demi Package Vignette

Sten Ilmjärv `sten.ilmjarv@gmail.com`
Hendrik Luuk `hendrik.luuk@gmail.com`

February 13, 2015

Contents

1 Introduction

The R package demi is an implementation of the DEMI (Differential Expression from Multiple Indicators) framework into a self-contained software tool for the differential expression analysis of high-density gene expression microarrays.

The DEMI framework is a 3-step methodology for estimating differential expression directly from probe-level data without the summarization of probe signals. As output, DEMI delivers lists of differentially expressed genes, transcripts, genomic regions and pathways. We hope that as a software tool, DEMI facilitates the analysis of large-scale gene expression data by providing a single command to launch the analysis workflow starting from CEL files and ending with tabulated results. The software is intended for use by lifescientists and bioinformaticians alike.

High-density microarrays contain thousands to millions of covalently bound 25-mer (Affymetrix$^{®}$) nucleotides designed to measure the expression of specific regions on the genome by the nucleic acid hybridization principle. Based on sequence identity, a probe can be designed to target an exon, thereby reporting the expression of the exon, to target an exon-exon junction, thereby reporting the expression of a specific transcript, or to target intergenic regions that do not correspond to any of the known genes. Ultimately, choosing a set of targets for differential expression analysis is down to the user's choice. For example, one might be interested in evaluating differential expression only on exon level in order to determine differential splicing between two experimental conditions. Another user might prefer a more general approach being interested in studying up- and down-regulation on the level of transcripts, genes or, even more generally, on the level of pathways. In addition, one can study the expression of large genomic regions where the probes might also target intergenic sequences. All of this can be done with DEMI.

By using a common statistical test to classify probes based on their expression profiles into up-regulated, down-regulated or equally expressed clusters and by studying the enrichment of differentially expressed probes among on-target probes as opposed to off-target probes, DEMI arrives at a differential expression estimate which accounts for the number of probes matching a target. In other words, even if the ratio of differentially expressed on-target probes is identical for two targets, the target with a smaller number of matching probes gets a higher (i.e. less significant) p-value. This makes sense intuitively as a target which has been independently measured by 15 probes is expected to yield more reliable differential expression estimates than a target that was measured by only 3 probes. In fact, this peculiarity of the DEMI framework enables the user to study differential expression in very small samples of n < 3 while still maintaining a reasonable degree of accuracy.

2 Installation

To start off you need to install the demi package. You should be able to install the demi package from a CRAN repository with the function:

```
> install.packages( "demi" )
```

Now you can to load the DEMI package:

```
> library( demi )
```

When you run the DEMI analysis for the first time you are required to download and install the demi data packages as well. It will be done automatically but can take some time for some annotation packages contain several microarray platforms and are therefore quite big in size. Make sure you have enough room on your hard drive where R libraries will be installed. However this is done only once and future analysis will load the installed demi data packages from the local library. If for some reason the packages won't be installed automatically or demi repository is down, then you can manually download and install the data packages from the website `http://biit.cs.ut.ee/demi`. There you can also find the manual and other useful information.

For the demonstration of demi package we will use Microarray Quality Consortium (MAQC) [1] dataset on HG-U133 Plus 2 microarray platform to illustrate the DEMI usage with some examples. First you need to download the *CEL* files used in the examples to a destination folder.

```
# an example of a destination folder
> celdir <- "demitest/testdata/"

# download all 8 CEL files to the destination folder and rename them
# according to their feature
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247694/
suppl/GSM247694.CEL.gz", destfile=paste(celdir, "UHR01_GSM247694.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247695/
suppl/GSM247695.CEL.gz", destfile=paste(celdir, "UHR02_GSM247695.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247698/
suppl/GSM247698.CEL.gz", destfile=paste(celdir, "UHR03_GSM247698.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247699/
suppl/GSM247699.CEL.gz", destfile=paste(celdir, "UHR04_GSM247699.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247696/
suppl/GSM247696.CEL.gz", destfile=paste(celdir, "BRAIN01_GSM247696.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247697/
suppl/GSM247697.CEL.gz", destfile=paste(celdir, "BRAIN02_GSM247697.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247700/
suppl/GSM247700.CEL.gz", destfile=paste(celdir, "BRAIN03_GSM247700.CEL.gz",
sep = ""))
download.file("ftp://ftp.ncbi.nlm.nih.gov/geo/samples/GSM247nnn/GSM247701/
suppl/GSM247701.CEL.gz", destfile=paste(celdir, "BRAIN04_GSM247701.CEL.gz",
sep = ""))

# We need the gunzip function (located in the R.utils package) to unpack
# the downloaded gz files in the destination folder
# Also we will remove the original unpacked files for we won't need them
library( R.utils )
for( i in list.files( celdir ) ) {
    gunzip( paste( celdir, i, sep = "" ), remove = TRUE )
}
```

The data are raw expression measurements in *CEL* files used in a paper by Pradervand et al. [2]

