unique pathway
                cluster = 2, # which cluster
                method = "silhouette", max_cluster= 24, cluster_method = "kmeans", main= "Kmeans for 24
```



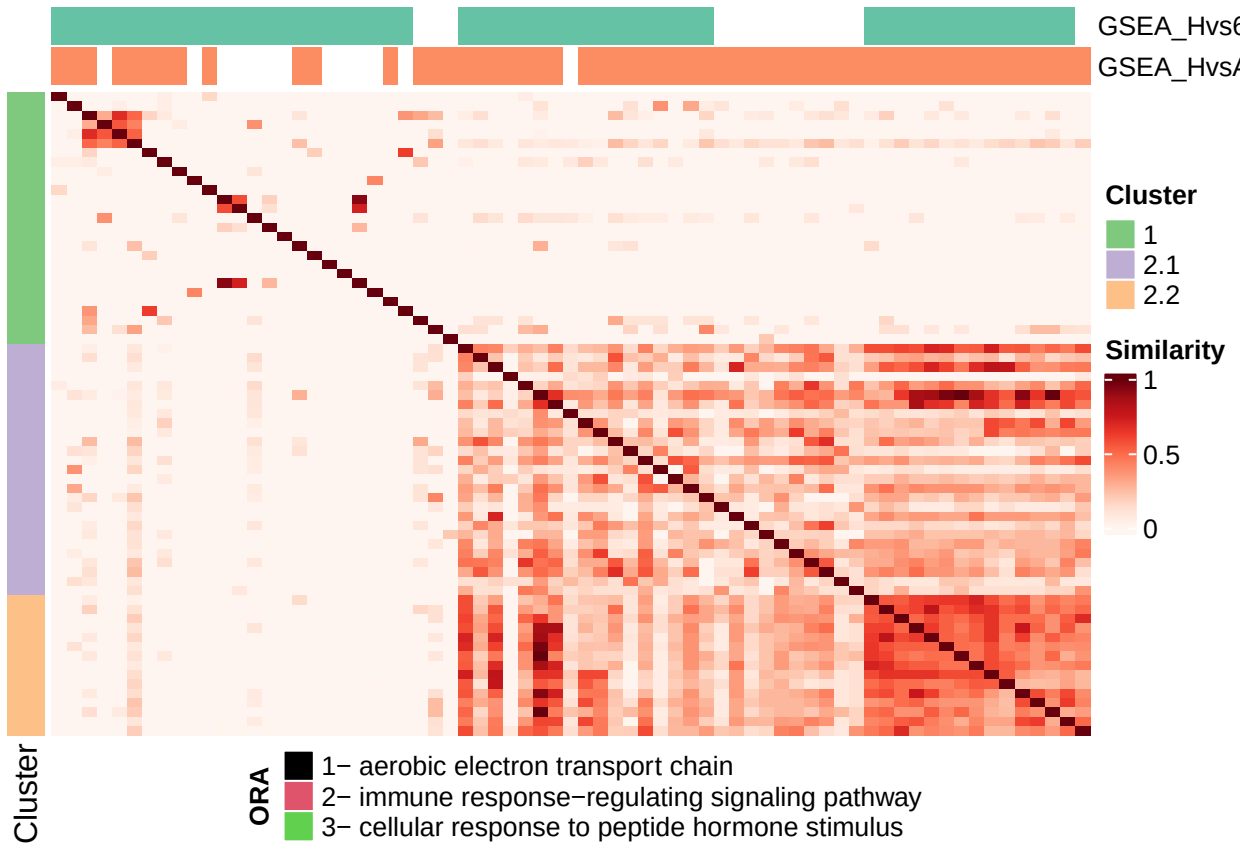

```
plotoptimalcluster2 # optimal 2 break up cluster 2 in 2 clusters
```




```
GSEA.Object3breakup <- BreakUpCluster(GSEA.Object3,
                                      breakup.cluster = 2, # which cluster
                                      sub.cluster = 2, # in how many cluster split up
                                      uniquePathways = TRUE) # conside unique pathways

plotuniquebreakup <- PlotGeneSets(GSEA.Object3breakup,
                                  uniquePathways = TRUE,
                                  wordcloud = FALSE, # wordcloud only supported for GO terms
                                  doORA = T) # do ora per cluster
```




```
plotuniquebreakup
```


GSEA_Hvs6
GSEA_HvsA
Cluster
1
2.1
2.2
Similarity
1
0.5
0
Cluster
ORA
1− aerobic electron transport chain
2− immune response−regulating signaling pathway
3− cellular response to peptide hormone stimulus