The RTools4TB package: data mining of public microarray data through connections to the TranscriptomeBrowser database.

A. Bergon, F. Lopez, J. Textoris, S. Granjeaud and D. Puthier

October 28, 2009

TAGC/Inserm U928. Parc Scientifique de Luminy case 928. 163, avenue de Luminy. 13288 MARSEILLE cedex 09. FRANCE
bergon@tagc.univ-mrs.fr

Contents

1 Overview

TranscriptomeBrowser (TBrowser, `http://tagc.univ-mrs.fr/tbrowser`) hosts a large collection of transcriptional signatures (TS) automatically extracted from the Gene Expression Omnibus (GEO) database. Each GEO experiment (GSE) was processed so that a subset of the original expression matrix containing the most relevant/informative genes was kept and organized into a set of homogeneous signatures [1]. Each signature was tested for functional enrichment using annotations terms obtained from numerous ontologies or curated databases (Gene Ontology, KEGG, BioCarta, Swiss-Prot, BBID, SMART, NIH Genetic Association DB, COG/KOG...) using the DAVID knowledgebase [2].
RTools4TB is a library for data mining of public microarray data. *RTools4TB* can be helpful (i) to define the biological contexts (i.e, experiments) in which a set of genes are co-expressed and (ii) to define their most frequent neighbors [1]. The RTools4TB package also implements the DBF-MCL algorithm ("Density Based Filtering And Markov Clustering") that can be used for fast and automated partitioning of microarray data. DBF-MCL is a tree-step adaptative algorithm that (i) find elements located in dense

areas (ie. clusters) (ii) uses selected items to construct a graph and (iii) performs graph partitioning using MCL [3]. Note that a UNIX-like systems is required to use DBF-MCL.

2 Fetching transcriptional signatures from TBrowserDB

2.1 The getSignatures function

Connection to the TranscriptomeBrowser database (TBrowserDB) relies on the `getSignatures`, `get-ExpressionMatrix` and `getTBInfo` functions.

Basically, the `getSignatures` function can be used to retrieve transcriptional signature IDs using gene symbol(s), probe ID(s), experiment ID, microarray platform ID or annotation term(s) as input. This is controled by the ”field” argument.

```
> library(RTools4TB)
> args(getSignatures)

function (field = c("gene", "probe", "platform", "experiment",
    "annotation"), value = NULL, qValue = NULL, nbMin = NULL,
    verbose = TRUE, save = FALSE)
NULL
```

Once the field argument is set, one need to provide a value as input. For instance the following query use gene name as input with value ”PCNA”.

```
> res <- getSignatures(field = "gene", value = "PCNA")
> head(res)
```

Transcriptional signature IDs can also be obtained by selecting the relevant experiment IDs, platform IDs and probe IDs. To get all transcriptional signature IDs associated with GSE2004 experiment, one should use the following syntax:

```
> res <- getSignatures(field = "experiment", value = "GSE2004")

23 signatures were found for the request:
 GSE  =  GSE2004
```

To get all signatures obtained on GPL96 platform, use the following syntax:

```
> res <- getSignatures(field = "platform", value = "GPL96")

3377 signatures were found for the request:
 GPL  =  GPL96
```

Moreover, as all signatures were tested for functional enrichment using keywords from the DAVID knowledgebase these terms can be used to query the database (DAVID collects a wide range of annotation from several databases including: GO, BIOCARTA, KEGG, PANTHER, BBID,...) . The `annotationList` dataset contains the annotations terms.

```
> data(annotationList)
> names(annotationList)

[1] "Keyword"   "TableName"
```

```
> attach(annotationList)
> annotationList[1:4, ]
```

```
                       Keyword TableName
1              1.RBphosphoE2F      BBID
2  100.MAPK_signaling_cascades      BBID
3        104.Insulin_signaling      BBID
4 105.Signaling_glucose_uptake      BBID
```

```
> table(TableName)
```

```
TableName
                  BBID               BIOCARTA        COG_KOG_ONTOLOGY
                    57                    468                      22
              CYTOBAND GENETIC_ASSOCIATION_DB            GOTERM_BP_ALL
                   526                     68                    1273
         GOTERM_CC_ALL          GOTERM_MF_ALL           INTERPRO_NAME
                   328                    639                     777
          KEGG_PATHWAY          KEGG_REACTION           OMIM_PHENOTYPE
                   334                     78                      10
       PANTHER_PATHWAY              PFAM_NAME     PIR_HOMOLOGY_DOMAIN
                   104                    531                      86
  PIR_SUPERFAMILY_NAME              PUBMED_ID               SMART_NAME
                   160                   4887                     235
       SP_PIR_KEYWORDS
                   568
```