(download available from GEO `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE9819`). There are 4 samples of both universal human reference (UHR) and brain (BRAIN) tissue. To see which *CEL* files are available in the directory, try:

```
> list.files( celdir )

[1] "BRAIN01_GSM247696.CEL" "BRAIN02_GSM247697.CEL" "BRAIN03_GSM247700.CEL"
[4] "BRAIN04_GSM247701.CEL" "UHR01_GSM247694.CEL"   "UHR02_GSM247695.CEL"
[7] "UHR03_GSM247698.CEL"   "UHR04_GSM247699.CEL"
```

3 Differential expression analysis

3.1 Defining data

In order to run the analysis one must first define an experiment. Experiment is a ***DEMIExperiment*** object that contains the required metadata that will be used in the analysis. Several analysis can be run on the same ***DEMIExperiment*** object which reduces the redundancy of having to load the metadata several times. An experiment can be created using the function ***DEMIExperiment***, that depending on the analysis type has four or five compulsory parameters: ***analysis***, ***experiment***, ***organism***, ***celpath*** and ***sectionsize***. ***DEMIExperiment*** is not the actual analysis itself but more like a container of data that is used in the later stages of the analysis.

analysis Defines the type of the analysis to be performed. It can be either "exon", "transcript", "gene" or "genome". The first three are all self explanatory but "genome" analysis needs further explanation. More specifically, the "genome" analysis uses genomic regions of specified size (set by the ***sectionsize*** parameter) as the targets for differential expression analysis. This kind of analysis has the potential of detecting differential expression in larger clusters of genes based on genomic proximity providing a genome-level view of differential expression events. For example, one might use it to detect long range epigenetic silencing effects where neighbouring genes are predominantly down-regulated.

experiment Defines a name for the experiment which is useful for the user to identify experiments later.

organism Defines the organism name. The organism name needs to be provided in lower case letters with words seperated by underscore (e.g. "homo_sapiens"). This is important for the demi package to be able to load annotations in the demi annotation packages for different microarray platforms. Each demi data package is specified by the organism name and it contains all currently implemented microarray platform annotations for this species. If, for some reason, your microarray platform is not available for analysis, you can contact us and we will try to add it and update the demi annotation packages accordingly.

celpath Defines the directory of the *CEL* files or is a character vector of the *CEL* file names (with the full path). It is a good practice to name your *CEL* files according to their features. For example, names of *CEL* files representing replicates of an experimental condition must contain a common tag in order for DEMI to identify them. As in the example dataset above, all *CEL* files that measure gene expression activity in the brain contained the substring "BRAIN" in their file names. All *CEL* files measuring UHR contained the "UHR" substring in their file names. This makes it easy to recoqnize different features of the files and allows DEMI analysis to group them accordingly.

sectionsize This is only used when the `analysis` parameter is set to “genome”. It specifies the size of genomic regions used as targets in the analysis. The regions are designed to overlap with the neighboring regions by 50%. Currently sectionsizes 100000, 500000 and 1000000 are available.

An example of using “gene” analysis with specified *CEL* files directory:

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc",
organism = "homo_sapiens", celpath = celdir )
```

To run the analysis on transcripts, exons or genome regions as targets, simply change the `analysis` parameter accordingly. An example in using “genome” analysis with specified `sectionsize` and specific *CEL* files:

```
> demiexp <- DEMIExperiment( analysis = "genome", experiment = "maqc",
organism = "homo_sapiens", celpath = paste( celdir, list.files( celdir ),
sep = "" ), sectionsize = 500000 )
```

By providing these parameters the package automatically detects the microarray platform from the *CEL* files and loads the required data from demi data packages (if the data packages are not available in the local library it will download and install the data packages from demi repository on the first run of demi analysis). It then loads the raw expression matrix from the *CEL* files and automatically normalizes it with `DEMIExperiment` default parameters (see section 4.1 on how to override the default parameters). When the `DEMIExperiment` object has been created one can move on to cluster the probes based on their expression dynamics as inferred from the normalized intensity values.

3.2 Clustering of probes - evaluation of probe expression profiles

Clustering of probes means that the normalized signal intensities located in the `DEMIExperiment` object (see section 3.1) are statistically evaluated in order to group probes with similar expression profiles. In the simplest case of comparing groups TEST and CONTROL, this will produce three lists of probes - probes that have significantly higher median signal intensity value in condition TEST compared to condition CONTROL, probes that have significantly lower median signal intensity value in condition TEST compared to CONTROL and probes with no significant difference in the median signal between the conditions. For example, in the MAQC dataset two types of RNA samples are used - brain RNA (BRAIN) and universal human reference RNA (UHR). Clustering will produce two list of probes with higher and lower median signal intensities in the BRAIN, respectively.

The clustering described above is done with a `DEMIClust` object by running the `DEMIClust` function. Two parameters are required by `DEMIClust`: `experiment` and `group`.

experiment Denotes the previously created `DEMIExperiment` object that contains all the metadata used in the analysis.

group Is a two element character vector containing group (can also denote conditions or features) names used in the comparison. These group names are checked against the names of *CEL* files. All *CEL* files that contain the first character in `group` vector are set to groupA and all the *CEL* files that contain the second character in `group` vector are set to groupB. The clustering will then compare signal intensities between groupA and groupB.

