DE Analysis of ALL, AML, CML Data with Limma

- Bioconductor package: leukemiasEset
- by Kohlmann et al. 2008, Haferlach et al. 2010
- class instruction by John Blischak and DataCamp

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

```
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: 'leukemiasEset'”
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'”
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)

Installing package(s) 'fgsea'

also installing the dependencies ‘formatR’, ‘lambda.r’, ‘futile.options’, ‘futile.logger’, ‘snow’, ‘Rcpp’, ‘BiocParallel’,


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'
```

```
#load and attach project necessary R packages
library(Biobase)
library(leukemiasEset)
library(limma)
library(fgsea)
```

```
#load desired dataset and confirm that object is ExpressionSet
data("leukemiasEset")
class(leukemiasEset)
```

'ExpressionSet'

```
#set easy use variable and view the dimensions
eset <- leukemiasEset
dim(eset)
```

Features: 20172 Samples: 60

```
head(exprs(eset))
```

	GSM330151.CEL	GSM330153.CEL	GSM330154.CEL	GSM330157.CEL
ENSG00000000003	3.386743	3.687029	3.360517	3.459388
ENSG00000000005	3.539030	3.836208	3.246327	3.063286
ENSG00000000419	9.822758	7.969170	9.457491	9.591018
ENSG00000000457	4.747283	4.866344	4.981642	5.982854
ENSG00000000460	3.307188	4.046402	5.529369	4.619444
ENSG00000000938	8.230721	7.945818	6.411830	6.882017

```
head(fData(eset))
```

A data.frame: 6 × 0

ENSG00000000003

ENSG00000000005

ENSG00000000419

ENSG00000000457

ENSG00000000460

ENSG00000000938

```
#add a column of gene identifiers to the featureData slot of the ExpressionSet object,
#allowing for easy association of genes with their expression values in subsequent analyses
#create an AnnotatedDataFrame object which is a special Bioconductor structure that associates metadata with rows or columns of
#here we are associating gene identifiers with expression values
#create a data frame with column called "ensembl", extract the rownames (our gene identifiers) from the expression matrix of our
#'stringsAsFactors' to FAlSE ensures our gene identifiers are represented as 'characters' and not as 'factors'
#assign our AnnotatedDataFrame object to the fData slot of our eset object
featureData(eset) <- AnnotatedDataFrame(data.frame(ensembl = rownames(exprs(eset)),
                                                   stringsAsFactors = FALSE))
head(fData(eset))
```

A data.frame: 6 × 1

	ensembl <chr>
1	ENSG00000000003
2	ENSG00000000005
3	ENSG00000000419
4	ENSG00000000457
5	ENSG00000000460
6	ENSG00000000938

```
head(pData(eset))
```

A data.frame: 6 × 5

	Project	Tissue	LeukemiaType	LeukemiaTypeFullName	
	<fct>	<chr>	<fct>	<fct>	
GSM330151.CEL	Mile1	BoneMarrow	ALL	Acute Lymphoblastic Leukemia	c_ALI w
GSM330153.CEL	Mile1	BoneMarrow	ALL	Acute Lymphoblastic Leukemia	c_ALI w
GSM330154.CEL	Mile1	BoneMarrow	ALL	Acute Lymphoblastic Leukemia	c_ALI w
GSM330157.CEL	Mile1	BoneMarrow	ALL	Acute Lymphoblastic Leukemia	c_ALI w

```
#create a frequency table to evaluate the different values and occurrences within "LeukemiaType"
table(pData(eset)[, "LeukemiaType"])
```

```
ALL AML CLL CML NoL 
 12  12  12  12  12 
```

```
#we are evaluating only ALL, AML, and CML
#so subset to only include ALL, AML, and CML
#%in% operator returns a logical vector indicating whether each element of the left-hand side is in the right-hand side
#then recheck dimensions to ensure data has not been lost
eset <- eset[, pData(eset)[, "LeukemiaType"] %in% c("ALL","AML", "CML")]
dim(eset)
```

```
Features:        20172 Samples:        36
```

```
#Ensure that our desired phenotype data is easy to read
#follow similar steps as we did for fData
#create an AnnotatedDF object by creating a data frame with column 'type' subsetted off of column 'LeukemiaType'
#ensure that our "LeukemiaType" is seen as 'characters' and not 'factors'
phenoData(eset) <- AnnotatedDataFrame(data.frame(type = as.character(pData(eset)[, "LeukemiaType"]),
                                                 stringsAsFactors = FALSE))
head(pData(eset))
```

