Differential expression analysis by subtype

2024-11-27

```
library(ComplexHeatmap)
```

```
##      grid
```

```
## ========================================
## ComplexHeatmap version 2.20.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
library(DESeq2)
```

```
##      S4Vectors
```

```
##      stats4
```

```
##      BiocGenerics
```

```
##
##    'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
```

```
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min

##
##    'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

##      IRanges

##
##    'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

##      GenomicRanges

##      GenomeInfoDb

##      SummarizedExperiment

##      MatrixGenerics

##      matrixStats

##
##    'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
```

```
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

##      Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
##    'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

```
library(pheatmap)
```

```
## Warning:   'pheatmap' R 4.4.2

##
##    'pheatmap'

## The following object is masked from 'package:ComplexHeatmap':
##
##     pheatmap
```

```
library(ggplot2)
```

```
## Warning:   'ggplot2' R 4.4.2
```

```
library(clusterProfiler)
```

```
##

## clusterProfiler v4.12.6 Learn more at https://yulab-smu.top/contribution-knowledge-mining/
##
## Please cite:
##
## T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan,
## X Fu, S Liu, X Bo, and G Yu. clusterProfiler 4.0: A universal
## enrichment tool for interpreting omics data. The Innovation. 2021,
## 2(3):100141
```

```
##
##     'clusterProfiler'

## The following object is masked from 'package:IRanges':
##
##     slice

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

## The following object is masked from 'package:stats':
##
##     filter
```

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

```
##      AnnotationDbi

##
##     'AnnotationDbi'

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

##
```

```
library(AnnotationDbi)
library(dplyr)
```

```
##
##     'dplyr'

## The following object is masked from 'package:AnnotationDbi':
##
##     select

## The following object is masked from 'package:Biobase':
##
##     combine

## The following object is masked from 'package:matrixStats':
##
##     count

## The following objects are masked from 'package:GenomicRanges':
##
##     intersect, setdiff, union
```

```
## The following object is masked from 'package:GenomeInfoDb':
##
##     intersect

## The following objects are masked from 'package:IRanges':
##
##     collapse, desc, intersect, setdiff, slice, union

## The following objects are masked from 'package:S4Vectors':
##
##     first, intersect, rename, setdiff, setequal, union

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
##
##     'tidyr'

## The following object is masked from 'package:S4Vectors':
##
##     expand
```

```
library(survival)
library(survminer)
```

```
## Warning:   'survminer' R 4.4.2

##      ggpubr

##
##     'survminer'

## The following object is masked from 'package:survival':
##
##     myeloma
```

```
library(readr)
library(KEGGREST)
```

```r
setwd("C:\\Users\\COLORFUL\\Documents\\bmeg310-1")

raw.clinical.patients <- read.table("data_clinical_patient.txt", sep = "\t",
                                    header = TRUE)
raw.data.mutations <- read.table("data_mutations.txt", sep = "\t",
                                 header = TRUE)
raw.data.RNAseq <- read.csv("RNAseq_BRCA.csv", header = TRUE, stringsAsFactors = FALSE)

#Preprocess of gene name
raw.data.RNAseq$X <- gsub("\\..*", "", raw.data.RNAseq$X)

raw.data.RNAseq$X <- make.unique(raw.data.RNAseq$X)

rownames(raw.data.RNAseq) <- raw.data.RNAseq[,1]
raw.data.RNAseq <- raw.data.RNAseq[,-1]
raw.data.RNAseq <- as.matrix(raw.data.RNAseq)
storage.mode(raw.data.RNAseq) <- "numeric"

#Filter data where you only have 0 or 1 read count across all samples.

raw.data.RNAseq <- raw.data.RNAseq[rowSums(raw.data.RNAseq)>1,]
```

```r
simplified_names <- sapply(colnames(raw.data.RNAseq), function(name) {
  segments <- strsplit(name, "\\.")[[1]]
  paste(segments[1:3], collapse = "-")
})

colnames(raw.data.RNAseq) <- make.unique(sapply(colnames(raw.data.RNAseq), function(name) {
  segments <- strsplit(name, "\\.")[[1]]
  paste(segments[1:3], collapse = "-")
}))
unique.RNA <- raw.data.RNAseq
```

```r
#Unique Patients in each data set
unique.clinical <- as.data.frame(unique(raw.clinical.patients$PATIENT_ID))
unique.mutations <- as.data.frame(unique
                                  (raw.data.mutations$Tumor_Sample_Barcode))