```
> demiclust <- DEMIClust( experiment = demiexp, group = c( "BRAIN", "UHR" ) )
```

It is also possible to create a `DEMIClust` object with probes selected specifically by the user. If the user wants to run the differential expression analysis on specific probes only, the `DEMIClust` object allows the user to submit custom probe lists. This is done by using two parameters: `experiment` and `cluster`.

experiment Denotes the previously created `DEMIExperiment` object that contains all the metadata used in the analysis. In this case the probes in the `cluster` lists are checked for their presence in the metadata. If they are not available an error message will be displayed. This restricts the user to use only probes that are available in the analysis. An obvious use case for custom clusters would be when the user wants to group probes with differential expression in more than one comparison by taking the overlap of probes that have differential expression in several comparisons.

cluster Is a list of vectors containing customly chosen probes. The vectors in the list need to be properly named, otherwise they won't be recoqnizable in the later stages of the analysis.

For this example we utilize the function `getTargetProbes` (see section 4.3 for more information about the function) to retrieve probes of specific genes that we know are differentially expressed in our dataset. We use these probes to make three custom clusters of probes, each representing a cluster containing probes measuring the specified gene.

```
> custom_clusters <- list(
      tcf4 = getTargetProbes( demiexp, "TCF4" )$probeID,
      cd44 = getTargetProbes( demiexp, "CD44" )$probeID,
      calm1 = getTargetProbes( demiexp, "CALM1" )$probeID
      )
```

```
> demicustom <- DEMIClust( experiment = demiexp, cluster = custom_clusters )
```

When the probe clusters have been created one can move on to performe differential expression analysis.

3.3 Differential expression analysis - from probes to annotation

The goal of differential expression analysis is to determine exons, transcripts, genes or genomic regions that have distinct expression profiles between experimental conditions. By now we have loaded metadata for the analysis in a `DEMIExperiment` object and clustered the probes based on their normalized signal intensities with a `DEMIClust` object. Although we already have lists of probes that have statistically significant differential signal intensities between two experimental conditions it is difficult to make conclusions just based on probes. We need to be able to relate these probes either to exons, transcripts, genes or genomic regions i.e. to their corresponding targets. To understand how targets behave in specified experimental conditions we need to create a *DEMIDiff* object that requires only one parameter: *cluster*.

The *cluster* parameter is actually a previously created *DEMIClust* object (see section 3.2). The *DEMIDiff* object takes the input *cluster* parameter and uses hypergeometric probability distribution to determine whether the target is differentially expressed. The calculation is done using the following formula:

$$pXH = HG(XH, XT, T - XT, H) \tag{1}$$

pHX H0 probability that target X (a gene or a transcript, for example) is up-regulated

HG Hypergeometric probability function

XH Number of up-regulated probes matching target X

XT Total number of probes matching target X

T Total number of probes in the analysis

H Total number of up-regulated probes in the analysis

The *DEMIDiff* object analyzes the data with the given formula, annotates the results and subsequently stores them in a *DEMIResults* object stored within a *DEMIDiff* object.

Let us now calculate differentially expressed genes for the two *DEMIClust* objects that were created before - for automatically clustered probes and for user defined probe clusters (see section 3.2).

```
> demidiff <- DEMIDiff( cluster = demiclust )
```

```
> customdiff <- DEMIDiff( cluster = demicustom )
```

3.4 Functional annotation analysis

The demi R package contains built in functionality to run functional annotation analysis on the differential expression results obtain with DEMI analysis. The function to perform functional annotation analysis is *DEMIPathway* that returns the results of functional annotation analysis in a *data.frame*. The *DEMIPathway* function requires as input a *DEMIDiff* object (see section 3.3) where the results from differential expression analysis are stored. The function *DEMIPathway* can only be used if the analysis was contucted on ”gene” or ”transcript” as analysis type.

