DE Analysis of Breast Cancer VDX Data Using Limma Package

- Bioconductor package breastCancerVDX
- Published in Wang et al., 2005 and Minn et al., 2007
- 344 patients: 209 ER+, 135 ER-
- Instruction help from John Blischak

Prepare Data

```
#install necessary R packages
install.packages("BiocManager")
BiocManager::install("Biobase")
BiocManager::install("breastCancerVDX")
BiocManager::install("limma")
BiocManager::install("GO.db")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

'getOption("repos")' replaces Bioconductor standard repositories, see
'help("repositories", package = "BiocManager")' for details.
Replacement repositories:
    CRAN: https://cran.rstudio.com

Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning message:
“package(s) not installed when version(s) same as or greater than current; use
  `force = TRUE` to re-install: 'Biobase'”
Old packages: 'bit', 'curl', 'DBI', 'devtools', 'digest', 'gargle', 'glue',
  'highr', 'isoband', 'openssl', 'pkgload', 'ps', 'ragg', 'readr', 'reprex',
  'rlang', 'roxygen2', 'textshaping', 'timechange', 'uuid', 'whisker', 'withr',
  'boot', 'MASS', 'Matrix', 'nlme'

'getOption("repos")' replaces Bioconductor standard repositories, see
'help("repositories", package = "BiocManager")' for details.
Replacement repositories:
    CRAN: https://cran.rstudio.com

Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning message:
“package(s) not installed when version(s) same as or greater than current; use
  `force = TRUE` to re-install: 'breastCancerVDX'”
Old packages: 'bit', 'curl', 'DBI', 'devtools', 'digest', 'gargle', 'glue',
  'highr', 'isoband', 'openssl', 'pkgload', 'ps', 'ragg', 'readr', 'reprex',
  'rlang', 'roxygen2', 'textshaping', 'timechange', 'uuid', 'whisker', 'withr',
  'boot', 'MASS', 'Matrix', 'nlme'

'getOption("repos")' replaces Bioconductor standard repositories, see
'help("repositories", package = "BiocManager")' for details.
Replacement repositories:
    CRAN: https://cran.rstudio.com

Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning message:
“package(s) not installed when version(s) same as or greater than current; use
  `force = TRUE` to re-install: 'limma'”
Old packages: 'bit', 'curl', 'DBI', 'devtools', 'digest', 'gargle', 'glue',
  'highr', 'isoband', 'openssl', 'pkgload', 'ps', 'ragg', 'readr', 'reprex',
  'rlang', 'roxygen2', 'textshaping', 'timechange', 'uuid', 'whisker', 'withr',
  'boot', 'MASS', 'Matrix', 'nlme'

'getOption("repos")' replaces Bioconductor standard repositories, see
'help("repositories", package = "BiocManager")' for details.
Replacement repositories:
    CRAN: https://cran.rstudio.com

Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning message:
“package(s) not installed when version(s) same as or greater than current; use
  `force = TRUE` to re-install: 'GO.db'”
```

```
#load and attach project necessary R packages
library(Biobase)
library(breastCancerVDX)
library(limma)
library(org.Hs.eg.db)


#data() loads desired dataset
#class() tells us the data type
#`ExpressionSet` is a data structure in R, commonly used in bioinformatics
#holds info related to gene expression experiments
#typically associated with microarray or RNA-sequencing
#organizes components of these experiments
#'ExpressionSet` object:
#1. Expression data: numeric values representing gene expression levels,
# often organized as a matrix where rows correspond to genes and columns correspond to samples.
#2. Feature data: Info about genes, such as gene symbols, annotations, and other metadata.
#3. Phenotype data: Info about the samples, such as sample annotations, treatment conditions, other experimental factors.

data("vdx")
class(vdx)
```

'ExpressionSet'

```
#dim() returns the dimensions of our object
#an ExpressionSet object is metadata > dim() output is a vector
#vector represents (number of genes, number of samples)
dim(vdx)
```

Features: 22283 Samples: 344

```
#remember the ExpressionSet object is made up of three parts
#expression data (exprs) which comes as a matrix
#phenotype data (pData) which comes as a data frame
#and feature data (fData) is metadata which should also come as a data frame
head(exprs(vdx))
```

