Differential Expression Meta-Analysis with DExMA package

Juan Antonio Villatoro-García [1,2] ***and Pedro Carmona-Sáez*** [*] [1,2]

[1] Department of Statistics and Operational Research. University of Granada
[2] Bioinformatics Unit. GENYO, Centre for Genomics and Oncological Research

[*] pedro.carmona@genyo.es

November 13, 2023

Abstract

DExMA (Differential Expression Meta-Analysis) performs all the necessary steps of differential expression meta-analysis considering the possible existence of missing genes. It controls the existence of unmeasured genes by two approaches: considering the genes that are contained in a proportion of datasets (set by the users) or imputing the expression values of the unmeasured genes using the *Knn method* in the space of samples. In addition, it allows to apply quality controls, download GEO datasets and show a graphical representation of the results.

packageVersionDExMA 1.10.4

Contents

1 Introduction

DExMA is a package designed to perform gene expression meta-analysis. Gene expression meta-analysis comprises a set of methods that combine the results of several differential expression studies into a single common result [1]. Furthermore, this package has the advantage that it allows to takes into account those genes that are not measured in a study. The first approach to control these missing genes, **DExMA** offers two options: apply the k-nearest neighbors method (knn method) in the space of samples to impute the expression values of these missing genes [2]. The second option is to take into account those genes that are contained in at least a certain proportion (set by the user) of datasets. The use of only the genes common to all the datasets could lead to the loss of some genes that are not measured in a single study, which would lead to the loss of information. Due to this fact, **DExMA** package is very useful to work with Microarray data, because this type of data is what usually produces the existence of uncommon genes between datasets with different annotations. However, previously normalized RNA-Seq data can also be used, as well as, both Microarray data and normalized RNA-Seq data at the same time.

DExMA package has implemented methods from the three main types of gene expression meta-analysis [1] meta-analysis based on effects sizes combination and meta-analysis based on p-values combination. Once one of the methods has been applied, **DExMA** package provides a specific and adapted results table. Moreover, this package contains some functions that allows the user to carry out a previous quality control in order to results obtained are more reliable. Finally, this package provides some additional function that, for example, help the user to download public Microarray data from NCBI GEO public database [3] or to visualize the significant genes in a heatmap.

This document gives a tutorial-style introduction to all the steps that must be carry out in order to properly perform gene expression meta-analysis by making use of **DExMA** package.

2 Previous steps: Meta-analysis object

DExMA uses a specific object as input, which is a list of nested lists where each nested list corresponds to a study. This object can be created directly by the users or they can use *createObjectMA()* function to create it.

For the examples that are going to be shown, synthetic data will be used. We load the sample data into our R session.

```
> library(DExMA)
> data("DExMAExampleData")
```

- listMatrixEX: a list of four expression arrays
- listPhenodatas: a list of the four phenodata corresponding to the four expression arrays
- listExpression: a list of four ExpressionSets object. It contains the same information as listMatrixEX and listPhenodatas
- ExpressionSetStudy5: an ExpressionSet object similar to the ExpressionSets objects of listExpression.
- maObjectDif: the meta-analysis object created from the listMatrixEX and listPhenodatas objects.

- maObject: the meta-analysis object after setting all the studies in Official Gene Symbol annotation

2.1 Meta-analysis object creation (objectMA)

As previously stated, the meta-analysis input in DExMA is a list of nested lists. Each nested list contains two elements:

- A gene expression matrix with genes in rows and samples in columns
- A vector of 0 and 1 indicating the group of each sample. 0 represents reference group (usually controls) and 1 represents experimental group (usually cases).

This object can be created directly by the user or we can make use of *createObjectMA()* function, which creates the *objectMA* after indicating how the reference and experimental groups are identified.

createObjectMA() function allows to create the object needed to perform meta-analysis. In this case, it is necessary to indicate as input of the function the variables that contain the experimental and reference groups:

- listEX: a list of dataframes or matrix (genes in rows and samples in columns). A list of ExpressionSets can be used too:

```
> #List of expression matrices
> data("DExMAExampleData")
> ls(listMatrixEX)

[1] "Study1" "Study2" "Study3" "Study4"

> head(listMatrixEX$Study1)

           Sample1  Sample2  Sample3   Sample4
100859927 5.439524 6.253319 2.926444 4.4304023
8086      5.769823 5.971453 1.831349 4.0466288
8212      7.558708 5.957130 2.365252 3.4352889
65985     6.070508 7.368602 2.971158 3.7151784
729522    6.129288 5.774229 3.670696 3.9171749
13        7.715065 7.516471 1.349453 0.3390772

> #List of ExpressionSets
> ls(listExpressionSets)

[1] "Study1" "Study2" "Study3" "Study4"

> listExpressionSets$Study1

ExpressionSet (storageMode: lockedEnvironment)
assayData: 200 features, 4 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: Sample1 Sample2 Sample3 Sample4
  varLabels: condition gender organism race
  varMetadata: labelDescription
featureData: none
```

```
experimentData: use 'experimentData(object)'
Annotation:
```

- listPheno: a list of phenodatas (samples in rows and covariables in columns). If the object listEX is a list of ExpressionSets this element can be null.

```
> data("DExMAExampleData")
> #Example of a phenodata object
> ls(listPhenodatas)

[1] "Study1" "Study2" "Study3" "Study4"

> listPhenodatas$Study1

        condition gender     organism race
Sample1  Diseased Female Homo Sapiens   AA
Sample2  Diseased Female Homo Sapiens   AA
Sample3   Healthy Female Homo Sapiens   AA
Sample4   Healthy Female Homo Sapiens    H
```

- namePheno: a list or vector of the different column names or column positions from the pheno used for perfoming the comparison between groups. Each element of namePheno correspont to its equivalent element in the listPheno. (default a vector of 1, all the first columns of each elements of listPheno are selected)
- expGroups: a list or vector of the group names or positions from namePheno variable used as experimental group (cases) to perform the comparison (default a vector of 1, all the first groups are selected).
- refGroups: a list or vector of the group names or positions from namePheno variable used as reference group (controls) to perform the comparison (default a vector of 2, all the second groups are selected).

It is important to note that if any element does not belong to the experimental or the reference group, that sample is not taken into account in the creation of meta-analysis object.

Here, we have included an example to show how exactly the function is used:

Since this function can be a bit complicated if there are many datasets, we recommend creating a vector to keep the column names of the phenodatas that contains the variable that identifies the groups to compare (*namePheno* argument). Moreover, we should create two others lits to indicate how to identify experimental (cases) and reference (controls) groups in these variables (*expGroups* and *refGroups* arguments).

If we look at the example phenodatas list we have the following four objects:

```
> listPhenodatas$Study1

        condition gender     organism race
Sample1  Diseased Female Homo Sapiens   AA
Sample2  Diseased Female Homo Sapiens   AA
Sample3   Healthy Female Homo Sapiens   AA
Sample4   Healthy Female Homo Sapiens    H
```

In the "Study1" phenoData, the groups variable is "condition". Experimental group is named as "Diseased" and reference group as "Healthy".

```
> listPhenodatas$Study2

         condition gender     organism race
Sample5   Diseased Female Homo Sapiens   AA
Sample6   Diseased Female Homo Sapiens   AA
Sample7        ill   Male Homo Sapiens    C
Sample8    Healthy Female Homo Sapiens    H
Sample9    control Female Homo Sapiens    H
Sample10   control   Male Homo Sapiens    H
```

In the "Study2" phenoData, the groups variable is "condition". Experimental group is named as "Diseased" or "ill" and reference group as "Healthy" or "control"

```
> listPhenodatas$Study3

            state gender     organism race
Sample11 Diseased Female Homo Sapiens   AA
Sample12 Diseased Female Homo Sapiens   AA
Sample13  Healthy Female Homo Sapiens   AA
Sample14  Healthy Female Homo Sapiens    H
```

In the "Study3" phenoData, the groups variable is "state". Experimental group is named as "Diseased" and reference group as "Healthy".

```
> listPhenodatas$Study4

           state gender     organism race
Sample15     ill Female Homo Sapiens   AA
Sample16     ill Female Homo Sapiens   AA
Sample17     ill   Male Homo Sapiens   AA
Sample18 control Female Homo Sapiens    C
Sample19 control Female Homo Sapiens    H
Sample20 control   Male Homo Sapiens    H
```

In this phenoData, the groups variable is "state". Experimental group is named as "ill" and reference group as "control".