A data.frame: 6 × 1

	type
	<chr>
1	ALL
2	ALL
3	ALL
4	ALL
5	ALL
6	ALL

```
#assign new column names to the samples in our ExpressionSet object (eset)
#sprintf() function is used to create formatted character strings
#generate column names like "Sample_01", and so on, up to the number of columns in our eset object
#colnames() function is then used to set these newly generated names as the column names for your ExpressionSet
colnames(eset) <- sprintf("Sample_%01d", 1:ncol(eset))
exprs(eset)[1:3, ]
```

	Sample_1	Sample_2	Sample_3	Sample_4	Sample_5	Sample_6
ENSG00000000003	3.386743	3.687029	3.360517	3.459388	3.598589	3.266450
ENSG00000000005	3.539030	3.836208	3.246327	3.063286	3.307543	2.898742
ENSG00000000419	9.822758	7.969170	9.457491	9.591018	9.863687	9.357091

```
#check-up on dimensions
dim(eset)
```

```
Features:    20172 Samples:    36
```

```
#check out the new look of our phenotypic data
head(pData(eset), 3)
```

A data.frame: 3 × 1

	type
	<chr>
Sample_1	ALL
Sample_2	ALL
Sample_3	ALL

```
#re-check frequency, evaluate for possible errors
table(pData(eset)[, "type"])
```

```
ALL AML CML 
 12  12  12 
```

```
#create a matrix of predictor (dummy) variables used to model the response variable within our statistical model
#~0 + type > include our 'type' variables and no intercept term
#no intercept term since dealing with categorical variables with a set of binary indicators
design <- model.matrix(~0 + type, data = pData(eset))
head(design, 3)
```

A matrix: 3 × 3 of type dbl

	typeALL	typeAML	typeCML
Sample_1	1	0	0
Sample_2	1	0	0
Sample_3	1	0	0

```
#Our tests:
#AML v. ALL: β2−β1=0
#CML v. ALL: β3−β1=0
#CML v. AML: β3−β2=0
#makeContrasts() is used to specify linear combinations of coefficients in a linear model
#identifies the differences in the 'levels' (AML, ALL, and CML) within our variable 'type' that we want to contrast
#The resulting object 'cm' is a contrast matrix
#that can be used in subsequent linear modeling (e.g., with eBayes or topTable)
#to test and summarize differences between the specified levels of the categorical variable

cm <- makeContrasts(AMLvALL = typeAML - typeALL,
                    CMLvALL = typeCML - typeALL,
                    CMLvAML = typeCML - typeAML,
                    levels = design)
cm
```

A matrix: 3 × 3 of type dbl

	AMLvALL	CMLvALL	CMLvAML
typeALL	-1	-1	0
typeAML	1	0	-1
typeCML	0	1	1

```
# Fit coefficients
fit <- lmFit(eset, design)
# Fit contrasts
fit2 <- contrasts.fit(fit, contrasts = cm)
# Calculate t-statistics
fit2 <- eBayes(fit2)
# Summarize results
results <- decideTests(fit2)
summary(results)
```

```
           AMLvALL CMLvALL CMLvAML
    Down       898    3401    1890
    NotSig   18323   13194   16408
    Up         951    3577    1874
```

```
#AML v ALL > the negative values suggest a down regulation in AML compared to ALL
#CML v ALL > the negative values suggest a down regulation in CML compaared to ALL
#CML v AML > the negative values suggest a down regulation in CML compared to AML
#AveExpr represents the average expression value across samples on the log2 scale
#F represents the F-statistic which measures the difference in variance between groups compared to within groups
#higher F values indicate greater variability between groups
#the p-value associated with the hypothesis test that the coefficients for the contrasts are equal to zero
#smaller p-values suggest stronger evidence against the null hypothesis
#adjusted p-value are obtained using methods like Benjamini-Hochberg correction
#accounts for multiple testing and is used to control the False Discovery Rate
#genes with significant p-values and low adjusted p-values should be considered potentially differentially expressed
topTable(fit2)
```