	VDX_3	VDX_5	VDX_6	VDX_7	VDX_8	VDX_9	VDX_1
1007_s_at	11.965135	11.798593	11.777625	11.538577	12.440765	12.248698	12.25924
1053_at	7.895424	7.885696	7.949535	7.481396	7.714932	7.708049	8.18289
117_at	8.259272	7.052025	8.225930	8.382408	8.603997	7.731319	8.22641
121_at	10.953178	10.666845	10.888819	10.795472	10.193032	10.886687	10.19290
1255_g_at	6.648178	5.452859	6.456149	6.147714	6.601399	6.551516	6.37677
1294_at	9.410451	9.585901	9.422906	9.456970	9.196971	9.831782	9.40429

```
head(pData(vdx))
```

	samplename	dataset	series	id	filename	size	age	
	<chr>	<chr>	<chr>	<int>	<chr>	<lgl>	<int>	<in
VDX_3	VDX_3	VDX	VDX	3	GSM36793.CEL.gz	NA	36	
VDX_5	VDX_5	VDX	VDX	5	GSM36796.CEL.gz	NA	47	
VDX_6	VDX_6	VDX	VDX	6	GSM36797.CEL.gz	NA	44	
VDX_7	VDX_7	VDX	VDX	7	GSM36798.CEL.gz	NA	41	
VDX_8	VDX_8	VDX	VDX	8	GSM36800.CEL.gz	NA	70	
VDX_9	VDX_9	VDX	VDX	9	GSM36801.CEL.gz	NA	62	

```
head(fData(vdx))
```

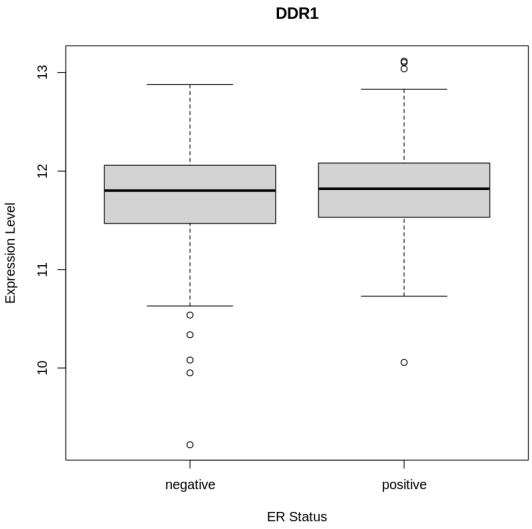
	probe	Gene.title	Gene.symbol	Gene.ID	EntrezGene.ID	UniGene
	<chr>	<chr>	<chr>	<chr>	<fct>	
1007_s_at	1007_s_at	discoidin domain receptor tyrosine kinase 1	DDR1	780	780	
1053_at	1053_at	replication factor C (activator 1) 2, 40kDa	RFC2	5982	5982	
117_at	117_at	heat shock 70kDa protein 6 (HSP70B')	HSPA6	3310	3310	
121_at	121_at	paired box 8	PAX8	7849	7849	
1255_g_at	1255_g_at	guanylate cyclase activator 1A (retina)	GUCA1A	2978	2978	
1294_at	1294_at	ubiquitin-like modifier activating enzyme 7	UBA7	7318	7318	

```
#extract data from the ExpressionSet object and assign it to easy use variable
x <- exprs(vdx)
f <- fData(vdx)
p <- pData(vdx)

#subset desired columns and rename columns for readability
f <- f[, c("Gene.symbol", "EntrezGene.ID", "Chromosome.location")]
colnames(f) <- c("symbol", "entrez", "chrom")

# Recode er as 0 = negative and any number besides 0 to = positive
p[, "er"] <- ifelse(p[, "er"] == 0, "negative", "positive")
p <- p[, c("id", "age", "er")]
```

Explore Data

```
#create a boxplot where we compare the expression levels of the first gene across the different estrogen receptor statuses
#'main' argument defines the main title of our boxplot
boxplot(x[1, ] ~ p[, "er"], main = f[1, "symbol"], xlab = "ER Status", ylab = "Expression Level")
#next create a new ExpressionSet object
#requires expression data (assayData), phenotype data, (phenoData), metadata (featureData)
#AnnotatedDataFrame() associates metadata with rows and columns of a dataset
eset <- ExpressionSet(assayData = x,
                      phenoData = AnnotatedDataFrame(p),
                      featureData = AnnotatedDataFrame(f))
#confirm proper creation and dimensions of our new object
dim(eset)
```