We all this information we can create the vector for *namePheno* argument and the two list for *expGroups* and *refGroups*:

```
> phenoGroups = c("condition", "condition", "state", "state")
> phenoCases = list(Study1 = "Diseased", Study2 = c("Diseased", "ill"),
+                     Study3 = "Diseased", Study4 = "ill")
> phenoControls = list(Study1 = "Healthy", Study2 = c("Healthy", "control"),
+                     Study3 = "Healthy", Study4 = "control")
```

Then, we can apply more easily *createObjectMA()* function:

```
> newObjectMA <- createObjectMA(listEX=listMatrixEX,
+                               listPheno = listPhenodatas,
+                               namePheno=phenoGroups,
+                               expGroups=phenoCases,
+                               refGroups = phenoControls)
> #Study 1
> head(newObjectMA[[1]][[1]])

           Sample1  Sample2  Sample3   Sample4
100859927 5.439524 6.253319 2.926444 4.4304023
8086      5.769823 5.971453 1.831349 4.0466288
8212      7.558708 5.957130 2.365252 3.4352889
65985     6.070508 7.368602 2.971158 3.7151784
729522    6.129288 5.774229 3.670696 3.9171749
13        7.715065 7.516471 1.349453 0.3390772

> newObjectMA[[1]][[2]]

[1] 1 1 0 0

> #Study 2
> head(newObjectMA[[2]][[1]])

           Sample5  Sample6  Sample7  Sample8  Sample9 Sample10
100859927 4.367690 6.648252 5.786897 2.820073 2.688687 3.331881
8086      5.937004 5.914434 7.124650 3.234439 3.228559 4.521235
8212      5.294553 4.092231 5.194514 3.368330 2.750491 5.369983
65985     5.685822 5.900363 5.539580 2.153294 3.166798 3.162232
729522    5.733054 5.553966 6.592146 3.275042 2.659061 2.452524
13        6.153159 5.435909 6.991360 4.860632 3.751608 3.121853

> newObjectMA[[2]][[2]]

[1] 1 1 1 0 0 0
```

The result obtained is the proper object to perform meta-analysis (objectMA).

2.2 Adding a new dataset to the meta-analysis object

It may happen that once the meta-analysis object is created we want to add a new dataset before doing the meta-analysis. **DExMA** provides the *elementObjectMA()* function, which allows the creation of an element ofthe meta-analysis object. This function contains the following arguments:

- expressionMatrix: a dataframe or matrix that contanining genes in rows and samples in columns. An ExpressionSet object can be used too.
- pheno: a data frame or a matrix containing samples in rows and covariates in columns. If NULL (default), pheno is extracted from the ExpressionSet object
- groupPheno: the column name or position from pheno where experimental group (cases) and reference group (control) are identified
- expGroups: a vector of the names or positions from groupPheno variable used as experimental group (cases). By default the first group (character) is taken
- refGroups: a vector of the names or positions from groupPheno variable used as reference group (control). By default the second group (character) is taken.

As with the *createObjectMA()* function, if any element does not belong to the experimental or the reference group, that sample is not included in the creation of the object.

Here we provided an example of the use of this function, in which we create an element of the meta-analysis object from the information of the "*Study 2*" of the *listExpressionSets* object:

```
> data("DExMAExampleData")
> ExpressionSetStudy5

ExpressionSet (storageMode: lockedEnvironment)
assayData: 200 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  rowNames: newSample1 newSample2 ... newSample6 (6 total)
  varLabels: condition gender organism race
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> library(Biobase)
> pData(ExpressionSetStudy5)

           condition gender     organism race
newSample1  Diseased Female Homo Sapiens   AA
newSample2  Diseased Female Homo Sapiens   AA
newSample3       ill   Male Homo Sapiens    C
newSample4   Healthy Female Homo Sapiens    H
newSample5   control Female Homo Sapiens    H
newSample6   control   Male Homo Sapiens    H
```

We had to load *Biobase* package in order to the information in the *ExpressionSet*. In the phenoData we can observe that groups variable is "condition". In addition, Experimental group is name as "Diseased" or "ill" and reference group as "Healthy" or "control"

```
> newElem <-elementObjectMA(expressionMatrix = ExpressionSetStudy5,
+                             groupPheno = "condition",
+                             expGroup = c("Diseased", "ill"),
+                             refGroup = c("Healthy", "control"))
> head(newElem[[1]])

          newSample1 newSample2 newSample3 newSample4 newSample5 newSample6
100859927   5.867690   8.148252   7.286897   4.320073   4.188687   4.831881
8086        7.437004   7.414434   8.624650   4.734439   4.728559   6.021235
8212        6.794553   5.592231   6.694514   4.868330   4.250491   6.869983
65985       7.185822   7.400363   7.039580   3.653294   4.666798   4.662232
729522      7.233054   7.053966   8.092146   4.775042   4.159061   3.952524
13          7.653159   6.935909   8.491360   6.360632   5.251608   4.621853

> head(newElem[[2]])

[1] 1 1 1 0 0 0
```

As we can see, we obtain a list that has the same structure as the elements of the meta-analysis object (*objectMA*). This new element can be added to a *objectMA* that has been created previously:

```
> newObjectMA2 <- newObjectMA
> newObjectMA2[[5]] <- newElem
> head(newObjectMA2[[5]][[1]])

          newSample1 newSample2 newSample3 newSample4 newSample5 newSample6
100859927   5.867690   8.148252   7.286897   4.320073   4.188687   4.831881
8086        7.437004   7.414434   8.624650   4.734439   4.728559   6.021235
8212        6.794553   5.592231   6.694514   4.868330   4.250491   6.869983
65985       7.185822   7.400363   7.039580   3.653294   4.666798   4.662232
729522      7.233054   7.053966   8.092146   4.775042   4.159061   3.952524
13          7.653159   6.935909   8.491360   6.360632   5.251608   4.621853

> newObjectMA2[[5]][[2]]

[1] 1 1 1 0 0 0
```

Moreover, an advantage of this function is that it can be used to create one by one all the elements and finally join all of them to create the meta-analysis object.

3 Performing Meta-analysis

DExMA package contains the main gene expression meta-analysis methods:

- Meta-analysis based on effect size combination: Fixed Effects Model (FEM) and Random Effects Model (REM).
- Meta-analysis based on P-value combination: Fisher's method, Stouffer's method, Wilkonson's method (maxP), Tippett's method (minP) and Aggregated Cauchy Association Test method (ACAT).

These methods can be applied directly, but it is advisable to apply some previous **DExMA** function to ensure the results are accurate.

3.1 Gene annotation and quality controls

Before performing the meta-analysis, all the genes must be in the same annotation and a quality control should be done in order to obtain reliable results [1]. **DExMA** provides some useful functions to help the user to do it.

3.1.1 Setting all the datasets in the same annotation

All genes must be in the same annotation in order to perform the meta-analysis successfully and avoid incorrect interpretations of the results. In cases where all datasets do not have the same gene ID, **DExMA** contains a function called *allSameID()* that allows to annotated all the datasets with the same gene ID. Specifically, the function allows to annotate those datasets that use the Official Gene Symbol, Entrez or Ensembl. The inputs of this function are:

- *objectMA*: the meta-analysis object of **DExMA** package. The result obtained by *createObjectMA()* function should be used.
- *finalID*: a character that indicates the final gene ID that all the studies will have.
- *organism*: a character that indicates the organism that the datasets belong to. Use *avaliableOrganism* function to see the organism admitted.

Here we include an example of how to use this function. In this example we have used "newObjectMA" that has been created before. The first two expression arrays are annotated in "entrez", the third expression matrix in "Official Gene Symbol" and the last one in "Official Gene Symbol" with some synonyms:

```
> rownames(newObjectMA$Study1$mExpres)[1:20]

 [1] "100859927" "8086"      "8212"      "65985"     "729522"    "13"
 [7] "344752"    "126767"    "343066"    "51166"     "79719"     "22848"
[13] "14"        "15"        "16"        "57505"     "80755"     "132949"
[19] "60496"     "10157"

> rownames(newObjectMA$Study2$mExpres)[1:20]

 [1] "100859927" "8086"      "8212"      "65985"     "729522"    "13"
 [7] "344752"    "126767"    "343066"    "51166"     "79719"     "22848"
[13] "14"        "15"        "16"        "57505"     "80755"     "132949"
[19] "60496"     "10157"

> rownames(newObjectMA$Study3$mExpres)[1:20]

 [1] "AAA4"     "AAAS"     "AABT"     "AACS"     "AACSP1"   "AADAC"
 [7] "AADACL2"  "AADACL3"  "AADACL4"  "AADAT"    "AAGAB"    "AAK1"
[13] "AAMP"     "AANAT"    "AARS1"    "AARS2"    "AARSD1"   "AASDH"
[19] "AASDHPPT" "AASS"

> rownames(newObjectMA$Study4$mExpres)[1:20]

 [1] "AAA4"      "ADRACALIN" "AABT"      "ACSF1"     "AACSP1"    "AADAC"
 [7] "AADACL2"   "AADACL3"   "AADACL4"   "AADAT"     "PPKP1"     "AAK1"
[13] "AAMP"      "AANAT"     "AARS1"     "AARS2"     "AARSD1"    "AASDH"
[19] "AASDHPPT"  "AASS"
```

We can use *avaliableIDs* and *avaliableOrganism* in order to know how to write in the function the *finalID* and the *organism* arguments:

```
> head(avaliableIDs)

[1] "Entrez"     "Ensembl"    "GeneSymbol"

> avaliableOrganism

 [1] "Bos taurus"               "Caenorhabditis elegans"
 [3] "Canis familiaris"         "Danio rerio"
 [5] "Drosophila melanogaster"  "Gallus gallus"
 [7] "Homo sapiens"             "Mus musculus"
 [9] "Rattus norvegicus"        "Arabidopsis thaliana"
[11] "Saccharomyces cerevisiae" "Escherichia coli"
```