The selected terms can be used to select TS IDs. In this case, user should define a q-value. For instance one can select TS enriched in genes related to the "HSA04110:CELL CYCLE" KEGG pathway with q-value below $10e^{-20}$.

```
> cc <- getSignatures(field = "annotation", value = "HSA04110:CELL CYCLE",
+     qValue = 20)
```

```
66 signatures were found for the request:
 annotation  =  HSA04110:CELL CYCLE
```

Of note, one can also search for TS IDs containing genes located in the same chromosomal region. For instance one can select TS IDs enriched in genes located in the "8q" region which is frequently amplified or deleted in tumors. This will point out the biological contexts in which sets of genes located in the 8q region share the same expression profile, suggesting amplifications or deletions in some biological samples.

```
> query <- paste(grep("^8q", Keyword, val = T), collapse = "|")
> query
```

```
[1] "8q13|8q21|8q21.11|8q21.2|8q22.1|8q22.3|8q24|8q24.13|8q24.3"
```

```
> cc <- getSignatures(field = "annotation", value = query, qValue = 10)
```

```
4 signatures were found for the request:
 annotation  =  8q13|8q21|8q21.11|8q21.2|8q22.1|8q22.3|8q24|8q24.13|8q24.3
```

Next sections will introduce more complexe queries using sets of genes with or without Boolean operators.

2.1.1 Request without logical operators (gene list)

When field is set to ”gene” or ”probe”, user can perform a request using a list of item separated by blanks. These blanks are interpreted as the OR logical operators. In this case, all signatures containing at least one gene of the list will be returned. To select more informative signatures we suggest to use the nbMin argument that will select signatures containing at least nbMin genes out of the list.

The following examples search for signatures containing at least 2 genes of the input list (CD3D, CD3E and CD4).

```
> gl <- getSignatures(field = "gene", value = "CD3D CD3E CD4",
+     nbMin = 2)
```

```
150 signatures were found for the request:
 gene  =  CD3D CD3E CD4 and nbMin =  2
```

```
> head(gl)
```

```
  Signature nb.Genes
1 03AD63FB5        2
2 050367D10        2
3 053ECFACF        3
4 05F2203B7        2
5 0C0A8F888        2
6 0D2EA9D52        2
```

2.1.2 Request using logical operators

The ”value” argument of `getSignatures` may contain the following Boolean operators (see help section on TranscriptomeBrowser web site for more informations, `http://tagc.univ-mrs.fr/tbrowser`)

- & : AND
- | : OR
- ! : NOT , (used in conjonction with &)

This is a convenient way to create relevant queries. Suppose your field of interest is related to T-cell activation. You could be interested in retrieving all TS IDs that contain the CD4 gene as they likely contain additional T cell markers. Comparing these TS IDs should help you to define frequent CD4 neighbors (very likely related to TCR signaling cascade). Thereby, your request should be:

```
> res <- getSignatures(field = "gene", value = "CD4")
```

```
371 signatures were found for the request:
 gene  =  CD4
```

This gene is found in 371 TS (with the current database release). Obtaining associated gene lists would be time consuming and would not be as specific as expected. Indeed, the CD4 marker is also expressed by macrophages. Another solution would be to search for TS containing two T-cell markers (CD4 and CD3E for instance) and to exclude (using the NOT operator) those containing the CD14 marker (a macrophages marker). The syntax should be the following:

```
> res <- getSignatures(field = "gene", value = "CD4 & CD3E & !CD14")
```

```
55 signatures were found for the request:
 gene  =  CD4 & CD3E & !CD14
```

In the same way, one can try to exclude TS containing B-cells markers by discarding those containing the CD19 or IGHM markers. The resulting query would be the following:

```
> res <- getSignatures(field = "gene", value = "CD4 & CD3E & !(CD19 | IGHM)")

33 signatures were found for the request:
 gene  =  CD4 & CD3E & !(CD19 | IGHM)
```