#Addition patient ID's to Mutation data
mutation.patients <- as.data.frame(raw.data.mutations$Tumor_Sample_Barcode)
colnames(mutation.patients) <- "Patient_ID"
mutation.patients$Patient_ID <- substr(mutation.patients$Patient_ID, 1, 12)
raw.data.mutations <- cbind(mutation.patients, raw.data.mutations)

colnames(unique.clinical) <- "Patient_ID"
colnames(unique.mutations) <- "Patient_ID"
unique.RNA <- as.data.frame(unique.RNA)
unique.mutations$Patient_ID <- substr(unique.mutations$Patient_ID, 1, 12)

#Finding common patients
common_patient_ids <- Reduce(intersect, list(
  unique.clinical$Patient_ID,
  unique.mutations$Patient_ID,
```

```
  colnames(unique.RNA)
))

#3 data sets with all 975 common patients
clinical.patients <- raw.clinical.patients[raw.clinical.patients$PATIENT_ID
                                           %in% common_patient_ids, ]
data.mutation <- raw.data.mutations[raw.data.mutations$Patient_ID
                                    %in% common_patient_ids, ]
data.RNAseq.filtered <- raw.data.RNAseq[,colnames(raw.data.RNAseq)
                              %in% common_patient_ids]
```

```
#Sort  v and countdata
clinical.patients <- clinical.patients[order(clinical.patients$PATIENT_ID), ]
filtered.clinical <- clinical.patients
data.RNAseq.filtered <- as.data.frame(data.RNAseq.filtered)
data.RNAseq.filtered <- data.RNAseq.filtered[,order(colnames(data.RNAseq.filtered))]

#Condition for DESeq2 coldata
colData <- cbind(clinical.patients$PATIENT_ID,clinical.patients$SUBTYPE)
rownames(colData) <- colData[,1]
colData <- colData[,-1]
colData <- as.data.frame(colData)
#Write BRCA to empty condition name
colnames(colData) <- "condition"
colData$condition <- ifelse(
  is.na(colData$condition) | colData$condition == "",
  "BRCA",
  colData$condition
)
colData$condition <- as.factor(colData$condition)
```

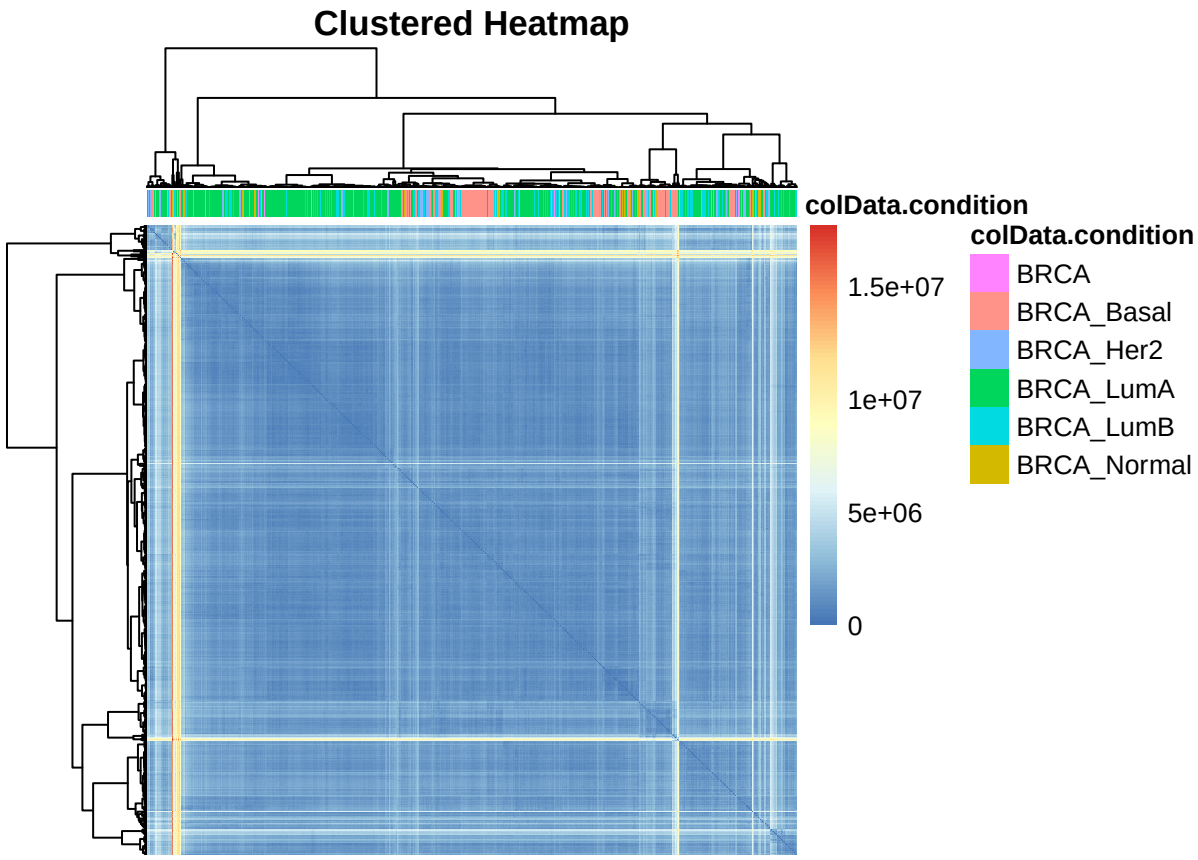
```
#Run DESeq2
dds = DESeqDataSetFromMatrix(countData=data.RNAseq.filtered,
                             colData=colData,
                             design=~ condition)
```

