R Notebook

Code

load the DESeq2 and ggplot2 libraries

This code allows you to access the functions provided by these libraries, enabling you to perform differential gene expression analysis using DESeq2 and create visually appealing plots using ggplot2.

Hide

```
library(DESeq2)
```

```
Loading required package: S4Vectors
Loading required package: stats4
Loading required package: BiocGenerics

Attaching package: '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, sort, table, tapply, union, unique, unsplit,
    which.max, which.min


Attaching package: 'S4Vectors'

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

    findMatches

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges

Attaching package: 'IRanges'

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

    windows

Loading required package: GenomicRanges
Loading required package: GenomeInfoDb
Loading required package: SummarizedExperiment
Loading required package: MatrixGenerics
Loading required package: matrixStats

Attaching package: '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

Loading required package: Biobase
Welcome to Bioconductor

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


Attaching package: ‘Biobase’

The following object is masked from ‘package:MatrixGenerics’:

    rowMedians

The following objects are masked from ‘package:matrixStats’:

    anyMissing, rowMedians
```

Hide

```
library(ggplot2)
```

Set the working directory in R to the specified path

Hide

```
workingdirectory <- setwd("C:\\Users\\HP\\Downloads\\GSE")
```

```
Warning: The working directory was changed to C:/Users/HP/Downloads/GSE inside a note
book chunk. The working directory will be reset when the chunk is finished running. U
se the knitr root.dir option in the setup chunk to change the working directory for n
otebook chunks.
```

Assign the file path. Naming the file path to the countName variable enables us to

Hide

```
countName <- "C:\\Users\\HP\\Downloads\\GSE\\newcancer.txt"
```

Read the text file and assign the data to a variable named countData. The read.delim() function reads data from a delimited text file into a data frame.

We specific the delimiter as the tab character (using the sep argument.

The header = TRUE argument indicates that the first row of the file contains the column names or headers.

Hide

```
countData <- read.delim(countName, header = TRUE, sep = "\t")
```

Assign new column names to the countData data frame. The names() function is used to access or modify the column names of a data frame.

This is for the sake of the metadata, to ensure the names in the column of countdata is represented in the metadata.

Hide

```
names(countData) <- c("gene_id", "SRR13147304", "SRR13147305", "SRR13147306", "SRR131
47307", "SRR13147308", "SRR13147309", "SRR13147310", "SRR13147312")
```

displays the first few rows of the countData data frame.

Hide

```
head(countData)
```

gene_id <chr>	SRR13147… <dbl>	SRR13147… <dbl>	SRR13147… <dbl>	SRR13147… <dbl>	SRR13147… <dbl>	SR
1 ENST00000434970.2	0	0	0	0	0	
2 ENST00000448914.1	0	0	0	0	0	
3 ENST00000415118.1	0	0	0	0	0	
4 ENST00000632684.1	0	0	0	0	0	
5 ENST00000631435.1	0	0	0	0	0	
6 ENST00000430425.1	0	0	0	0	0	

6 rows | 1-8 of 9 columns

Repeat the steps for the Metadata

Hide

```
metaData <- "C:\\Users\\HP\\Downloads\\GSE\\metaDataNewcancer.txt"
```

Hide

```
metaData <- read.delim( metaData, header = TRUE, sep = "\t")
```

Hide

```
metaData
```

samples <chr>	status <chr>
SRR13147304	Resistant
SRR13147305	Resistant
SRR13147306	Resistant
SRR13147307	Resistant
SRR13147308	sensitive

samples <chr>	status <chr>
SRR13147309	sensitive
SRR13147310	sensitive
SRR13147312	sensitive

8 rows

Data cleaning

The next line of code removes duplicate rows from the countData dataframe based on the gene_id column. It keeps only the first occurrence of each unique gene_id.

Hide

```
countData = countData[!duplicated(countData$gene_id),]
```

Replace all zero values in the countData dataframe with NA (missing values). It is often done to indicate missing or undefined values in the dataset.

Hide

```
countData[countData==0] <- NA
```

Remove rows containing any missing values (NA) from the countData dataframe. It helps in removing incomplete or inconsistent rows from the dataset.

Hide

```
countData = na.omit(countData)
```

Display the dimensions (number of rows and columns) of the countData dataframe.