2.2 Finding the biological contexts in which sets of genes are co-expressed

As mentioned by Lacroix *et al.*, ESR1, GATA3, XBP1 are co-expressed in breast cancer tumors (see [4]). This assumption can be easily verified using *RTools4TB*. For instance, in the following examples, we fetch transcriptional signature IDs that contain ”XBP1 & ESR1 & GATA3”. Next the `getTBInfo` function is used to retrieve the experiment description from which they are derived (here only for TS ID ”3DE64836D”).

```
> TS <- getSignatures(field = "gene", value = "XBP1 & ESR1 & GATA3")

14 signatures were found for the request:
 gene  =  XBP1 & ESR1 & GATA3

> head(TS)

[1] "0F2635383" "3DE64836D" "59A18E225" "8059848B4" "84E5E1077" "8F69864F9"

> a <- getTBInfo(field = "signature", value = "3DE64836D", verbose = FALSE)
> exp <- a["Experiment", 1]
> info <- getTBInfo(field = "experiment", value = exp, verbose = TRUE)

A result was found for : experiment  =  GSE7904
Name = GSE7904
Organism = Homo sapiens
PMID = NULL
Nb. samples = 62
Title = Expression data from human breast tissue
Summary = bulk breast tumor RNA from patientAbstract: Sporadic basal-like
     cancers (BLC) are a distinct class of human breast cancers that are
     phenotypically similar to BRCA1-associated cancers. Like BRCA1-deficient
     tumors, most BLC lack markers of a normal inactive X chromosome (Xi).
     Duplication of the active X chromosome and loss of Xi characterized almost
     half of BLC cases tested. Others contained biparental but
     nonheterochromatinized X chromosomes or gains of X chromosomal DNA. These
     abnormalities did not lead to a global increase in X chromosome
     transcription but were associated with overexpression of a small subset of
     X chromosomal genes. Other, equally aneuploid, but non-BLC rarely
     displayed these X chromosome abnormalities. These results suggest that X
     chromosome abnormalities contribute to the pathogenesis of BLC, both
     inherited and sporadic.total 62 sample incudes 43 tumor, 7 normal breast
     and 12 normal organelle
```

As expected the transcriptional signature ”3DE64836D” correspond to a breast cancer tumor analysis. This is also true for the other TS (not shown).

2.3 Finding transcriptional neighbors

One interesting feature of *RTools4TB* its ability to find genes frequently co-expressed with the input list. Indeed, results from a request to TBrowserDB can be displayed as a graph using the `createGraph4BioC` function. This function retrieves the list of TS that verify the constrain (here "XBP1 & ESR1 & GATA3"). A list of gene falling in at least one of the TS is next computed. A gene-gene matrix M is created that will record for each pair of gene the number of time they were observed in the same signature. In the following example, only a subset of this adjacency matrix (containing genes falling in a significant proportion of signatures, `prop=80%`) is used to create a graph.

```
> library(biocGraph)

> adjMat <- createGraph4BioC(request = "XBP1 & ESR1 & GATA3", prop = 80)

> g1 <- new("graphAM", adjMat = adjMat)
> nodes(g1)

 [1] "C6orf211"  "GREB1"     "WWP1"      "JMJD2B"    "KRT18"     "RNF103"
 [7] "ROGDI"     "SLC22A5"   "THSD4"     "NAT1"      "SLC39A6"   "ABAT"
[13] "CA12"      "CIRBP"     "LOC400451" "MAGED2"    "MCCC2"     "MLPH"
[19] "ANXA9"     "ERBB4"     "FOXA1"     "ESR1"      "GATA3"     "TBC1D9"
[25] "XBP1"

> nAt <- makeNodeAttrs(g1)
> nAt$fillcolor[match(rownames(as.matrix(nAt$fillcolor)), c("GATA3",
+     "XBP1", "ESR1"), nomatch = F) != 0] <- "green"
> nAt$fillcolor[match(rownames(as.matrix(nAt$fillcolor)), c("TBC1D9",
+     "FOXA1"), nomatch = F) != 0] <- "yellow"
> plot(g1, "fdp", nodeAttrs = nAt)
```

As expected the list of gene contains "XBP1 & ESR1 & GATA3" but also FOXA1/HNF3A that was reported to be co expressed with ESR1 in several experiments (see [4]). Other genes are also particularly relevant such as TBC1D9/MDR1 (Multidrug Resistance 1) (figure 1).