Features: 22283 Samples: 344



limma Pipeline

```
#create a design matrix for a linear model
#~er specifies a model with the response variable (dependent variable) being the levels of the factor variable er
#'data' argument specifies the data used to build the model matrix
#a matrix of dummy variables representing the levels of the factor variable er (in this case, positive or negative)
#design matrix is often used in the context of linear regression or differential expression analysis)
#models the relationship between gene expression and experimental factors
#each column of the design matrix corresponds to a different level of the factor variable
#the matrix is used to set up the linear model for subsequent analyses
design <- model.matrix(~er, data = pData(eset))
head(design, 2)
```

A matrix: 2 × 2 of type dbl

	(Intercept)	erpositive
VDX_3	1	0
VDX_5	1	1

```
colSums(design)
```

(Intercept): 344 erpositive: 209

```
#improve readability with a frequency table
#extract the phenotype data from the ExpressionSet object and then select 'er' column from pData
table(pData(eset)[, "er"])
```

```
negative positive
     135      209
```

```
#fit a model to our gene expression data
#extract the coefficients from the fitted model
#coeffs represent the estimated effect sizes or log-fold changes associated with each factor level in the design matrix
fit <- lmFit(eset, design)
head(fit$coefficients, 3)
```

A matrix: 3 × 2 of type dbl

	(Intercept)	erpositive
1007_s_at	11.725148	0.09878782
1053_at	8.126934	-0.54673000
117_at	7.972049	-0.17342654

```
#estimates the true variability of each gene by borrowing information across genes
#the idea of posterior probability - we update our prior beliefs based on new evidence
```

```
#particularly useful when the number of samples is limited
#t-statistics represent the significance of the coefficients, indicating whether the expression changes are statistically signif
#larger absolute t-statistics generally indicate more significant differences in gene expression
fit <- eBayes(fit)
head(fit$t, 3)
```

A matrix: 3 × 2 of type dbl

	(Intercept)	erpositive
1007_s_at	276.8043	1.817824
1053_at	122.5899	-6.428278
117_at	164.0240	-2.781294

```
#below decides on which genes are differentially expressed based off the moderated t-statistic from above
results <- decideTests(fit[, "erpositive"])
summary(results)
```

```
          erpositive
Down          6276
NotSig       11003
Up            5004
```

```
#intercepts in the context of categorical variables often do not have a meaningful interpretation
#~0 + er specifies that we are only interested in the main effect of the variable 'er' without including an intercept term
#this is considered an orthogonal design, simplifying the interpretation and stabilizing the parameter estimates
design <- model.matrix(~0 + er, data = pData(eset))
head(design, 2)
```

A matrix: 2 × 2 of type dbl

	ernegative	erpositive
VDX_3	1	0
VDX_5	0	1

```
colSums(design)
```

ernegative: 135 erpositive: 209

```
#matrix defines a contrast between two levels of the variable 'er'
#erpositive and ernegative are the two levels of the variable
#the result of the contrast will be stored in a column called 'status' within the contrast matrix
#'levels' argument refers to the design matrix that defines the linear model
#'levels' argument informs the function of the levels and coeffs to contrast
#'cm' will be used to test for differential expression between the two levels of 'er'
#helps assess whether there are significant differences in gene expression associated with the estrogen receptor status
#'cm' matrix defines the specific comparisons of interest
cm <- makeContrasts(status = erpositive - ernegative,
                    levels = design)
cm
```

A matrix: 2 × 1 of type dbl

	status
ernegative	-1
erpositive	1

```
#again
#estimates the true variability of each gene by borrowing information across genes
#the idea of posterior probability - we update our prior beliefs based on new evidence
#particularly useful when the number of samples is limited
#t-statistics represent the significance of the coefficients, indicating whether the expression changes are statistically signif
#larger absolute t-statistics generally indicate more significant differences in gene expression
fit <- lmFit(eset, design)
head(fit$coefficients)
```