```
> demipath <- DEMIPathway( demidiff )
```

3.5 DEMI wrapper function - one function to do it all

Since DEMI analysis in most cases follows a fairly standard procedure there is one function that implements the whole analysis from top to bottom without having to create seperate DEMI objects for each step. It is a wrapper around all DEMI objects like *DEMIExperiment*, *DEMIClust*, *DEMIDiff* and *DEMIPathway*. The function to implement this is ***demi***. It uses the same parameters as the DEMI objects described in previous sections, incorporating them all into one function. It also prints the results directly to the working directory of the R session and retuns a list containing a DEMIExperiment object with the attached results and a data.frame with functional annotation analysis results when performed. The parameters for ***demi*** function are:

analysis See section 3.1 for explanation of the same parameter under the *DEMIExperiment* object.

experiment See section 3.1 for explanation of the same parameter under the *DEMIExperiment* object.

organism See section 3.1 for explanation of the same parameter under the *DEMIExperiment* object.

celpath See section 3.1 for explanation of the same parameter under the *DEMIExperiment* object.

sectionsize See section 3.1 for explanation of the same parameter under the *DEMIExperiment* object.

group See section 3.2 for explanation of the same parameter under the *DEMIClust* object.

cluster See section 3.2 for explanation of the same parameter under the *DEMIClust* object.

filetag If the parameter ***filetag*** has been set then the output files automatically contain the character set in the ***filetag*** parameter.

pathway This parameter defines whether to perform functional annotation analysis on the results from differential expression analysis. If set to ***TRUE*** it will also print out the results from functional annotation analysis to a file in the working directory of the R session. This file contains the character string “pathway” in it. The pathway analysis can only be done if the *analysis* parameter has been set to either ”gene” or ”transcript”.

maxtargets See section 4.1 for explanation of the same parameter under the DEMIExperiment object.

maxprobes See section 4.1 for explanation of the same parameter under the DEMIExperiment object.

pmsize See section 4.1 for explanation of the same parameter under the DEMIExperiment object.

norm.method See section 4.1 for explanation of the same parameter under the DEMIExperiment object.

clust.method See section 4.2 for explanation of the same parameter under the DEMIClust object.

cutoff.pvalue See section 4.2 for explanation of the same parameter under the DEMIClust object.

The output files of the *demi* function starts with the string "demi" and incorporates the *analysis* parameter, the *experiment* parameter and *filetag* parameter when applied (if applied), separated by an underscore.

An example of differential expression analysis:

```
> demires <- demi(analysis = "gene", celpath = celdir, group = c( "BRAIN", "UHR" ),
experiment = "maqc", organism = "homo_sapiens", clust.method = demi.wilcox.test.fast,
pathway = FALSE)
```

An example of differential expression with functional annotation analysis:

```
> demires <- demi(analysis = "gene", celpath = celdir, group = c( "BRAIN", "UHR" ),
experiment = "maqc", organism = "homo_sapiens", clust.method = demi.wilcox.test.fast,
pathway = TRUE)
```

3.6 Viewing results

Using the function *getResultTable* returns the results stored in a *DEMIDiff* or *DEMIExperiment* object in a *data.frame*, sorted by the false discovery rate values.

```
> head( getResultTable( demidiff ) )
         clusterID        targetID probesOnTargetInCluster probesOnTarget
16694 BRAIN[H]_UHR ENSG00000196628                      90             90
8039  BRAIN[H]_UHR ENSG00000140538                      93             98
213   BRAIN[H]_UHR ENSG00000008277                      85             87
14294 BRAIN[H]_UHR ENSG00000178104                     100            111
30730 BRAIN[L]_UHR ENSG00000026508                      86             86
6998  BRAIN[H]_UHR ENSG00000134769                      81             84
      probesInCluster probesTotal      P.value          FDR geneSymbol
16694           87116      412770 1.511205e-61 9.148233e-57       TCF4
8039            87116      412770 2.987339e-56 9.042077e-52      NTRK3
213             87116      412770 8.492751e-55 1.432182e-50     ADAM22
14294           87116      412770 9.463343e-55 1.432182e-50    PDE4DIP
30730          100924      412770 2.402911e-53 2.909252e-49       CD44
6998            87116      412770 8.659616e-51 8.736975e-47       DTNA
                                                                description
16694               transcription factor 4 [Source:HGNC Symbol;Acc:11634]
8039  neurotrophic tyrosine kinase, receptor, type 3 [Source:HGNC Symbo...
213          ADAM metallopeptidase domain 22 [Source:HGNC Symbol;Acc:201]
14294 phosphodiesterase 4D interacting protein [Source:HGNC Symbol;Acc:...
30730    CD44 molecule (Indian blood group) [Source:HGNC Symbol;Acc:1681]
6998                   dystrobrevin, alpha [Source:HGNC Symbol;Acc:3057]
```

```
> head( getResultTable( customdiff ) )
```

```
      clusterID       targetID probesOnTargetInCluster probesOnTarget
16694      tcf4 ENSG00000196628                      90             90
30730      cd44 ENSG00000026508                      86             86
77822     calm1 ENSG00000198668                      66             66
9147       tcf4 ENSG00000148488                       3              3
17452      tcf4 ENSG00000201010                       3              3
18657      tcf4 ENSG00000213358                       3              3
      probesInCluster probesTotal       P.value           FDR geneSymbol
16694              90      412770  0.000000e+00  0.000000e+00       TCF4
30730              86      412770  0.000000e+00  0.000000e+00       CD44
77822              66      412770 1.262975e-278 3.822774e-274      CALM1
9147               90      412770  1.002291e-11  2.676825e-08    ST8SIA6
17452              90      412770  1.002291e-11  2.676825e-08        7SK
18657              90      412770  1.002291e-11  2.676825e-08 AC092933.4
                                                             description
16694              transcription factor 4 [Source:HGNC Symbol;Acc:11634]
30730   CD44 molecule (Indian blood group) [Source:HGNC Symbol;Acc:1681]
77822 calmodulin 1 (phosphorylase kinase, delta) [Source:HGNC Symbol;...
9147  ST8 alpha-N-acetyl-neuraminide alpha-2,8-sialyltransferase 6 [S...
17452                                  7SK RNA [Source:RFAM;Acc:RF00100]
18657
```

One can see from the table above, that the three genes we used for creating custom probe lists were also the best scoring differentially expressed genes i.e. genes with the smallest p-values. There are also some other genes with p-values below 0.05 because some of these probes match with several genes (see the use of *maxtarget* parameter in the section 4.1 to address this issue).

Since both differential expression results (demidiff, customdiff) were obtained using the same metadata, meaning that they used the same underlying *DEMIExperiment* object, we can utilize the function *attachResult*. It allows us to attach the results from a *DEMIDiff* object to the initial *DEMIExperiment* object. It is useful in the case where several differential expression analyses were made and the user wants to view them together in one table. The function *attachResult* requires two parameters: *object* and *diffObject*. The first parameter *object* is a *DEMIExperiment* object to which the second parameters *diffObject* results will be attached to. The function *attachResult* then returns a new *DEMIExperiment* object that has been updated with the results from *DEMIDiff* object. If the results have already been added to the *DEMIExperiment* object, the program will alert the user and the results won't be added again, thus prohibiting duplications.