A data.frame: 10 × 8

	ensembl	AMLvALL	CMLvALL	CMLvAML	AveExpr	
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	
ENSG00000164330	ENSG00000164330	-5.8859760	-5.818786	0.06719034	5.831635	72
ENSG00000177455	ENSG00000177455	-5.3510403	-5.642022	-0.29098183	7.624318	28
ENSG00000102935	ENSG00000102935	-5.3059285	-5.156976	0.14895208	5.198873	21
ENSG00000169575	ENSG00000169575	-6.4841524	-6.751170	-0.26701745	6.476759	19
ENSG00000167483	ENSG00000167483	-3.9424864	-3.768933	0.17355362	6.031832	17
ENSG00000160180	ENSG00000160180	-0.1646271	4.404545	4.56917166	6.371601	16

```
#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)
```

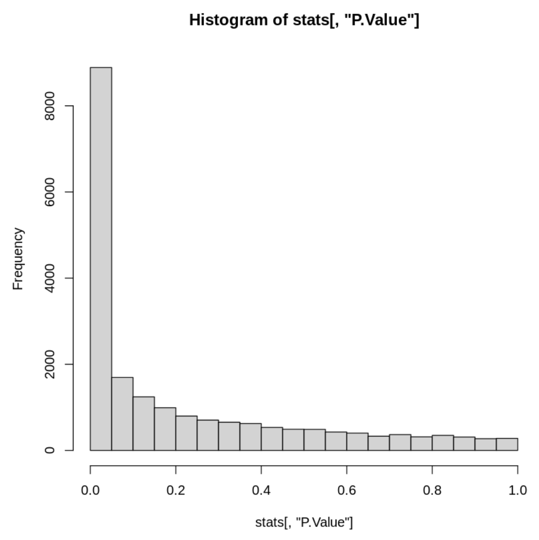
20172 · 8

```
head(stats)
```

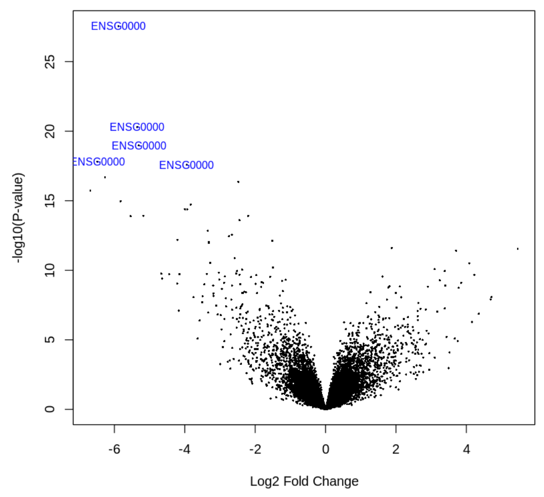
A data.frame: 6 × 8

	ensembl	AMLvALL	CMLvALL	CMLvAML	AveExpr
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
ENSG00000000003	ENSG00000000003	0.15387044	0.16772375	0.01385331	3.549314
ENSG00000000005	ENSG00000000005	0.09433822	0.12452800	0.03018978	3.326386
ENSG00000000419	ENSG00000000419	0.08073637	-0.37380408	-0.45454045	9.241923
ENSG00000000457	ENSG00000000457	0.39086669	-0.52768287	-0.91854956	5.529438

```
hist(stats[, 'P.Value'])
```



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



```
#create a contigency table off above results
#       AMLvALL CMLvALL CMLvAML
#Down       898    3401    1890
#NotSig   18323   13194   16408
#Up         951    3577    1874
#can also just namme summary(results) from above

contingency_table <- matrix(c(898, 3401, 1890, 18323, 13194, 16408, 951, 3577, 1874), nrow = 3, byrow = TRUE)

# Add row and column names
rownames(contingency_table) <- c("Down", "NotSig", "Up")
colnames(contingency_table) <- c("AMLvALL", "CMLvALL", "CMLvAML")

# Print the contingency table
print(contingency_table)
```

```
           AMLvALL CMLvALL CMLvAML
    Down       898    3401    1890
    NotSig   18323   13194   16408
    Up         951    3577    1874
```

```
#too large for this workspace but would apply Fisher's test
#would help assess whether the distribution of outcomes across the groups is
```

```
#significantly different from what would be expected by chance
#fisher.test(contingency_table)
```

Start coding or generate with AI.