2.4 Vizualising expression matrix.

The TS "3DE64836D" is related to experiment "GSE7904". In this experiments, the authors were interested in analysing several classes of breast cancer tumors especially "Sporadic basal-like cancers".

```
> a <- getTBInfo(field = "signature", value = "3DE64836D", verbose = FALSE)
> exp <- a["Experiment", 1]
> info <- getTBInfo(field = "experiment", value = exp, verbose = TRUE)

A result was found for : experiment  =  GSE7904
Name = GSE7904
Organism = Homo sapiens
PMID = NULL
Nb. samples = 62
Title = Expression data from human breast tissue
Summary = bulk breast tumor RNA from patientAbstract: Sporadic basal-like
     cancers (BLC) are a distinct class of human breast cancers that are
     phenotypically similar to BRCA1-associated cancers. Like BRCA1-deficient
     tumors, most BLC lack markers of a normal inactive X chromosome (Xi).
```

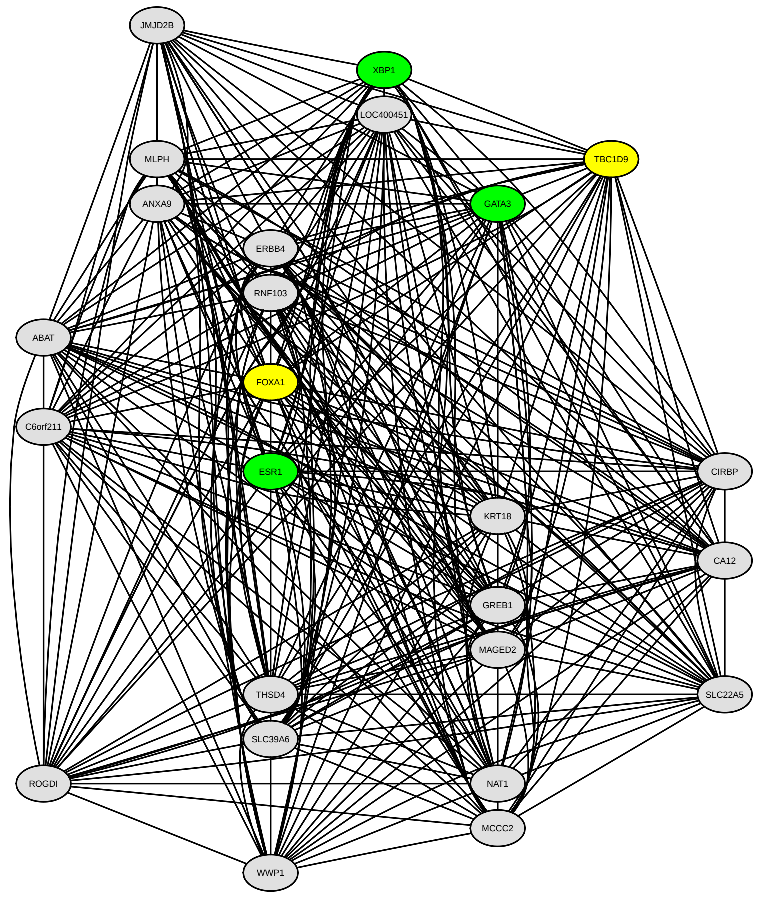

Figure 1: A graph containing "XBP1 & ESR1 & GATA3" together with their most frequent transcriptional neighbors.

```
    Duplication of the active X chromosome and loss of Xi characterized almost
    half of BLC cases tested. Others contained biparental but
    nonheterochromatinized X chromosomes or gains of X chromosomal DNA. These
    abnormalities did not lead to a global increase in X chromosome
    transcription but were associated with overexpression of a small subset of
    X chromosomal genes. Other, equally aneuploid, but non-BLC rarely
    displayed these X chromosome abnormalities. These results suggest that X
    chromosome abnormalities contribute to the pathogenesis of BLC, both
    inherited and sporadic.total 62 sample incudes 43 tumor, 7 normal breast
    and 12 normal organelle
```