```
> demiresult <- attachResult( demiexp, demidiff )
```

```
> demiresult <- attachResult( demiresult, customdiff )
```

The previously described *getResultTable* function can now be used with the newly created *DEMIExperiment* object.

```
> getResultTable( demiresult )
          clusterID        targetID probesOnTargetInCluster probesOnTarget
77230          tcf4 ENSG00000196628                      90             90
307301         cd44 ENSG00000026508                      86             86
778221        calm1 ENSG00000198668                      66             66
16694  BRAIN[H]_UHR ENSG00000196628                      90             90
8039   BRAIN[H]_UHR ENSG00000140538                      93             98
213    BRAIN[H]_UHR ENSG00000008277                      85             87
       probesInCluster probesTotal       P.value           FDR geneSymbol
77230               90      412770  0.000000e+00  0.000000e+00       TCF4
307301              86      412770  0.000000e+00  0.000000e+00       CD44
778221              66      412770 1.262975e-278 3.822774e-274      CALM1
16694            87116      412770  1.511205e-61  9.148233e-57       TCF4
8039             87116      412770  2.987339e-56  9.042077e-52      NTRK3
213              87116      412770  8.492751e-55  1.432182e-50     ADAM22
                                                               description
77230                transcription factor 4 [Source:HGNC Symbol;Acc:11634]
307301    CD44 molecule (Indian blood group) [Source:HGNC Symbol;Acc:1681]
778221 calmodulin 1 (phosphorylase kinase, delta) [Source:HGNC Symbol;A...
16694                transcription factor 4 [Source:HGNC Symbol;Acc:11634]
8039   neurotrophic tyrosine kinase, receptor, type 3 [Source:HGNC Symb...
213           ADAM metallopeptidase domain 22 [Source:HGNC Symbol;Acc:201]
```

One can see from the table above that the cluster names previously represented in different *DEMIDiff* objects are now attached to the same *DEMIExperiment* object and can be viewed together.

Another function helping the user understand their data is *demisummary* that has two input parameters: *object* and *target*. The first is a *DEMIExperiment* or *DEMIDiff* object and the second is a vector of ensembl gene ID's or gene symbols (e.g. "MAOB"), ensembl transcript ID's, ensembl peptide ID's or genomic region ID's. If no results have been attached to the *DEMIExperiment* object then only the mean normalized expression values over the entire dataset are returned.

```
> demisummary( demiexp, c( "SOX2", "NANOG" ) )
         targetID      ALL
1 ENSG00000111704 76.65586
2 ENSG00000181449 51.47945
```

```
> demisummary( demidiff, c( "SOX2", "NANOG" ) )
         targetID    BRAIN      UHR      ALL
1 ENSG00000111704 69.15389 66.56352 76.65586
2 ENSG00000181449 61.14888 62.07785 51.47945
```

If some results have been attached to a *DEMIExperiment* object then the function *demisummary* will also output the means over the groups that were used for differential expression analysis, as is the case when the input object is a *DEMIDiff* object.

```
> demisummary( demiresult, c( "SOX2", "NANOG" ) )

         targetID    BRAIN      UHR      ALL
1 ENSG00000111704 69.15389 66.56352 76.65586
2 ENSG00000181449 61.14888 62.07785 51.47945
```

Note that in the latter the mean normalized expression values for the customly created probe clusters are not shown. The reason is that no particular *CEL* files correspond to these clusters and therefore calculating the mean over the normalized expression values can't be done.

3.7 Graphic representation

There are two functions `probe.levels` and `probe.plot` that help to give a visual insight on the behaviour of the probes in different *CEL* files. Both functions require two input parameters: `object` and `target`. The `object` parameter specifies the `DEMIExperiment` object where all the raw and normalized expression values are stored. The `target` parameter can be an ensembl gene symbol, ensembl gene ID, ensembl transcript ID, ensembl protein ID or an genomic region ID.

`> probe.levels( demiresult, c( "MAOB" ) )` (Figure 1)

`> probe.plot( demiresult, c( "MAOB" ) )` (Figure 2)

4 Advance usage

4.1 Advance analysis - DEMIExperiment

The `DEMIExperiment` function has several parameters which can be used when the user wants to be more stringent in selecting the probes used in the analysis or to utilize a custom normalization method.

maxtarget The `maxtarget` parameter specifies the upper limit of distinct targets a probe can match to i.e. the number of targets a probe is designed to measure. For example, if two genes share a common sequence and the probe sequence is designed in a way that it matches to both of these genes then this probe is designed to measure 2 targets. In the case of "transcript" and "exon" analysis the number of targets is also calculated against genes, meaning that if two transcripts are measured by the same probe but they both correspond to the same gene, the number of distinct targets for this probe is 1 (and not 2). All probes matching to more targets than set by the `maxtarget` parameter will be excluded from the analysis. The default value for `maxtarget` is 0, meaning that a probe can measure any number of targets.

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, maxtarget = 1 )
```