Then, we are ready to run the *allSameID()* function. We are going to annotated all the datasets in Official Gene Symbol (*finalID="GeneSymbol"*):

```
> newObjectMA <- allSameID(newObjectMA, finalID="GeneSymbol", organism = "Homo sapiens")

  |
  |                                                                      |   0%
  |
  |=================                                                     |  25%
  |
  |===================================                                   |  50%
  |
  |====================================================                  |  75%
  |
  |======================================================================| 100%
```

```
> rownames(newObjectMA$Study1$mExpres)[1:20]
```

```
 [1] "AAA4"     "AAAS"     "AABT"     "AACS"     "AACSP1"   "AADAC"   
 [7] "AADACL2"  "AADACL3"  "AADACL4"  "AADAT"    "AAGAB"    "AAK1"    
[13] "AAMP"     "AANAT"    "AARS1"    "AARS2"    "AARSD1"   "AASDH"   
[19] "AASDHPPT" "AASS"    
```

```
> rownames(newObjectMA$Study2$mExpres)[1:20]
```

```
 [1] "AAA4"     "AAAS"     "AABT"     "AACS"     "AACSP1"   "AADAC"   
 [7] "AADACL2"  "AADACL3"  "AADACL4"  "AADAT"    "AAGAB"    "AAK1"    
[13] "AAMP"     "AANAT"    "AARS1"    "AARS2"    "AARSD1"   "AASDH"   
[19] "AASDHPPT" "AASS"    
```

```
> rownames(newObjectMA$Study3$mExpres)[1:20]
```

```
 [1] "AARS1"   "AATF"    "ABCC2"   "ABCD1P4" "ABCD1P3" "ABCD1P2" "ACAD8"  
 [8] "AAVS1"   "ACAD9"   "ABT1"    "VSX1"    "ABHD5"   "AADAT"   "ABI3"   
[15] "VRK3"    "VPS54"   "ABAT"    "VSIG10"  "VRTN"    "VPS53"  
```

```
> rownames(newObjectMA$Study4$mExpres)[1:20]
```

```
 [1] "AARS1"   "AATF"    "ABCC2"   "ABCD1P4" "ABCD1P3" "ABCD1P2" "ACAD8"  
 [8] "AAVS1"   "ACAD9"   "ABT1"    "VSX1"    "ABHD5"   "AADAT"   "ABI3"   
[15] "VRK3"    "VPS54"   "ABAT"    "VSIG10"  "VRTN"    "VPS53"  
```

As it can be seen, all the studies are now annotated in Official Gene Symbol.

3.1.2 Logarithm transformation

To avoid problems with the returned fold-change by the meta-analysis, log_2 should be applied to the gene expression values. We can make use of *dataLog()* function to check if each dataset expression values have the log_2 applied already. If not, the function will make the transformation:

```
> newObjectMA <- dataLog(newObjectMA)
> head(newObjectMA[[1]][[1]])
```

```
        Sample1  Sample2  Sample3   Sample4
AAA4   5.439524 6.253319 2.926444 4.4304023
AAAS   5.769823 5.971453 1.831349 4.0466288
AABT   7.558708 5.957130 2.365252 3.4352889
AACS   6.070508 7.368602 2.971158 3.7151784
AACSP1 6.129288 5.774229 3.670696 3.9171749
AADAC  7.715065 7.516471 1.349453 0.3390772
```

3.1.3 Heterogeneity study

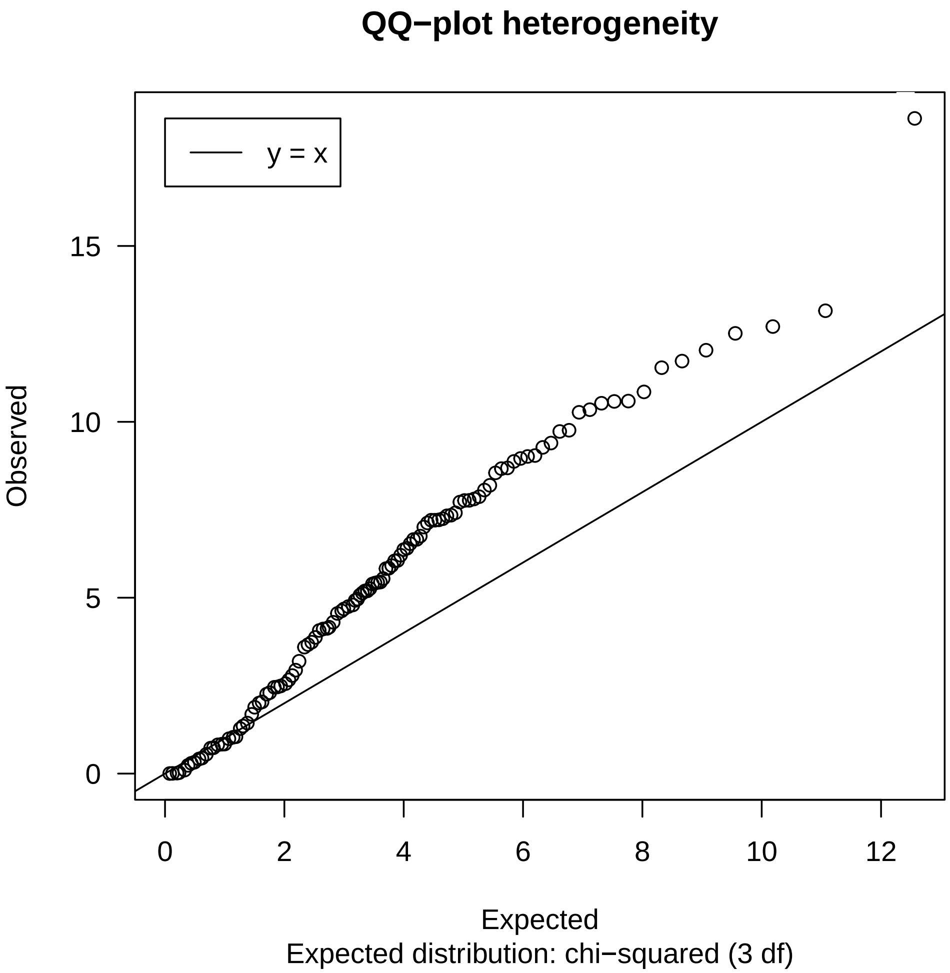
Some heterogeneity between studies may lead to some methods, such as the Fixed Effects Model, not providing reliable results. Therefore, it is advisable to carry out a study of heterogeneity to correctly choose the meta-analysis method [1]. The *heterogeneityTest()* function shows two ways of measuring heterogeneity.

On the one hand, it returns a QQ-plot of the Cochran's test [4]. In this plot, if most of the values are close to the central line, that is, most of the Cochran's test values are close to the expected distribution (chi-squared distribution), it can be said that there is homogeneity. In the case that these values deviate greatly from the expected distribution, it must be assumed that there is heterogeneity.

On the other hand, I^2 measures the percentage of variation across studies due to heterogeneity [5]. In the case of gene expression data, an I^2 for each gene across datasets would have to be calculated. As in the case of many genes, it can be difficult to observe all the I^2 values obtained, the *heterogeneityTest()* function returns the quantiles of the different I^2 values calculated. I^2 values equal to 0 indicate homogeneity and values less than 0.25 are usually categorized as low heterogeneity [5]. Therefore, to assume homogeneity in the gene expression meta-analysis, almost all I_2 values must be 0 or at least less than 0.25.

In the example shown below, it is observed that in the QQ-plot of the Cochran's test, Q-values deviate considerably from the expected distribution and approximately 10% of the I^2 values are greater than 0.25, therefore homogeneity could not be assumed.

```
> heterogeneityTest(newObjectMA)

[1] "I^2 Quantiles"
       0%       25%       50%       75%
0.0000000 0.0000000 0.1752591 0.5363021
```



3.2 Imputing missing genes expression values (optional)

Optionally, the *missGenesImput()* function allows to impute the expression values of the unmeasured genes. To perform this imputation, the function apply the knn method in the space of samples (*sampleKnn*) [2]. Once the function have been applied, it returns the same *objectMA* with all the datasets imputed and some imputation indicators:

```
> nrow(newObjectMA$Study1[[1]])

[1] 144

> nrow(newObjectMA$Study2[[1]])

[1] 144

> imputation <- missGenesImput(maObject, k =7)

[1] "Number of values imputed 310 (0.089 %)"
[1] "Number of genes imputed in Study1: 31 of 175 (17.7%)"
[1] "Number of genes imputed in Study2: 31 of 175 (17.7%)"
```

```
[1] "Number of genes imputed in Study3: 0 of 175 (0%)"
[1] "Number of genes imputed in Study4: 0 of 175 (0%)"

> maObject_imput <- imputation$objectMA
```

If we observe the number of genes in Study1, once the imputation is performed, we can observe that it goes from having 144 genes to 175.

```
> nrow(maObject_imput$Study1[[1]])

[1] 175

> nrow(maObject_imput$Study2[[1]])

[1] 175
```

In addition, the function returns some indicators to evaluate the reliability of the imputation:

- imputValuesSample: Number of missing values imputed per sample
- imputPercentageSample: Percentage of missing values imputed per sample
- imputValuesGene: Number of missing values imputed per gene
- imputPercentageGene: Percentage of missing values imputed per gen

```
> head(imputation$imputIndicators$imputValuesSample)

Sample1 Sample2 Sample3 Sample4 Sample5 Sample6
     31      31      31      31      31      31

> head(imputation$imputIndicators$imputPercentageSample)

 Sample1  Sample2  Sample3  Sample4  Sample5  Sample6
17.71429 17.71429 17.71429 17.71429 17.71429 17.71429

> head(imputation$imputIndicators$imputValuesGene)

  AAA4   AAAS   AABT   AACS AACSP1  AADAC
     0      0      0      0      0      0

> head(imputation$imputIndicators$imputPercentageGene)

  AAA4   AAAS   AABT   AACS AACSP1  AADAC
     0      0      0      0      0      0
```

3.3 Performing meta-analysis: *metaAnalysisDE()*

The *metaAnalysisDE()* function allows to perform a meta-analysis in only one step, needing only the meta-analysis object created previously.