A matrix: 6 × 2 of type dbl

	ernegative	erpositive
1007_s_at	11.725148	11.823936
1053_at	8.126934	7.580204
117_at	7.972049	7.798623
121_at	10.168975	10.086393
1255_g_at	5.903189	5.729195
1294_at	9.166436	9.390949

```
#contrasts.fit() applies the contrasts to the linear model
#this adjusted model can then be used for downstream analyses
#such as identifying differentially expressed genes based on the specified comparisons
#contrasts help test specific hypotheses about the differences in gene expression between the conditions defined in the contrast
fit2 <- contrasts.fit(fit, contrasts = cm)
head(fit2$coefficients)
```

A matrix: 6 × 1 of type dbl

	status
1007_s_at	0.09878782
1053_at	-0.54673000
117_at	-0.17342654
121_at	-0.08258267
1255_g_at	-0.17399402
1294_at	0.22451339

```
fit2 <- eBayes(fit2)
results <- decideTests(fit2)
summary(results)
```

```
       status
Down     6276
NotSig  11003
Up       5004
```

```
#extracts the top-ranked genes based on certain statistical measures from a linear model fit
#ranking genes based on some measure of interest (fold changes, t-statistics, p-values, or other statistical measures)
#example with arguments for future use - topTable(fit, coef = "your_contrast", number = 10, sort.by = "p.value")
topTable(fit2)
```

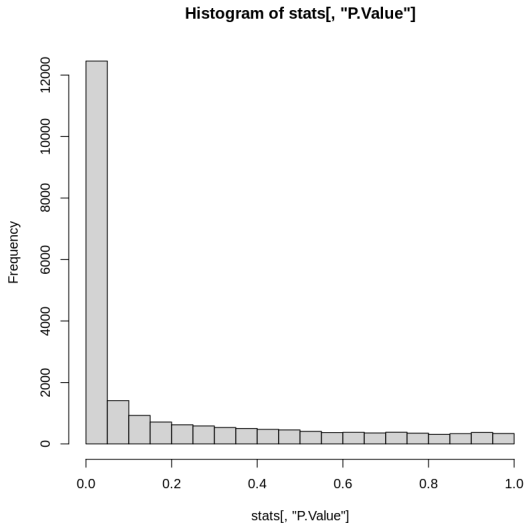
A data.frame: 10 × 9

	symbol	entrez	chrom	logFC	AveExpr	t	P.Value	ad
	<chr>	<fct>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	
205225_at	ESR1	2099	6q25.1	3.762901	11.377735	22.68392	2.001001e-70	4
209603_at	GATA3	2625	10p15	3.052348	9.941990	18.98154	1.486522e-55	1
209604_s_at	GATA3	2625	10p15	2.431309	13.185334	17.59968	5.839050e-50	4
212956_at	TBC1D9	23158	4q31.21	2.157435	11.702942	17.48711	1.665700e-49	9
202088_at	SLC39A6	25800	18q12.2	1.719680	13.119496	17.30104	9.412084e-49	4
212496_s_at	KDM4B	23030	19p13.3	1.459843	10.703942	16.85070	6.188671e-47	2
							1.074845e-	3

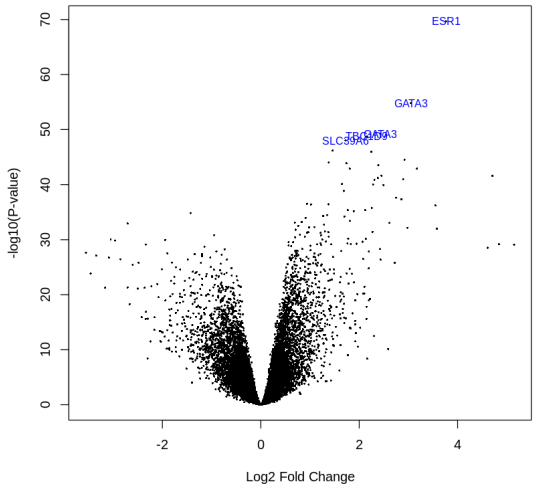
```
#example with arguments for future use - topTable(fit, coef = "your_contrast", number = 10, sort.by = "p.value")
stats <- topTable(fit2, number = nrow(fit2), sort.by = "none")
dim(stats)
```