maxprobes Since hypergeometric probability values are calculated to indicate whether the target is differentially expressed between the experimental conditions, differentially expressed targets with many matching probes (overrepresented targets) tend to occupy the top of differentially expressed target list. To counteract this issue the user can set `maxprobes` parameter which dictates the upper limit on the number of probes a target can possess. For targets with

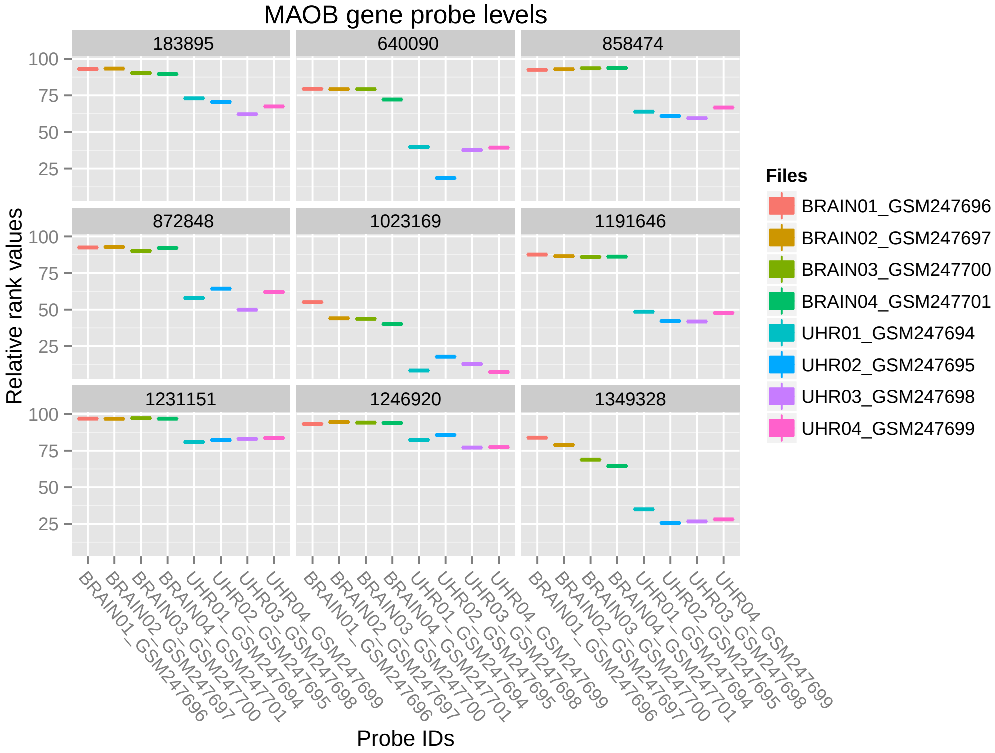

Figure 1: Normalized probe levels measuring the gene MAOB

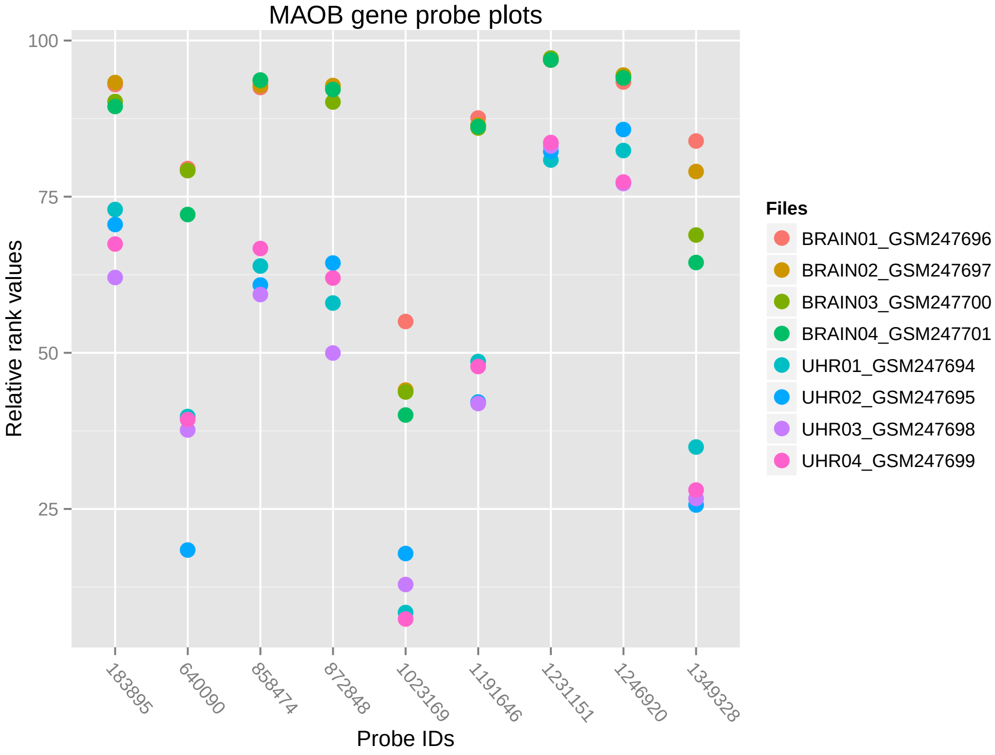

Figure 2: Plot of normalized probe levels measuring the gene MAOB

more matching probes than allowed by the `maxprobes` parameter, the number of matching probes and the number of differentially expressed matching probes are scaled down to the number of probes set by `maxprobes` parameter. The user can also set the `maxprobes` parameter to "median" instructing the software to set the `maxprobes` parameter to the value corresponding to the median number of probes per target. By default the `maxprobes` parameter is not set and no scaling takes place, which would be the same as setting `maxprobes` to "max".

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, maxprobes = "max" )
```

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, maxprobes = "median" )
```

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, maxprobes = 13 )
```

pmsize The parameter `pmsize` denotes the minimum alignment hit size between the probe and the target sequence. The default for `pmsize` is 25 which means that only perfectly matching probes will be regarded as measuring matching targets. If the user sets `pmsize` to 24, then probes with a single mismatch will also be regarded as measuring the targets. By setting `pmsize` parameter to 25 might reduce the noise from off-target hybridization. However, the kinetics of oligonucleotide hybridization are rather complex and this need not be so (when the mismatch occurs in the terminus, for example). In any case, the user is allowed to choose a `pmsize` between 23 and 25.

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, pmsize = 23 )
```

norm.method The `norm.method` parameter denotes a function to be used for the normalization of the raw expression matrix. It should take `matrix` as input and return the normalized `matrix`. By default the `norm.method` is set to use relative rank normalization (`norm.rrank`) which normalizes all the *CEL* files individually. Each probe will have a rank value denoting the procentage of probes with signal intensity value smaller or equal to the current probe.