The samples that were used are the following.

```
> sampleInfo <- getTBInfo(field = "samples", value = "3DE64836D")
> head(sampleInfo[, 1:2])

     sampleID    Title
[1,] "GSM194397" "Basal (T118)"
[2,] "GSM194398" "Basal (T134)"
[3,] "GSM194399" "Basal (T140)"
[4,] "GSM194400" "Basal (T141)"
[5,] "GSM194401" "Basal (T146)"
[6,] "GSM194402" "Basal (T147)"
```

Using the `getExpressionMatrix` function, the expression matrix for signature "3DE64836D" can be fetched in order to visualize the expression profile of ESR1, GATA3 and XBP1 tumors compare to normal breast tissues.

```
> em <- getExpressionMatrix(signatureID = "3DE64836D")

Downloading expression matrix for transcriptional signature:  3DE64836D ( 62 samples x 143 probes)

> class(em)

[1] "data.frame"
```

The `getExpressionMatrix` function returns a `data.frame`. The first two columns store probe IDs and gene symbols. Additional columns contain corresponding expression values (figure 2).

```
> library(RColorBrewer)
> col <- colorRampPalette(brewer.pal(10, "RdBu"))(256)
> geneNames <- paste(em[, 1], em[, 2], sep = "||")
> em <- as.matrix(em[, -c(1, 2)])
> ind <- match(colnames(em), sampleInfo[, 1])
> colnames(em) <- sampleInfo[ind, 2]
> row <- rep(1, nrow(em))
> ind <- grep("(XBP1)|(ESR1)|(GATA3)", geneNames, perl = TRUE)
> row[ind] <- 2
> rc <- rainbow(2, start = 0, end = 0.3)
> rc <- rc[row]
> col <- colorRampPalette(brewer.pal(10, "RdBu"))(256)
> split <- strsplit(colnames(em), " (", fixed = TRUE)
> pheno <- unlist(lapply(split, "[", 1))
> pheno <- as.factor(pheno)
> levels(pheno) <- 1:5
> cc <- rainbow(5, start = 0, end = 0.3)
> cc <- cc[pheno]
> heatmap(em, col = col, RowSideColors = rc, ColSideColors = cc,
+     labRow = geneNames, cexRow = 0.3)
```

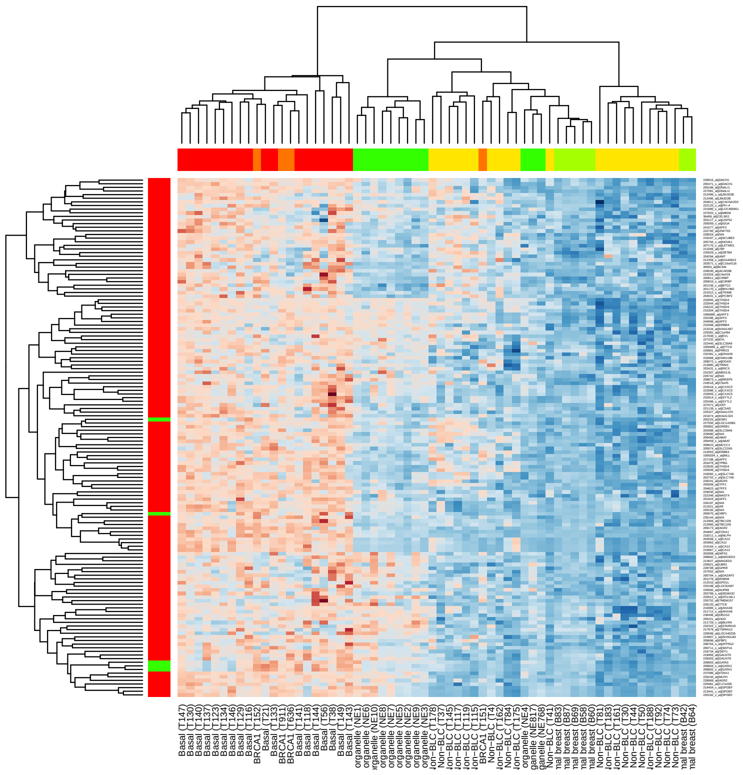

Figure 2: The expression matrix corresponding to signature ”3DE64836D”.