This function has as input:

- objectMA: The meta-analysis object of DExMA package. The result obtained by *createObjectMA* function should be used.
- effectS: A list of three elements. The first element is a dataframe with genes in rows and studies in columns. Each component of the dataframe is the effect of a gene in a study. The second element of the list is also a dataframe with the same structure, but in this case each component of the dataframe represent the

variance of the effect of a gene in a study. The result of the calculateES function can be used too. The third element of the list is also a dataframe with the same structure, but in this case each component of the dataframe represent the log fold change of a gene in a study. This argument should be only used in the case that objectMA argument is null.

- pvalues: A list of two elements. The first element is a dataframe with genes in rows and studies in columns. Each component of the dataframe is the p-value of a gene in a study. The second element of the list is also a dataframe with the same structure, but in this case each component of the dataframe represent the log fold change of a gene in a study. This argument should be only used in the case that objectMA argument is null.
- weight: A vector of the weights of each dataset. This argument should only be included in case objectMA is null and you want to use "Stouffer" or "ACAT" method.
- typeMethod: a character that indicates the method to be performed:
  - "FEM": Fixed Effects model.
  - "REM": Random Effects model.
  - "Fisher": Fisher's method (sum of logarithms of p-values)
  - "Stouffer": Stouffer's method (sum of z-scores)
  - "maxP": Wilkinson's method (maximun of p-values)
  - "minP": Tippet's method (minimun of p-values)
  - "ACAT": Aggregated Cauchy Association Test method
- missAllow: a number between 0 and 1 that indicates the maximum proportion of missing values allows in a sample. If the sample has more proportion of missing values, the sample will be eliminated. In the other case, the missing values will be imputed by using the K-NN algorithm included in *impute* package [6]. In case the *objectMA* has been previously imputed, this element is not necessary.
- proportionData: a number between 0 and 1 that indicates the minimum proportion of datasets in which a gene must be contained to be included.In case the *objectMA* has been previously imputed, this element is not necessary.
- Adjusted p-value from which a gene is considered significant. Default 0.05. This value only takes you into account in the results generated in rank combination methods.

In the following example, we have applied a Random Effect model to the *DExMA* object ("newObjectMA") we have been working with so far. In addition we have allowed a 0.3 proportion of missing values in a sample and a gene must have been contained in at least the 50% of studies.

```
> resultsMA <- metaAnalysisDE(newObjectMA, typeMethod="REM",
+                              missAllow=0.3, proportionData=0.50)

[1] "Performing Random Effects Model"
```

The output of this function is a dataframe with the results of the meta-analysis where rows are the genes and columns are the different variables provided by the meta-analysis:

```
> head(resultsMA)

         Com.ES    ES.var      Qval      Qpval      tau2     Zval         Pval
AAA4   2.359381 0.7981372 6.0473427 0.10932948 1.5664238 2.640944 8.267527e-03
AAAS   2.281224 0.3370171 0.8210328 0.84442992 0.0000000 3.929542 8.510791e-05
AABT   2.493595 0.8816888 6.2077523 0.10192867 1.7585296 2.655635 7.915935e-03
AACS   2.313949 0.5884358 4.6107227 0.20262416 0.8221804 3.016506 2.557062e-03
AACSP1 3.772840 0.5852004 2.0056235 0.57124034 0.0000000 4.931921 8.142482e-07
AADAC  2.697529 1.0092867 6.5349523 0.08829422 2.0628748 2.685090 7.251027e-03
                FDR    AveFC propDataset
AAA4   0.0401893697 2.751718           1
AAAS   0.0018617356 3.116787           1
AABT   0.0401893697 2.705684           1
AACS   0.0178994351 2.928612           1
AACSP1 0.0001424934 3.021529           1
AADAC  0.0396540550 2.927683           1
```

Moreover, this function can also be applied to the imputed *objectMA*, but in this case, it is not necessary to indicate the *missAllow* and *proportionData* elements:

```
> resultsMA_imput <- metaAnalysisDE(maObject_imput, typeMethod="REM",)

[1] "Performing Random Effects Model"

> head(resultsMA_imput)

         Com.ES    ES.var      Qval      Qpval      tau2     Zval         Pval
AAA4   2.359381 0.7981372 6.0473427 0.10932948 1.5664238 2.640944 8.267527e-03
AAAS   2.281224 0.3370171 0.8210328 0.84442992 0.0000000 3.929542 8.510791e-05
AABT   2.493595 0.8816888 6.2077523 0.10192867 1.7585296 2.655635 7.915935e-03
AACS   2.313949 0.5884358 4.6107227 0.20262416 0.8221804 3.016506 2.557062e-03
AACSP1 3.772840 0.5852004 2.0056235 0.57124034 0.0000000 4.931921 8.142482e-07
AADAC  2.697529 1.0092867 6.5349523 0.08829422 2.0628748 2.685090 7.251027e-03
                FDR    AveFC propDataset
AAA4   0.0498902520 2.751718           1
AAAS   0.0021276978 3.116787           1
AABT   0.0494745922 2.705684           1
AACS   0.0279678674 2.928612           1
AACSP1 0.0001424934 3.021529           1
AADAC  0.0469973985 2.927683           1
```

The variables of the dataframe change from one type of meta-analysis to another. A more detailed explanation of these results will be addressed in the following sections.

3.3.1 Effects size combination results

The "FEM" and "REM" methods provide a dataframe with the variables:

- Com.ES: combined effect of the gene.
- ES.var: variance of the combined effect of the gene.
- Qval: total variance of the gene.
- Qpval:p-value for the total variance of the gene.

- tau2: between-study variance of the gene.
- zval: combined effect value for a standard normal. I can be use in order to find out if the gene is overexpressed (positive value) or underexpressed (negative value).
- Pval: P-value of the meta-analysis for the gene.
- FDR: P-value adjusted of the meta-analysis for the gene.
- AveFC: Weighted average of log Fold-Change values for the gene used in order to find out if the gene is overexpressed (positive value) or underexpressed (negative value).
- Prop.dataset: Proportion of the datasets in which the gene is included.

```
> resultsES <- metaAnalysisDE(newObjectMA, typeMethod="REM", proportionData=0.5)

[1] "Performing Random Effects Model"

> head(resultsES)

         Com.ES    ES.var      Qval      Qpval      tau2     Zval         Pval
AAA4   2.359381 0.7981372 6.0473427 0.10932948 1.5664238 2.640944 8.267527e-03
AAAS   2.281224 0.3370171 0.8210328 0.84442992 0.0000000 3.929542 8.510791e-05
AABT   2.493595 0.8816888 6.2077523 0.10192867 1.7585296 2.655635 7.915935e-03
AACS   2.313949 0.5884358 4.6107227 0.20262416 0.8221804 3.016506 2.557062e-03
AACSP1 3.772840 0.5852004 2.0056235 0.57124034 0.0000000 4.931921 8.142482e-07
AADAC  2.697529 1.0092867 6.5349523 0.08829422 2.0628748 2.685090 7.251027e-03
                FDR    AveFC propDataset
AAA4   0.0401893697 2.751718           1
AAAS   0.0018617356 3.116787           1
AABT   0.0401893697 2.705684           1
AACS   0.0178994351 2.928612           1
AACSP1 0.0001424934 3.021529           1
AADAC  0.0396540550 2.927683           1
```

3.3.2 P-value combination results

The "Fisher", "Stouffer", "minP", "maxP" and "ACAT" methods provide a dataframe with the following variables:

- Stat: Statistical calculated in the method
- Pval: P-value of the meta-analysis for the gene.
- FDR: P-value adjusted of the meta-analysis for the gene.
- AveFC: Average of log Fold-Change values for the gene used in order to find out if the gene is overexpressed (positive value) or underexpressed (negative value).
- Prop.dataset: Proportion of the datasets in which the gene is included.