```
> demiexp <- DEMIExperiment( analysis = "gene", experiment = "maqc", organism
= "homo_sapiens", celpath = celdir, norm.method = norm.rrank )
```

4.2 Advance analysis - DEMIClust

When clustering the probes based on normalized expression matrix the user can modify several parameters in a `DEMIClust` function. For example the user can build their own function for clustering or set a specific cutoff on the differential expression p-value of a probe.

clust.method The `clust.method` parameter is a function that is used to cluster the normalized expression matrix of probes. It takes as input the `DEMIClust` object containing information about grouping of the *CEL* files and a normalized expression matrix and returns a list of probes that are up- or down-regulated. The names of the lists are the corresponding cluster names. The user can write a novel clustering function as long as the same input and output are used. See subsection 4.3 for some functions that can be helpful in writing a new normalization method.

```
> demiclust <- DEMIClust( experiment = demiexp, group = c("BRAIN", "UHR" ),
clust.method = demi.wilcox.test.fast )
```

cutoff.pvalue The *cutoff.pvalue* parameter dictates the statistical p-value cutoff that is used to call statistical significance. All probes whose p-value is below the set cutoff will be regarded as differentially expressed.

```
> demiclust <- DEMIClust( experiment = demiexp, group = c("BRAIN", "UHR" ),
cutoff.pvalue = 0.05 )
```

4.3 Extra functionality

This sections describes other functions that the user might find necessary.

getAnalysis Returns the analysis of the *DEMIExperiment* object.

```
> getAnalysis( demiexp )
```

getExperiment Returns the name of the experiment in the *DEMIExperiment* object. In the case of *DEMIDiff* and *DEMIClust* objects this function returns the original *DEMIExperiment* object.

```
> getExperiment( demiexp )
```

getCelpath Returns the directory or the files vector of the *CEL* files in the *DEMIExperiment* object.

```
> getCelpath( demiexp )
```

getOrganism Returns the organism in the *DEMIExperiment* object.

```
> getOrganism( demiexp )
```

getArraytype Returns the microarray platform type in the *DEMIExperiment* object.

```
> getArraytype( demiexp )
```

getMaxtargets Returns the maximum number of targets a probe is allowed to measure from the *DEMIExperiment* object.

```
> getMaxtargets( demiexp )
```