Of note, the `plotGeneExpProfiles` is a high level function to visualize gene expression levels in a signature (figure 3).

```
> plotGeneExpProfiles(data = em, X11 = FALSE)
```

3 Creating transcriptional signatures from a user defined data set using DBF-MCL algorithm.

When analyzing a noisy dataset, one is interested in isolating dense regions as they are populated with genes/elements that display weak distances to their nearest neighbors (i.e. strong profile similarities). To isolate these regions DBF-MCL computes, for each gene/element, the distance with its kth nearest neighbor (DKNN). In order to define a critical DKNN value that will depend on the dataset and below which a gene/element will be considered as falling in a dense area, DBF-MCL computes simulated DKNN values by using an empirical randomization procedure. Given a dataset containing n genes

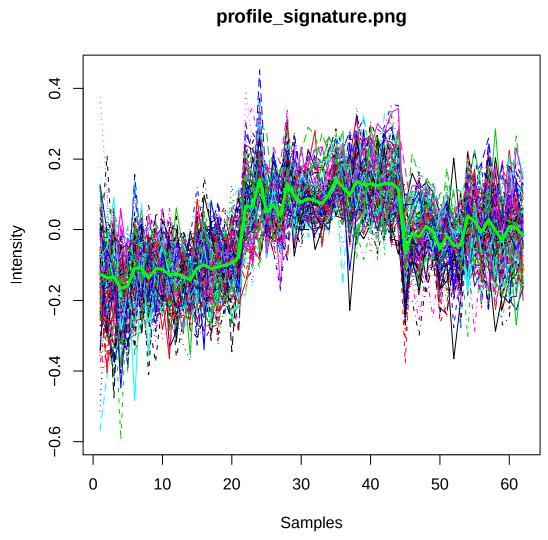

Figure 3: Gene expression profiles of signatures containing XBP1, ESR1 and GATA3: the centroid is highlighted in green.

and p samples, a simulated DKNN value is obtained by sampling n distance values from the gene-gene distance matrix D and by extracting the kth-smallest value. This procedure is repeated n times to obtain a set of simulated DKNN values S. Computed distributions of simulated DKNN are used to compute a FDR value for each observed DKNN value. The critical value of DKNN is the one for which a user-defined FDR value (typically 10%) is observed. Genes with DKNN value below this threshold are selected and used to construct a graph. In this graph, edges are constructed between two genes (nodes) if one of them belongs to the k-nearest neighbors of the other. Edges are weighted based on the respective coefficient of correlation (*i.e.*, similarity) and the graph obtained is partitioned using the Markov CLustering algorithm (MCL).

3.1 Installation

With the current implementation DBFMCL function works only on UNIX-like plateforms. MCL is required and can be installed using your package manager or using the following command lines pasted in a terminal:

```
# Download the latest version of mcl
# (the library has been tested successfully with the 06-058 version)
wget http://micans.org/mcl/src/mcl-latest.tar.gz
# Uncompress and install mcl
tar xvfz mcl-latest.tar.gz
cd mcl-xx-xxx
./configure
make
sudo make install
# You should get mcl in your path
mcl -h
```

3.2 Examples

We will search for transcriptional signatures in a subset of the ALL dataset.

```
> library(ALL)
> data(ALL)
> sub <- exprs(ALL)[1:3000, ]
```

First, we will normalize the data set using the `doNormalScore` function. This function performs normal score transformation of a matrix. The `doNormalScore` transforms each sample to follow a normal distribution (with `mean = 0` and `sd = 1`). Alternatively, users may also use other normalization routines such as : `doRankTransformation` or `limma::normalizeQuantiles`.

```
> subNorm <- doNormalScore(sub)
```

The `DBFMCL` function allows one to extract TS from a data set. Its behaviour is controlled by several arguments.

```
> args(DBFMCL)

function (data = NULL, filename = NULL, path = ".", name = NULL,
    distance.method = c("pearson", "spearman", "euclidean", "spm",
        "spgm"), clustering = TRUE, silent = FALSE, verbose = TRUE,
    k = 150, random = 3, memory.used = 1024, fdr = 10, inflation = 2,
    set.seed = 123, returnRank = FALSE)
NULL
```

The `DBFMCL` function accepts a tab-delimited file (argument `filename`), an expressionSet, a data.frame or a matrix (argument `data`) as input. The input data must contain an expression matrix with gene as rows and samples as columns. Note that space characters inside gene names are not allowed (as they are not supported by the mcl command-line program).