Here we present an example making use of "maxP" method:

```
> resultsPV <- metaAnalysisDE(newObjectMA, typeMethod="maxP", proportionData=0.5)

[1] "Performing MaxP's method"

> head(resultsPV)
```

```
             Stat         Pval          FDR    AveFC propDataset
AAA4   0.13754548 3.579194e-04 3.296626e-03 2.831428           1
AAAS   0.07824187 3.747632e-05 6.811215e-04 3.180932           1
AABT   0.27659986 5.853395e-03 2.768497e-02 2.935515           1
AACS   0.11561455 1.786693e-04 1.737063e-03 3.060848           1
AACSP1 0.01862270 1.202736e-07 1.052394e-05 2.953126           1
AADAC  0.22367842 2.503204e-03 1.510554e-02 3.629187           1
```

3.4 Visualization of the results: heatmap

Finally, we can represent in a heatmap the significant genes in order to observe how they are expressed in each of the studies. In *makeHeatmap()* function we have to include both the object that has been used in the meta-analysis, the result of it and the applied method. In addition, this package offers three different scaling approaches (*scaling*) in order to compare properly the gene expression of the studies in the heatmap:

- "zscor": It calculates a z-score value for each gene, that is, the mean gene expression from each gene is subtracted from each gene expression value and then it is divided by the standard deviation.
- "swr": Scaling relative to reference dataset approach [7].
- "rscale": It uses the rescale function of the *scales* package to scale the gene expresion [8].
- "none": no scaling approach is applied.

Moreover, in *regulation* argument, we can choose if we want to represent the overexpressed or underexpressed genes:

- "up": only up-expressed genes are represented.
- "down: only down-expressed genes are represented
- "all": up-expressed and down-expressed genes are represented.

We can choose the number of significant genes (*numSig*) that we want to be shown on the graph and the adjusted p-value from which a gene is considered as significant (*fdrSig*). In addition, the genes that are not presented in one sample are represented in gray.

Here we present an example of the heatmap which have been obtained from the result of applying a random effects model to the object "*newObjectMA*" and making use of a "zscor" scaling approach.