getMaxprobes Returns the maximum number of probes a target can have a match against in the *DEMIExperiment* object.

```
> getMaxprobes( demiexp )
```

getAnnotation Returns the annotation table in the *DEMIExperiment* object.

```
> getAnnotation( demiexp )
```

getAlignment Returns the sequence alignment information table in the *DEMIExperiment* object. This can be a very useful function if the user wants to build custom filters for probes when normalizing the raw expression matrix.

```
> getAlignment( demiexp )
```

getCytoband Returns the karyotype information table in the *DEMIExperiment* object. It returns it only if the *analysis* paramater of the *DEMIExperiment* object has been set to "genome".

```
> getCytoband( demiexp )
```

getCelMatrix Returns the raw expression matrix in the *DEMIExperiment* object. This can be used to view raw probe signal intensities of probes. Note that the probe ID's are returned as row names.

```
> getCelMatrix( demiexp )
```

getNormMatrix Returns the normalized expression matrix in the *DEMIExperiment* object. Note that the probe ID's are returned as row names.

```
> getNormMatrix( demiexp )
```

getProbeLevel Returns the normalized probe levels of selected probes of a *DEMIExperiment* or a *DEMIDiff* object. Note that the probe ID's are returned as row names.

```
> getProbeLevel( demiexp, c( 1171,1182 ), TRUE )
```

```
> getProbeLevel( demidiff, c( 1171,1182 ), TRUE )
```

getTargetProbes Returns the probes of selected targets of a *DEMIExperiment* or a *DEMIDiff* object. The target can be case insensitive.

```
> getTargetProbes( demiexp, "MAOB" )
```

```
> getTargetProbes( demidiff, "MAOB" )
```

getTargetAnnotation Returns the annotation of selected targets of a *DEMIExperiment* object. The target can be case insensitive.

```
> getTargetAnnotation( demiexp, "MAOB" )
```

getGroup Returns the *DEMIGroup* object of a *DEMIClust* or a *DEMIDiff* object.

```
> getGroup( demiclust )
```

```
> getGroup( demidiff )
```

getClustMethod Returns the clustering function used in the *DEMIClust* object.

```
> getClustMethod( demiclust )
```

getCutoffPvalue Returns the cutoff p-value used in the *DEMIClust* object.

```
> getCutoffPvalue( demiclust )
```

getCluster Returns the clusters of the *DEMIClust* object.

```
> getCluster( demiclust )
```

getName Returns the name of the *DEMIDiff* object.

```
> getName( demidiff )
```

getGroupA Returns the group A name of the *DEMIGroup* object that is stored within the *DEMIClust* or *DEMIDiff* object.

```
> getGroupA( getGroup( demiclust ) )
```

```
> getGroupA( getGroup( demidiff ) )
```

getGroupB Returns the group B name of the *DEMIGroup* object stored within the *DEMIClust* or *DEMIDiff* object.

```
> getGroupB( getGroup( demiclust ) )
```

```
> getGroupB( getGroup( demidiff ) )
```

getIndexA Returns the group A column indexes of the normalized expression matrix of the *DEMIGroup* object that is stored within the *DEMIClust* or *DEMIDiff* object.

```
> getIndexA( getGroup( demiclust ) )
```

```
> getIndexA( getGroup( demidiff ) )
```

getIndexB Returns the group B column indexes of the normalized expression matrix of the *DEMIGroup* object that is stored within the *DEMIClust* or *DEMIDiff* object.

```
> getIndexB( getGroup( demiclust ) )
```

```
> getIndexB( getGroup( demidiff ) )
```

getGroupNames In the case when probe clusters were manually created the function *getGroupNames* returns the names of the probe clusters set by the user in the *DEMIGroup* object that is stored within the *DEMIClust* or *DEMIDiff* object.

```
> getGroupNames( getGroup( demicustom ) )
```

```
> getGroupNames( getGroup( customdiff ) )
```

5 Summary

To sum up the functionality of DEMI package the user is only required to run four functions ***`DEMIExperiment`***, ***`DEMIClust`***, ***`DEMIDiff`*** and ***`getResultTable`*** to retrieve a table of differentially expressed targets. The ***`DEMIExperiment`*** holds the metadata and normalizes the raw expression matrix while ***`DEMIClust`*** clusters the probes into up- or down-regulated clusters based on the selected features in the names of the *CEL* files. ***`DEMIDiff`*** evaluates differential expression of the targets with hypergeometric probability while also annotating the results. Finally the user can review the results by utilizing the ***`getResultTable`*** function. This whole functionality is summarized with a function ***`demi`*** that performs all of the above with a single command and prints out the results into the users working directory. In case of questions, do not hesitate to ask. We will gladly help you to get the hang of the software and the methodology involved.

References

[1] MAQC Consortium. The microarray quality control (maqc) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature Biotechnology*, 24:1151–1161, 2006.

[2] Pradervand S., Paillusson A., Thomas J., Weber J., Wirapati P., Hagenbüchle O., and Harshman K. Affymetrix whole-transcript human gene 1.0 st array is highly concordant with standard 3' expression arrays. *Biotechniques*, 44(6):759–762, 2008.