```
## converting counts to integer mode
```

```
dds = estimateSizeFactors(dds)
```

```
#Normalize the data by using DEseq2
data.RNAseq.normalized.DESeq <- counts(dds, normalized = TRUE)

# Calculate variances
gene_variances <- apply(data.RNAseq.normalized.DESeq, 1, var)

#Pick the top 1000 most variable genes
top_genes <- order(gene_variances, decreasing = TRUE)[1:1000]

data.RNAseq.normalized.DESeq.top1000 <- data.RNAseq.normalized.DESeq[top_genes, ]
```

```r
#Use pheat map to find cluster
RNAseq.distance <- dist(t(data.RNAseq.normalized.DESeq.top1000))
annot_col = data.frame(colData$condition)
row.names(annot_col) <- rownames(colData)
RNAseq.distance.Matrix = as.matrix(RNAseq.distance)
rownames(RNAseq.distance.Matrix) = colnames(data.RNAseq.filtered)
colnames(RNAseq.distance.Matrix) = colnames(data.RNAseq.filtered)

pheatmap(
  RNAseq.distance.Matrix,
  annotation_col = annot_col,
  clustering_distance_rows = RNAseq.distance,
  clustering_distance_cols = RNAseq.distance,
  clustering_method = "ward.D",
  main = "Clustered Heatmap",
  show_rownames=FALSE,
  show_colnames=FALSE
)
```




```r
RNAseq.distance <- dist(t(data.RNAseq.normalized.DESeq.top1000),upper = TRUE)

#Use ward.D2 to divide the result into 3 groups
hclust_result <- hclust(RNAseq.distance, method = "ward.D")
cluster_assignments <- cutree(hclust_result, k = 3)
colData$Cluster <- as.factor(cluster_assignments)
```

```r
clinical.patients$Cluster <- as.factor(cluster_assignments[match(clinical.patients$PATIENT_ID, rownames
```

```r
#Show number of patients in each cluster
cluster_counts <- table(colData$Cluster)
barplot(cluster_counts,
        main = "Number of Patients in Each Cluster",
        xlab = "Cluster",
        ylab = "Number of Patients")
```


```r
#Sort coldata and countdata
filtered.clinical <- filtered.clinical[order(filtered.clinical$PATIENT_ID), ]
data.RNAseq.filtered <- data.RNAseq.filtered[,order(colnames(data.RNAseq.filtered))]
```

```r
#DE analysis on all genes
dds_by_Cluster <- DESeqDataSetFromMatrix(
  countData = data.RNAseq.filtered,
  colData = colData,
  design = ~ Cluster
)
```

```
## converting counts to integer mode
```

```r
dds_by_Cluster <- DESeq(dds_by_Cluster)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 9514 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```r
res_Cluster12 <- results(dds_by_Cluster, contrast = c("Cluster", "1", "2"))  #Compare Cluster 1 and Clu
res_Cluster23 <- results(dds_by_Cluster, contrast = c("Cluster", "2", "3"))  #Compare Cluster 1 and Clu
res_Cluster13 <- results(dds_by_Cluster, contrast = c("Cluster", "1", "3"))  #Compare Cluster 1 and Clu
sig_genes12 <- subset(res_Cluster12, padj < 0.05)
sig_genes23 <- subset(res_Cluster23, padj < 0.05)
sig_genes13 <- subset(res_Cluster13, padj < 0.05)