Hide

```
dim(countData)
```

```
[1] 46122     9
```

The next line creates a new variable called myRowNames that contains the values from the gene_id column of countData. This is to store the original gene IDs before modifying the dataset.

Hide

```
myRowNames <- countData$gene_id
```

Create a new dataframe called countData.fixed by selecting columns 2 to 9 from the countData dataframe. Exclude the gene_id column, which is the first column.

Hide

```
countData.fixed <- countData[,c(2:9)]
```

display the contents of the countData.fixed

Hide

```
View(countData.fixed)
```

Assign new column names to the metaData dataframe.

Hide

```
names(metaData) <- c("id", 'status')
```

The next code creates a DESeqDataSet object, dds, from the count data provided in the countData.fixed dataframe. It also includes the metadata information from the metaData dataframe.

The design formula specifies the statistical design of the experiment, where "~status" indicates that the differential expression analysis will be performed based on the "status" variable.

The tidy = FALSE argument indicates that the output should not be converted to a tidy format.

Hide

```
dds <- DESeqDataSetFromMatrix(countData=round(countData.fixed),
                              colData=metaData,
                              design=~status, tidy = FALSE)
```

```
converting counts to integer mode
Warning: some variables in design formula are characters, converting to factors
```

displays the values of the "status" to show the different groups or conditions used for differential expression analysis.

Hide

```
dds$status
```

```
[1] Resistant Resistant Resistant Resistant sensitive sensitive sensitive sensitive
Levels: Resistant sensitive
```

Run the differential expression analysis using the DESeq function on the dds object.

This estimates the size factors, dispersion, and performs statistical tests to identify differentially expressed genes between the specified groups or conditions

Hide

```
dds <- DESeq(dds)
```

```
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```

The next line code retrieves the results of the differential expression analysis from the dds object. It provides a summary of the analysis, including log2 fold changes, p-values, and adjusted p-values (if applicable).

Hide

```
res <- results(dds)
```

Display the top few rows of the results from the differential expression analysis in a tabular format. The tidy=TRUE argument ensures a tidy output with each row representing a gene and its associated statistics.

Hide

```
head(results(dds, tidy=TRUE))
```

r... <chr>	baseMean <dbl>	log2FoldChange <dbl>	lfcSE <dbl>	stat <dbl>	pvalue <dbl>	padj <dbl>
1 86	1.1994133	0.4265766	1.6670964	0.2558800	0.7980435	0.999729
2 181	0.1108438	0.5692960	3.5338115	0.1610997	0.8720149	0.999729
3 188	1.1994133	0.4265766	1.6670964	0.2558800	0.7980435	0.999729
4 235	2.2151750	-0.1525002	0.9593884	-0.1589556	0.8737038	0.999729
5 261	0.0000000	*NA*	*NA*	*NA*	*NA*	*NA*
6 280	2.4439207	0.4257980	0.9057907	0.4700843	0.6382948	0.999729

6 rows

Get the summary of the results obtained from the differential expression analysis.

Hide

```
summary(res)
```

```

out of 44463 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 2, 0.0045%
LFC < 0 (down)     : 0, 0%
outliers [1]       : 208, 0.47%
low counts [2]     : 0, 0%
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Reorder the res dataframe based on the adjusted p-values (padj column) in ascending order. This is to sort the results from the most statistically significant to the least significant.

Hide

```
res <- res[order(res$padj),]
```

Display the top few rows of the reordered res dataframe, showing the genes with the lowest adjusted p-values. It provides a quick glimpse of the most significant differentially expressed genes.

Hide

```
head(res)
```

```
log2 fold change (MLE): status sensitive vs Resistant
Wald test p-value: status sensitive vs Resistant
DataFrame with 6 rows and 6 columns
        baseMean log2FoldChange     lfcSE      stat      pvalue      padj
       <numeric>      <numeric> <numeric> <numeric>   <numeric> <numeric>
7560    18.76147        3.60380  0.748239   4.81638 1.46187e-06 0.0646951
158296  14.65621        2.92655  0.628452   4.65676 3.21223e-06 0.0710785
12466   18.79454       -4.68081  1.054631  -4.43834 9.06535e-06 0.1337290
5455    18.63082        4.07361  0.990152   4.11413 3.88648e-05 0.3584545
22973   45.81891       -2.50757  0.611600  -4.10001 4.13127e-05 0.3584545
108734   9.51544       -3.93042  0.976223  -4.02615 5.66983e-05 0.3584545
```

Plotting and Visualization

Hide

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

This line sets the plot layout to a single plot, the default arrangement of one plot per figure.