Visualize and Stats

```
#create a histogram off 'stats' column "P.Value"
hist(stats[, "P.Value"])
```




```
#visualize with a volcano plot
#compare names to topTable results
volcanoplot(fit2, highlight = 5, names = fit2$genes[, "symbol"])
```




```
#The Fisher's Exact Test will assess whether the distribution of outcomes across the groups is
#significantly different from what would be expected by chance
#this can be particularly useful when dealing with small sample sizes or sparse data
#p-value from the test helps you decide whether the observed association is statistically significant
#1000 genes (10% in gene set), 100 are DE (10% in gene set)
#our output implies no association on multiple fronts
fisher.test(matrix(c(10, 100, 90, 900), nrow = 2))
```

```
        Fisher's Exact Test for Count Data

data:  matrix(c(10, 100, 90, 900), nrow = 2)
p-value = 1
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.4490765 2.0076377
sample estimates:
odds ratio
         1
```

```
#1000 genes (10% in gene set), 100 are DE (30% in gene set)
#with the increase of DE in our gene set, now our output is significant
#small p-value (less that 0.05), ours is drastically less,
#indicating that the observed distribution is highly unlikely under the assumption of no association (null hypothesis)
#the confidence interval not including 1 provides strong evidence to reject the null hypothesis,
#suggesting a statistically significant association between the two categorical variables
#odds ratio >1 in conjunction with the CIs also supports the association between the two categorical variables
fisher.test(matrix(c(30, 100, 70, 900), nrow = 2))
```

```
        Fisher's Exact Test for Count Data

data:  matrix(c(30, 100, 70, 900), nrow = 2)
p-value = 1.88e-07
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 2.306911 6.320992
sample estimates:
odds ratio
  3.850476
```

Enrichment

Lastly, enrichment analysis, we will utilize two databases to investigate pathways (KEGG) and functional analysis (GO).

```
#start preparing for enrichment
head(fit2$genes, 3)
```

A data.frame: 3 × 3

	symbol	entrez	chrom
	<chr>	<fct>	<chr>
1007_s_at	DDR1	780	6p21.3
1053_at	RFC2	5982	7q11.23
117_at	HSPA6	3310	1q23

KEGG (Kyoto Encyclopedia of Genes and Genomes):

KEGG is primarily focused on pathways and systems-level information. It provides a comprehensive resource for understanding the high-level functions and utilities of biological systems, emphasizing pathways and networks.

KEGG includes information on metabolic pathways, signaling pathways, diseases, drugs, and more. It often represents molecular interactions and reactions in a network format.

KEGG is commonly used for pathway analysis, studying how genes or proteins contribute to specific biological processes or pathways.

```
#enrichment analysis
#Kyoto Encyclopedia of Genes and Genomes (KEGG)
#associate genes by their Entrez ID
#kegga() function performs gene enrichment analysis
#takes fitted model, associates gene IDs, and specifies species creating an 'enrich_kegg' object
#then extract the top 4 KEGG pathways
entrez <- fit2$genes[, "entrez"]
```

```
enrich_kegg <- kegga(fit2, geneid = entrez, species = "Hs")
topKEGG_results <- topKEGG(enrich_kegg, number = 4)
```

```
head(topKEGG_results, 3)
```

A data.frame: 3 × 6

	Pathway	N	Up	Down	P.Up	P.Down
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
hsa04110	Cell cycle	140	41	96	0.5251773	2.718600e-16
hsa05166	Human T-cell leukemia virus 1 infection	204	50	124	0.9435467	2.010120e-14

GO (Gene Ontology):

GO is focused on annotating genes and gene products in terms of their associated biological processes, cellular components, and molecular functions. It provides a standardized vocabulary for describing the attributes of genes and their products.

GO terms are organized into three main ontologies—Biological Process (BP), Cellular Component (CC), and Molecular Function (MF). Each term represents a specific aspect of gene function or location.

GO is widely used for gene set enrichment analysis to understand the functional implications of a list of genes. It helps researchers identify overrepresented functional categories within a gene set.

```
enrich_go <- goana(fit2, geneid = entrez, species = "Hs")
topGO(enrich_go, ontology = c("BP"), number = 3)
```

A data.frame: 3 × 7

	Term	Ont	N	Up	Down	P.Up	P.Down
	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
GO:0002376	immune system process	BP	2020	455	944	1	3.461143e-33
GO:0006955	immune response	BP	1400	276	680	1	4.419296e-29

```
Start coding or generate with AI.
```