head(sig_genes12)
```

```
## log2 fold change (MLE): Cluster 1 vs 2
## Wald test p-value: Cluster 1 vs 2
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000419  2374.367       0.291500 0.0545703   5.34174 9.20584e-08
## ENSG00000000460   723.165       0.364167 0.0766411   4.75159 2.01825e-06
## ENSG00000000938   585.958      -0.403576 0.0976666  -4.13218 3.59343e-05
## ENSG00000000971  3306.434      -0.894386 0.0975834  -9.16535 4.93864e-20
## ENSG00000001036  2708.086       0.169741 0.0649514   2.61336 8.96576e-03
## ENSG00000001167  3213.085       0.273414 0.0639921   4.27261 1.93195e-05
##                        padj
##                   <numeric>
## ENSG00000000419 1.00104e-06
## ENSG00000000460 1.57027e-05
## ENSG00000000938 1.98735e-04
## ENSG00000000971 4.06145e-18
## ENSG00000001036 2.30919e-02
## ENSG00000001167 1.15590e-04
```

```
head(sig_genes23)
```

```
## log2 fold change (MLE): Cluster 2 vs 3
## Wald test p-value: Cluster 2 vs 3
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003  3137.890      -0.655725  0.161102  -4.07023 4.69662e-05
## ENSG00000000005   129.186      -3.624678  0.431229  -8.40545 4.26206e-17
## ENSG00000001036  2708.086      -0.543431  0.114249  -4.75654 1.96937e-06
## ENSG00000001084  2250.665      -0.500214  0.114608  -4.36455 1.27383e-05
## ENSG00000001497  3485.593      -0.192487  0.082076  -2.34523 1.90156e-02
## ENSG00000001561  1841.468      -0.522658  0.176715  -2.95763 3.10013e-03
##                        padj
##                   <numeric>
## ENSG00000000003 1.52898e-04
## ENSG00000000005 6.87700e-16
## ENSG00000001036 8.08338e-06
## ENSG00000001084 4.58798e-05
## ENSG00000001497 3.58032e-02
## ENSG00000001561 7.03702e-03
```

```
head(sig_genes13)
```

```
## log2 fold change (MLE): Cluster 1 vs 3
## Wald test p-value: Cluster 1 vs 3
## DataFrame with 6 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003  3137.890      -0.529363 0.1411490  -3.75039 1.76561e-04
## ENSG00000000005   129.186      -3.735039 0.3776870  -9.88925 4.63508e-23
## ENSG00000000457  1582.739       0.215604 0.0842905   2.55787 1.05314e-02
## ENSG00000000971  3306.434      -0.959158 0.1503982  -6.37746 1.80053e-10
## ENSG00000001036  2708.086      -0.373690 0.1000986  -3.73322 1.89051e-04
## ENSG00000001084  2250.665      -0.362402 0.1004139  -3.60908 3.07282e-04
##                        padj
##                   <numeric>
## ENSG00000000003 5.52891e-04
## ENSG00000000005 1.03941e-21
## ENSG00000000457 2.25613e-02
## ENSG00000000971 1.31221e-09
## ENSG00000001036 5.88837e-04
## ENSG00000001084 9.21728e-04
```

```
#pathway analysis
gene_list12 <- rownames(sig_genes12)
mapped_genes12 <- mapIds(
  org.Hs.eg.db,
  keys = gene_list12,
  column = "ENTREZID",
  keytype = "ENSEMBL",
  multiVals = "first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
mapped_genes12 <- na.omit(mapped_genes12)
head(mapped_genes12)
```

```
## ENSG00000000419 ENSG00000000460 ENSG00000000938 ENSG00000000971 ENSG00000001036
##          "8813"         "55732"          "2268"          "3075"          "2519"
## ENSG00000001167
##          "4800"
```

```
#pathway analysis
gene_list23 <- rownames(sig_genes23)
mapped_genes23 <- mapIds(
  org.Hs.eg.db,
  keys = gene_list23,
  column = "ENTREZID",
  keytype = "ENSEMBL",
  multiVals = "first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
mapped_genes23 <- na.omit(mapped_genes23)
head(mapped_genes23)
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000001036 ENSG00000001084 ENSG00000001497
##          "7105"         "64102"          "2519"          "2729"         "81887"
## ENSG00000001561
##         "22875"
```

```
#pathway analysis
gene_list13 <- rownames(sig_genes13)
mapped_genes13 <- mapIds(
  org.Hs.eg.db,
  keys = gene_list13,
  column = "ENTREZID",
  keytype = "ENSEMBL",
  multiVals = "first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
mapped_genes13 <- na.omit(mapped_genes13)
head(mapped_genes13)
```

```
## ENSG00000000003 ENSG00000000005 ENSG00000000457 ENSG00000000971 ENSG00000001036
##          "7105"         "64102"         "57147"          "3075"          "2519"
## ENSG00000001084
##          "2729"
```