Hide

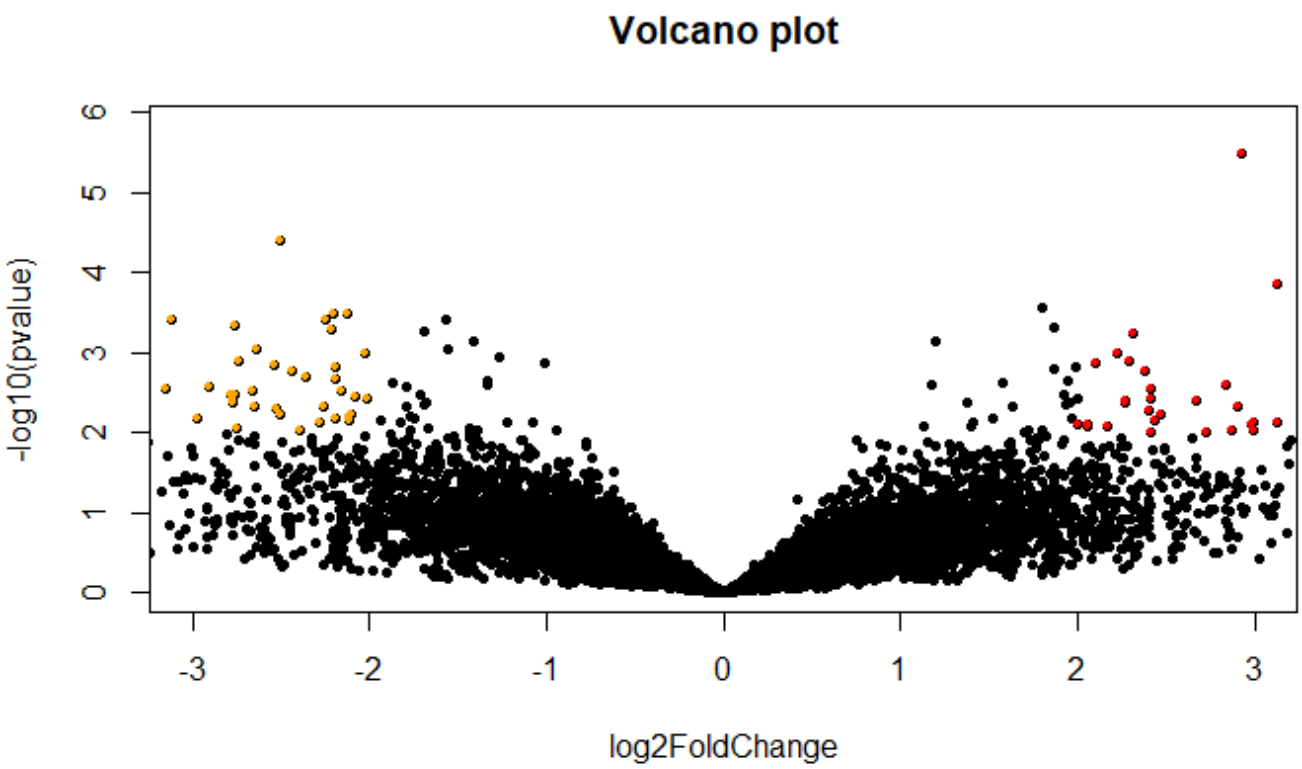
```
with(res, plot(log2FoldChange, -log10(pvalue), pch=20, main="Volcano plot", xlim=c(-
3,3)))

leftyellow <- subset(res,log2FoldChange <= -2 & -log10(pvalue) >= 2)
points(x = leftyellow$log2FoldChange, y = -log10(leftyellow$pvalue),
       col = "orange",cex = 0.5, pch = 19 )
```

Hide

```

rightyellow <- subset(res,log2FoldChange > 2 & -log10(pvalue) >= 2)
points(x = rightyellow$log2FoldChange, y = -log10(rightyellow$pvalue),
       col = "red",cex = 0.5, pch = 19 )
```



Generate a volcano plot to visualize the differential expression results. The log2FoldChange values are plotted on the x-axis, while the negative logarithm of the p-values (-log10(pvalue)) is plotted on the y-axis.