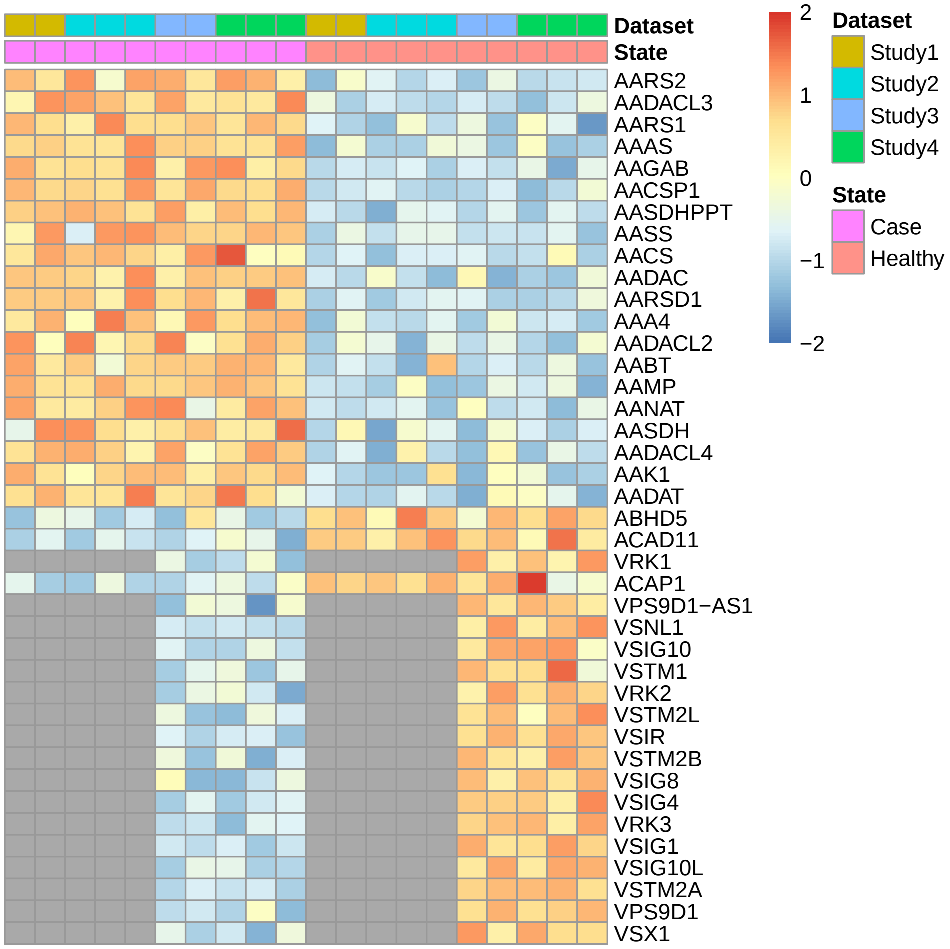
```
> makeHeatmap(objectMA=newObjectMA, resMA=resultsMA, scaling = "zscor",
+             regulation = "all", numSig=40,
+             fdrSig = 0.05, logFCSig = 1.5, show_rownames = TRUE)

[1] "scaling using z-score..."
           Sample1.Study1 Sample2.Study1 Sample3.Study1 Sample4.Study1
AARS2           0.9682999     0.50830923     -1.3357281     -0.1408811
AADACL3         0.1962999     1.27309503     -0.3733564     -1.0960385
AARS1           0.9972642     0.69944418     -0.6523539     -1.0443545
AAAS            0.7106395     0.81561074     -1.3397753     -0.1864749
AAGAB           1.0769772     0.61814883     -0.9507280     -0.7443980
```

```
AACSP1          0.9985590     0.71637534     -0.9554120     -0.7595223
AASDHPPT        0.8071383     0.91551061     -0.7459808     -0.9766682
AASS            0.2068204     1.27069616     -1.0930385     -0.3844781
AACS            0.5086833     1.14412657     -1.0085114     -0.6442985
AADAC           0.8863152     0.83580877     -0.7325829     -0.9895411
AARSD1          0.8645997     0.83776414     -1.0765674     -0.6257964
AAA4            0.4724367     1.04024800     -1.2810233     -0.2316614
AADACL2         1.3011499     0.02106791     -1.1261009     -0.1961170
AABT            1.1556595     0.47758562     -1.0431379     -0.5901072
AAMP            1.1004181     0.59274116     -0.7999765     -0.8931828
AANAT           1.1659982     0.49625758     -0.7516622     -0.9105936
AASDH          -0.4675273     1.32468488     -1.0027167      0.1455591
AADACL4         0.5993446     1.07065287     -1.0586534     -0.6113440
AAK1            1.0835176     0.59586852     -0.6653927     -1.0139935
AADAT           0.6561996     1.04082847     -0.6966487     -1.0003793
ABHD5          -1.2412829    -0.36578028      0.6729645      0.9340987
ACAD11         -1.1108356    -0.58004971      0.8409657      0.8499196
VRK1                   NA             NA             NA             NA
ACAP1          -0.5392450    -1.13710621      0.9127669      0.7635843
VPS9D1-AS1             NA             NA             NA             NA
VSNL1                  NA             NA             NA             NA
VSIG10                 NA             NA             NA             NA
VSTM1                  NA             NA             NA             NA
VRK2                   NA             NA             NA             NA
VSTM2L                 NA             NA             NA             NA
VSIR                   NA             NA             NA             NA
VSTM2B                 NA             NA             NA             NA
VSIG8                  NA             NA             NA             NA
VSIG4                  NA             NA             NA             NA
VRK3                   NA             NA             NA             NA
VSIG1                  NA             NA             NA             NA
VSIG10L                NA             NA             NA             NA
VSTM2A                 NA             NA             NA             NA
VPS9D1                 NA             NA             NA             NA
VSX1                   NA             NA             NA             NA
           Sample5.Study2 Sample6.Study2 Sample7.Study2 Sample8.Study2
AARS2          1.29003246     -0.1493767      1.1881662     -0.6257615
AADACL3        1.18101154      0.9235015      0.5709817     -0.7336264
AARS1          0.32261014      1.3600903      0.7049359     -1.2920142
AAAS           0.59178756      0.5776332      1.3366176     -1.1031207
AAGAB          0.66979718      0.5913302      1.3541761     -0.8630518
AACSP1         0.75769816      0.6575860      1.2379418     -0.6163623
AASDHPPT       1.03567093      0.9364763      0.5862831     -1.4423486
AASS          -0.69488947      1.2593696      1.2923725     -0.8916272
AACS           0.87262982      1.0046754      0.7826212     -1.3015653
AADAC          0.75691392      0.2637028      1.3332952     -0.1318817
AARSD1         0.88971101      0.2844332      1.3444491     -1.1891260
AAA4           0.05717809      1.4476910      0.9225027     -0.8864413
AADACL2        1.40453080      0.2067201      0.7295036     -0.4744804
AABT           0.85037008     -0.2263859      0.7607782     -0.8746852
AAMP           0.61483783      1.0780356      0.7377432     -1.1325895
```

```
AANAT          0.42906396      0.8208962      1.3123105     -0.7528130
AASDH          1.28122621      0.6882994      0.3313130     -1.5532911
AADACL4        1.09122261      0.8224126      0.2253374     -1.4373346
AAK1           0.03726305      0.7816216      0.9843186     -1.2179787
AADAT          0.55307845      0.5457234      1.4479847     -1.0401099
ABHD5         -0.48331045     -1.1912463     -0.7252009      0.1050830
ACAD11        -1.16714015     -0.5365638     -0.8363435      0.3055333
VRK1                   NA             NA             NA             NA
ACAP1         -1.17329107     -0.3798016     -1.0565983      0.8999814
VPS9D1-AS1             NA             NA             NA             NA
VSNL1                  NA             NA             NA             NA
VSIG10                 NA             NA             NA             NA
VSTM1                  NA             NA             NA             NA
VRK2                   NA             NA             NA             NA
VSTM2L                 NA             NA             NA             NA
VSIR                   NA             NA             NA             NA
VSTM2B                 NA             NA             NA             NA
VSIG8                  NA             NA             NA             NA
VSIG4                  NA             NA             NA             NA
VRK3                   NA             NA             NA             NA
VSIG1                  NA             NA             NA             NA
VSIG10L                NA             NA             NA             NA
VSTM2A                 NA             NA             NA             NA
VPS9D1                 NA             NA             NA             NA
VSX1                   NA             NA             NA             NA
           Sample9.Study2 Sample10.Study2 Sample11.Study3 Sample12.Study3
AARS2         -1.02470491      -0.6783556      1.07650640      0.51153684
AADACL3       -0.93417370      -1.0076946      1.17910221      0.47201904
AARS1         -0.17388068      -0.9217414      0.70494679      0.90042094
AAAS          -1.10680832      -0.2961094      0.80558853      0.82527829
AAGAB         -0.66653569      -1.0857160      0.31553865      1.27223060
AACSP1        -0.96070349      -1.0761601      0.54696710      1.12154963
AASDHPPT      -0.51632041      -0.5997614      1.23228200      0.35363820
AASS          -0.48814487      -0.4770805      0.96104960      0.76457415
AACS          -0.67777542      -0.6805857      0.34483610      1.24541482
AADAC         -0.89449227      -1.3275379      0.31689617      0.92739195
AARSD1        -0.77010577      -0.5593615      0.67181245      1.01080164
AAA4          -0.96655055      -0.5743799      0.16657615      1.23892903
AADACL2       -1.42596732      -0.4403068      1.39997181     -0.03779453
AABT          -1.42799917       0.9179220      0.86569742      0.84936487
AAMP          -0.02038311      -1.2776440      0.72858636      0.89381798
AANAT         -0.57603553      -1.2334222      1.39020370     -0.46059046
AASDH         -0.17029375      -0.5772538      0.61292009      0.91008948
AADACL4        0.26967066      -0.9713088      1.16074438     -0.03447650
AAK1          -1.23178212       0.6465576      0.98948573      0.37121363
AADAT         -0.54991192      -0.9567647      0.55098394      0.77784274
ABHD5          1.44534631       0.8493283     -1.30616930      0.51088949
ACAD11         0.94328094       1.2912332     -1.04270733     -0.65971765
VRK1                   NA              NA     -0.39913284     -1.13280881
ACAP1          0.64129122       1.0684184     -1.05514675     -0.60626801
VPS9D1-AS1             NA              NA     -1.29470085     -0.22293598
```

```
VSNL1                 NA              NA     -0.73435713     -0.87721110
VSIG10                NA              NA     -0.59436451     -1.04617358
VSTM1                 NA              NA     -1.12416482     -0.53846020
VRK2                  NA              NA     -1.12083641     -0.39565683
VSTM2L                NA              NA     -0.35759710     -1.23743923
VSIR                  NA              NA     -0.63721534     -1.04789507
VSTM2B                NA              NA     -0.31304906     -1.25324398
VSIG8                 NA              NA      0.08029449     -1.38472445
VSIG4                 NA              NA     -1.12859941     -0.55504243
VRK3                  NA              NA     -0.91730441     -0.80874956
VSIG1                 NA              NA     -0.76532124     -0.92388309
VSIG10L               NA              NA     -1.14388088     -0.45363675
VSTM2A                NA              NA     -1.00843179     -0.70457483
VPS9D1                NA              NA     -0.94935574     -0.75514748
VSX1                  NA              NA     -0.47913046     -1.07220352
           Sample13.Study3 Sample14.Study3 Sample15.Study4 Sample16.Study4
AARS2         -1.193097649     -0.39494560       1.2047919      1.07007824
AADACL3       -0.716438370     -0.93468287       0.6082930      0.47422102
AARS1         -0.353412568     -1.25195516       0.5683881      0.99032956
AAAS          -0.403074596     -1.22779222       0.5868315      0.60634862
AAGAB         -0.690542772     -0.89722647       1.3253285      0.34768914
AACSP1        -0.993913293     -0.67460343       0.7438120      0.70095867
AASDHPPT      -1.015277040     -0.57064316       0.9699261      0.67584429
AASS          -0.901030836     -0.82459291       0.7651139      1.02484771
AACS          -0.614044000     -0.97620692       1.7246362      0.01073741
AADAC          0.173278628     -1.41756675       0.8166646      0.83547046
AARSD1        -0.605348342     -1.07726574       0.3208675      1.54083274
AAA4          -1.176875375     -0.22862980       0.6756384      0.98541331
AADACL2       -0.916553115     -0.44562416       0.6482716      1.08819763
AABT          -1.028452101     -0.68661020       1.0582122      1.01834320
AAMP          -1.232006994     -0.39039735       1.0541916      0.87742615
AANAT         -0.004861139     -0.92475211       0.4283357      1.18895237
AASDH         -1.325546900     -0.19746268       0.3936066      0.46554875
AADACL4       -1.276311403      0.15004352       0.6127288      1.15154039
AAK1          -1.372219271      0.01151991       0.8914460      0.71351002
AADAT         -1.441744415      0.11291774       1.4996941      0.68539206
ABHD5         -0.200974617      0.99625443      -0.4455769     -1.18859139
ACAD11         0.732030611      0.97039437      -0.1564368     -0.49161788
VRK1           1.204157191      0.32778445      -0.9103028     -0.22005511
ACAP1          0.567121587      1.09429317      -0.3501057     -0.91685493
VPS9D1-AS1     1.006243659      0.51139316      -0.3766663     -1.71358002
VSNL1          0.362701612      1.24886662      -0.7663241     -0.88793936
VSIG10         0.498889938      1.14164815      -1.0549086     -0.36552545
VSTM1          1.010411751      0.65221327      -0.3253520     -1.22683282
VRK2           0.290643408      1.22584984      -0.2475666     -0.75203655
VSTM2L         0.612048448      0.98298788      -1.3242282     -0.30928837
VSIR           0.647524480      1.03758593      -0.7307610     -0.66790259
VSTM2B         1.011994595      0.55429844      -0.2874520     -1.45039745
VSIG8          0.987865267      0.31656469      -1.3685133     -0.83409512
VSIG4          0.862633571      0.82100827      -1.1848497     -0.76428048
VRK3           0.776871260      0.94918271      -1.3310580     -0.56329900
```

```
VSIG1          1.103462931      0.58574140      -0.6755072     -1.18598708
VSIG10L        0.474938718      1.12257892      -0.4928980     -1.07710114
VSTM2A         0.757903621      0.95510300      -0.8669201     -0.72768843
VPS9D1         0.657559123      1.04694410      -1.0661534     -0.04791842
VSX1           1.232555953      0.31877803      -0.7673579     -1.42403271
           Sample17.Study4 Sample18.Study4 Sample19.Study4 Sample20.Study4
AARS2           0.33548921    -0.978168251      -0.8478691     -0.78432197
AADACL3         1.38821537    -1.296333970      -0.8194346     -0.35496081
AARS1           0.74282169    -0.047906853      -0.5778748     -1.67575772
AAAS            1.21907318    -0.053486390      -1.2717939     -1.08697301
AAGAB           0.71612384    -0.436915497      -1.4827170     -0.46950899
AACSP1          1.08356763    -1.350536645      -0.9520291     -0.22577249
AASDHPPT        1.01563000    -1.194586455      -0.5487922     -0.91802170
AASS            0.87945786    -0.847080284      -0.5849513     -1.23738786
AACS            0.12387329    -0.900151433       0.1287143     -1.08780978
AADAC           0.93597171    -1.091781144      -1.2211976     -0.27512806
AARSD1          0.52121887    -1.087312823      -0.9866706     -0.30893567
AAA4            1.02608096    -0.790252699      -0.7292882     -1.16759179
AADACL2         0.79068870    -1.024505524      -1.2952874     -0.20736496
AABT            0.48724030    -0.977729098      -0.3503127     -1.23575392
AAMP            0.61802876    -0.782743132      -0.3527996     -1.41410376
AANAT           0.92838867    -0.766724415      -1.3200648     -0.45888756
AASDH           1.58695523    -0.670733809      -1.0926837     -0.68269299
AADACL4         0.84468751    -1.244625253      -0.4527872     -0.91154424
AAK1            0.96135491    -0.243104099      -1.2461008     -1.07710606
AADAT          -0.24171822    -0.035805706      -0.5054236     -1.40213857
ABHD5          -0.97877645     0.681061366       1.1904156      0.74146782
ACAD11         -1.46496299     0.124966311       1.5398884      0.44816300
VRK1           -1.27953065     0.944625717       0.2227733      1.24248951
ACAP1          -0.06453924     1.948688579      -0.4582433     -0.15894545
VPS9D1-AS1     -0.17021280     1.012295278       0.8351452      0.41301863
VSNL1          -0.96501288     0.388287090       0.9517297      1.27925953
VSIG10         -0.90635013     1.163428345       1.2531779     -0.08982205
VSTM1          -0.47355971     0.701930653       1.6094881     -0.28567422
VRK2           -1.48970178     0.651594331       1.0616402      0.77607049
VSTM2L         -0.66769822     0.004892528       0.9632726      1.33304965
VSIR           -1.25441145     0.664738911       1.1119140      0.87642214
VSTM2B         -0.68584179     0.331013623       1.1948646      0.89781307
VSIG8          -0.35261052     0.941047309       0.5633877      1.05078393
VSIG4          -0.59129239     0.830268073       0.3497454      1.36040909
VRK3           -0.64249280     0.994620857       0.3713171      1.17091190
VSIG1          -0.79024789     0.686231980       1.2173862      0.74812392
VSIG10L        -1.02829017     0.429639636       1.1256359      1.04301371
VSTM2A         -1.09845817     0.971509565       1.0688633      0.65269384
VPS9D1         -1.34019192     0.658610706       0.8004777      0.99517526
VSX1           -0.33785216     1.144663968       0.6788051      0.70577373
```