The two main parameters of DBF-MCL are `k` that controls the size of the neighborhood and the `inflation` (range 1.1 to 5) which controls the way the underlying graph is partitioned. In the following example, the neighborhood size (`k`) is set to 150 and the MCL inflation parameters is set to 2.0 (default MCL setting). Most generally these default parameters give very good results on microarray datasets. For a detailed discussion about these parameters please read the section ”Performances of DBF-MCL on Complex9RN200 dataset” in the article describing TranscriptomeBrowser stategy [1]. Morevover, in our example, the distance method is set to ”pearson” although the ”spearman” (that is the default method for computing TS in the TranscriptomeBrowser projet) also give very relevant results. Note that additional distance, including ”euclidean” and two mixtures of pearson” and ”spearman” (”spm” and ”spgm”) are also available.

```
> res <- DBFMCL(subNorm, distance.method = "pearson", memory = 512)
```

The results are stored in an instance of class `DBFMCLresult`.

```
> class(res)

[1] "DBFMCLresult"
attr(,"package")
[1] "RTools4TB"

> res
```

```
An object of class DBFMCLresult
Name: exprs
Memory used:  1140412
Number of samples:  128
Number of informative genes:  1053
Number of clusters:  3
This object contains the following informations:
 -  name
 -  data
 -  cluster
 -  size
 -  center
 -  parameters
 *  distanceMethod  =  pearson
 *  k  =  150
 *  random  =  3
 *  fdr  =  10
 *  set.seed  =  123
 *  inflation  =  2
```

The expression matrix is stored in the `data` slot. This matrix contains only genes detected as informative (that is falling into a cluster).

```
> head(res@data[, 1:2])
```

```
  01005            01010
1153_f_at   1.14383543   1.14401685
1025_g_at -0.35119623 -0.39702813
31604_at    0.21453883  0.35885572
1455_f_at  -0.06960184  0.01066486
1908_at      0.31048075  0.42706913
1492_f_at   0.27323570  0.44352271
```

The partitioning results are stored in the `cluster` slot.

```
> slotNames(res)
```

```
[1] "name"       "data"       "cluster"    "size"       "center"
[6] "parameters"
```

Here, 3 TS were found.

```
> res@size
```

```
[1] 498 461  94
```

The following instruction can be used to get the expression matrix corresponding to the first TS.

```
> res@data[res@cluster == 1, ]
```

The high level function `plotGeneExpProfiles` can be used to visualize, for instance, gene expression profiles corresponding to the first signature.

```
> plotGeneExpProfiles(res, sign = 1)
```

To stored the partitioning results onto disk (as a tab-delimited file), use the `writeDBFMCLresult` function as show below.

```
> writeDBFMCLresult(res, filename.out = "ALL.sign.txt")
```

References

[1] Lopez F.,Textoris J., Bergon A., Didier G., Remy E., Granjeaud S., Imbert J. , Nguyen C. and Puthier D. TranscriptomeBrowser: a powerful and flexible toolbox to explore productively the transcriptional landscape of the Gene Expression Omnibus database. PLoSONE, 2008;3(12):e4001.

[2] Sherman BT, Huang DW, Tan Q, Guo Y, Bour S, Liu D, Stephens R, Baseler MW, Lane HC, Lempicki RA. DAVID Knowledgebase: A Gene-centered Database Integrating Heterogeneous Gene Annotation Resources to Facilitate High-throughput Gene Functional Analysis. BMC Bioinformatics. 2007 Nov 2;8(1):426.

[3] Van Dongen S. (2000) A cluster algorithm for graphs. National Research Institute for Mathematics and Computer Science in the 1386-3681.

[4] Lacroix M, Leclercq G. About GATA3, HNF3A, and XBP1, three genes co-expressed with the oestrogen receptor-alpha gene (ESR1) in breast cancer. Mol Cell Endocrinol. 2004 Apr 30;219(1-2):1-7.