The points in the plot are represented by filled circles (pch=20). The main title of the plot is set as “Volcano plot”. The xlim argument sets the limits for the x-axis, limiting the plot to the range of -3 to 3.

Subsets the res dataframe to extract the rows where the log2FoldChange is less than or equal to -2 and the negative logarithm of the p-value (-log10(pvalue)) is greater than or equal to 2.

The selected data points are stored in the leftyellow object. The points are colored orange

Again, subset the res dataframe to extract the rows where the log2FoldChange is greater than 2 and the negative logarithm of the p-value (-log10(pvalue)) is greater than or equal to 2.

The selected data points are stored in the rightyellow object.The points are colored red.

Hide

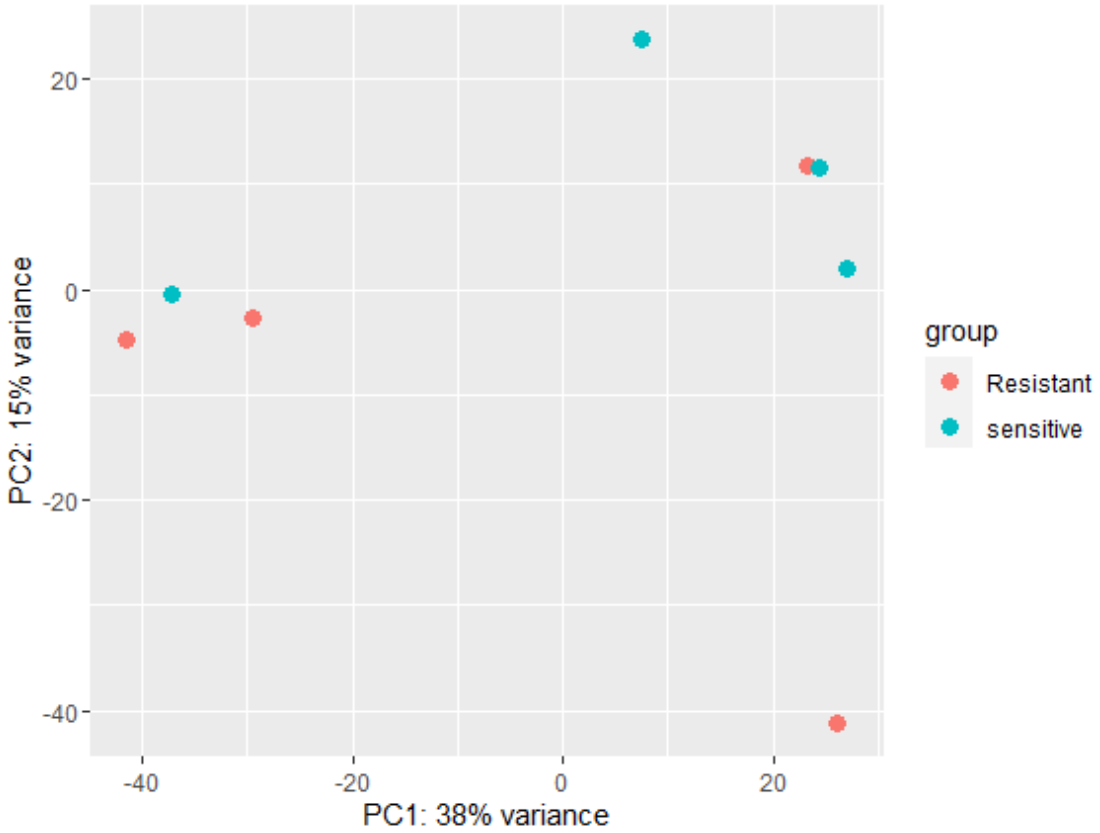
```
vsdata <- vst(dds, blind=FALSE)
```

This line performs variance stabilizing transformation (VST) on the dds object using the vst() function from the DESeq2 package.

The transformed data is stored in the vsdata object. VST is a normalization technique that stabilizes the variance of gene expression across samples, making it more suitable for downstream analyses.

Hide

```
plotPCA(vsdata, intgroup="status")
```



This line generates a PCA plot using the plotPCA() function from the DESeq2 package