4 Additional information

DExMA provides some functions which may be useful for the user, although they are not essential to perform meta-analysis.

4.1 GEO microarray data download

In addition to using own user data, **DExMA** package allows to make use of public microarray data from the NCBI GEO public database [3]. For doing that, we can make use of *downloadGEOData()* function. This function uses internally *GEOquey* package in order to download some files at the same time. This function has an input a character vector (*GEOobject*) with the GEO ID of the different datasets that we want to download and a character (*directory*) that indicates the directory where GSE Series Matrix files [9] are going to be stored.

```
> GEOobjects<- c("GSE4588", "GSE10325")
> dataGEO<-downloadGEOData(GEOobjects)
```

Once the download process is completed, we get a list of ExpressionSets. This list can be used as input of *createObjectMA()* function, although it is advisable to homogenize gene annotation, in case genes IDs are not Entrez, Official Gene Symbol or Ensembl.

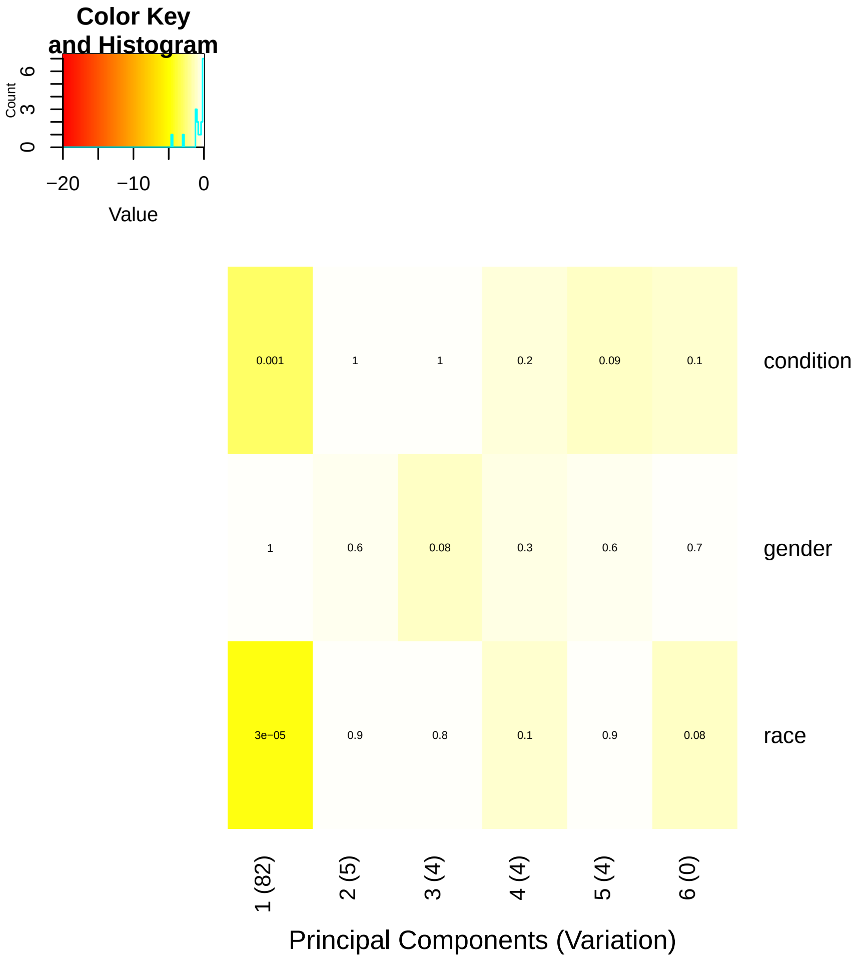
4.2 Using RNA-Seq data

DExMA internally uses *limma* package in order to assess differential expression. Therefore, RNA-Seq data must be previously normalized by the user in order to be able to include correctly theses data in the gene-expression meta-analysis. Since *limma* is used internally, we recommend to apply the steps described in the limma user's guide for the RNA-Seq data normalization [10], although the users can use the type of normalization they prefer.

4.3 Removing Batch Effects

Before the creation of the *objectMA*, a batch effect correction can be applied in order to reduce the effect of covariates that may be affecting to gene expression [1]. Firstly, with functions *prince()* and *prince.plot()* function of the *swamp* package [11], we can obtain a visualization of the p-values of each principal component associated with the categorical covariates

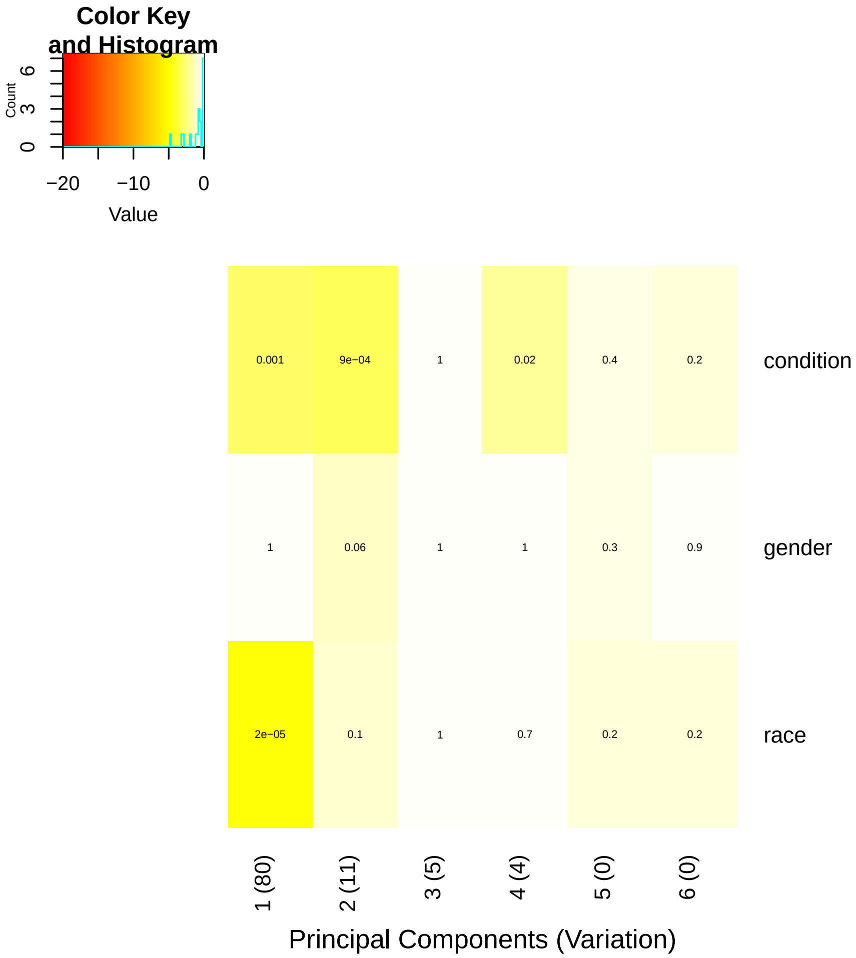
```
> library(swamp)
> pheno <- listPhenodatas$Study2
> pheno <- pheno[,apply(pheno,2,function(x) length(table(x)))>1]
> # Character variables must be converted in numeric
> pheno <- data.frame(apply(pheno, 2, factor), stringsAsFactors = TRUE)
> res_prince <- prince(listMatrixEX$Study2, pheno,top=ncol(listMatrixEX$Study2))
> prince.plot(res_prince,note=TRUE , notecex=0.5)
```



The categorical variables that may be causing a batch effect can be corrected by using the *removeBatch()* function. The input of this function is the expression matrix and the phenodata. In addition , We also have to add a formula with the variables for which we want to correct the gene expression and the name of the variable that contains the cases and controls groups. Finally, if there is a covariate inside the formula that we want to give greater importance, we will have the option to indicate in the function (*mainCov()*). Here we show an example in which we have corrected the gene expression of the previous study by two of their covariates:

```
> listMatrixEX$Study2 <- batchRemove(listMatrixEX$Study2, listPhenodatas$Study2,
+                                     formula=~gender+race,
+                                     mainCov = "race", nameGroup="condition")

Coefficients not estimable: batch1 batch2 (Intercept) raceC raceH

> head(listMatrixEX$Study2)

           Sample5  Sample6  Sample7  Sample8  Sample9 Sample10
100859927 4.367690 6.648252 5.143704 2.820073 2.688687 2.688687
8086      5.937004 5.914434 5.831974 3.234439 3.228559 3.228559
8212      5.294553 4.092231 2.575022 3.368330 2.750491 2.750491
65985     5.685822 5.900363 5.544146 2.153294 3.166798 3.166798
729522    5.733054 5.553966 6.798683 3.275042 2.659061 2.659061
```

```
13         6.153159 5.435909 7.621115 4.860632 3.751608 3.751608
```



4.4 Calculating Effects size

The *calculateES()* function returns the effects size in each of the studies. Moreover, it calculates the variance of each of the effects and the log_2 Fold Change. The effects size are calculated by making use of the *Hedges' g estimator* [1].

```
> effects <- calculateES(newObjectMA)
> head(effects$ES)

          Study1    Study2    Study3   Study4
AAA4   1.448942 2.5008385 1.1221262 6.642233
AAAS   1.506204 2.9643045 2.2593392 2.430575
AABT   2.288894 0.7656633 5.7273196 3.478798
AACS   2.579019 5.3315818 1.8723418 1.207949
AACSP1 5.705553 5.1083967 2.9008711 3.138604
AADAC  7.515582 2.1970319 0.8345501 3.776154

> head(effects$Var)
```

```
         Study1   Study2   Study3    Study4
AAA4   1.262429 1.187849 1.157396 4.3432711
AAAS   1.283581 1.398925 1.638077 1.1589745
AABT   1.654879 0.715520 5.100274 1.6751698
AACS   1.831418 3.035480 1.438208 0.7882618
AACSP1 5.069167 2.841310 2.051882 1.4875697
AADAC  8.060497 1.068912 1.087059 1.8549449

> head(effects$logFC)

         Study1   Study2   Study3   Study4
AAA4   2.167998 2.654066 3.118286 3.385361
AAAS   2.931649 2.663952 4.321073 2.807056
AABT   3.857649 1.030831 3.259410 3.594168
AACS   3.376387 2.881147 4.252159 1.733698
AACSP1 2.157823 3.164180 3.998437 2.492064
AADAC  6.771502 2.282112 1.747278 3.715857
```

4.5 Calculating Individual P-values

Similar to the calculation of effects sizes, the individual p-values of each of the studies and the log_2 fold change of each one can also be calculated by applying *pvalueIndAnalysis()*. P-value are obtained by assessing differential expression with *limma* package.

```
> pvalues <- pvalueIndAnalysis(newObjectMA)
> head(pvalues$p)

            Study1       Study2      Study3       Study4
AAA4   0.084586213 0.0124328637 0.137545477 0.0002392007
AAAS   0.078241874 0.0065691298 0.032924671 0.0139623839
AABT   0.031889060 0.2765998575 0.003696266 0.0035646909
AACS   0.024335148 0.0005986238 0.049748561 0.1156145550
AACSP1 0.003718102 0.0007176307 0.018622696 0.0053471186
AADAC  0.001900842 0.0197199607 0.223678418 0.0025641765

> head(pvalues$logFC)

         Study1   Study2   Study3   Study4
AAA4   2.167998 2.654066 3.118286 3.385361
AAAS   2.931649 2.663952 4.321073 2.807056
AABT   3.857649 1.030831 3.259410 3.594168
AACS   3.376387 2.881147 4.252159 1.733698
AACSP1 2.157823 3.164180 3.998437 2.492064
AADAC  6.771502 2.282112 1.747278 3.715857
```

5 Session info

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.3 LTS
```

```
Matrix products: default
BLAS:   /home/biocbuild/bbs-3.18-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] swamp_1.5.1         MASS_7.3-60         gplots_3.1.3
[4] amap_0.8-19         impute_1.76.0       Biobase_2.62.0
[7] BiocGenerics_0.48.1 DExMA_1.10.4        DExMAdata_1.10.0

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.0        farver_2.1.1            dplyr_1.1.3
 [4] blob_1.2.4              Biostrings_2.70.1       bitops_1.0-7
 [7] fastmap_1.1.1           RCurl_1.98-1.13         GEOquery_2.70.0
[10] XML_3.99-0.15           digest_0.6.33           lifecycle_1.0.4
[13] bnstruct_1.0.14         survival_3.5-7          statmod_1.5.0
[16] KEGGREST_1.42.0         RSQLite_2.3.3           magrittr_2.0.3
[19] genefilter_1.84.0       compiler_4.3.2          rlang_1.1.2
[22] tools_4.3.2             utf8_1.2.4              igraph_1.5.1
[25] yaml_2.3.7              data.table_1.14.8       knitr_1.45
[28] bit_4.0.5               plyr_1.8.9              xml2_1.3.5
[31] RColorBrewer_1.1-3      KernSmooth_2.23-22      BiocParallel_1.36.0
[34] purrr_1.0.2             grid_4.3.2              stats4_4.3.2
[37] fansi_1.0.5             caTools_1.18.2          xtable_1.8-4
[40] colorspace_2.1-0        edgeR_4.0.1             scales_1.2.1
[43] gtools_3.9.4            cli_3.6.1               rmarkdown_2.25
[46] crayon_1.5.2            generics_0.1.3          tzdb_0.4.0
[49] httr_1.4.7              DBI_1.1.3               cachem_1.0.8
[52] zlibbioc_1.48.0         splines_4.3.2           parallel_4.3.2
[55] AnnotationDbi_1.64.1    BiocManager_1.30.22     XVector_0.42.0
[58] matrixStats_1.1.0       vctrs_0.6.4             Matrix_1.6-2
[61] sva_3.50.0              hms_1.1.3               IRanges_2.36.0
[64] S4Vectors_0.40.1        bit64_4.0.5             locfit_1.5-9.8
[67] limma_3.58.1            tidyr_1.3.0             snpStats_1.52.0
[70] annotate_1.80.0         glue_1.6.2              codetools_0.2-19
[73] gtable_0.3.4            GenomeInfoDb_1.38.1     munsell_0.5.0
[76] tibble_3.2.1            pillar_1.9.0            htmltools_0.5.7
[79] GenomeInfoDbData_1.2.11 R6_2.5.1                evaluate_0.23
```

```
[82] lattice_0.22-5          readr_2.1.4             png_0.1-8
[85] pheatmap_1.0.12         memoise_2.0.1           BiocStyle_2.30.0
[88] Rcpp_1.0.11             nlme_3.1-163            mgcv_1.9-0
[91] xfun_0.41               MatrixGenerics_1.14.0   pkgconfig_2.0.3
```

References

[1] Toro-Domínguez D., Villatoro-García J.A., Martorell-Marugán J., and et al. A survey of gene expression meta-analysis:methods and applications. *Briefings in Bioinformatics*, pages 1–12, 2020. doi:https://doi.org/10.1093/bib/bbaa019.

[2] Christopher A. Mancuso, Jacob L. Canfield, Deepak Singla, and Arjun Krishnan. A flexible, interpretable, and accurate approach for imputing the expression of unmeasured genes. *Nucleic Acids Research*, 48(21):e125–e125, 2020. URL: https://doi.org/10.1093/nar/gkaa881.

[3] Barret T., Wilhite S., Ledoux P., and et al. Ncbi geo: archive for functional genomics data sets–update. *Nucleic Acids Research*, pages 991–995, 2020. doi:https://doi.org/10.1093/nar/gks1193.

[4] Higgins J. and Thompson S. Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine*, pages 1539–1558, 2002. doi:10.1002/sim.1186.

[5] Higgins J., Thompson S., Deeks J., and Altman D. Measuring inconsistency in meta-analyses. *BMJ*, pages 557–560, 2003. doi:10.1136/bmj.327.7414.557.

[6] Hastie T., Tibshirani R., Narasimhan B., and Chu G. impute: Imputation for microarray data. 2019.

[7] Lazar C., Meganck S., Taminau J., and et al. Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in Bioinformatics*, pages 469–490, 2013. doi:10.1093/bib/bbs037.

[8] Wickham H. and Siedel D. Scale functions for visualization. 2020.

[9] Davis S. and Meltzer P.S. Geoquery: a bridge between the gene expression omnibus (geo) and bioconductor. *Bioinformatics*, pages 1846–1847, 2007. doi:0.1093/bioinformatics/btm254.

[10] Smyth G. K., Ritchie M., Thorne N., Yifang Hu, and et al. Linear models for microarray and rna-seq data user's guide. 2020.

[11] Lauss M. Visualization, analysis and adjustment of high-dimensional data in respect to sample annotations. 2019.