RESEARCH

Analyse multiple disease subtypes and build associated gene networks using genome-wide expression profiles

Sara Aibar[1], Celia Fontanillo[1,3], Conrad Droste[1], Beatriz Rosón[1], Francisco J. Campos-Laborie[1], Jesus M. Hernández-Rivas[2] and Javier De Las Rivas[1*]

Abstract

Background: Despite the large increase of transcriptomic studies that look for gene signatures on diseases, there is still a need for integrative approaches that obtain separation of multiple pathological states providing robust selection of gene markers for each disease subtype and information about the possible links or relations between those genes.

Results: We present a network-oriented and data-driven bioinformatic approach that searches for association of genes and diseases based on the analysis of genome-wide expression data derived from microarrays or RNA-Seq studies. The approach aims to **(i)** identify gene sets associated to different pathological states analysed together; **(ii)** identify a minimum subset within these genes that unequivocally differentiates and classifies the compared disease subtypes; **(iii)** provide a measurement of the *discriminant power* of these genes and **(iv)** identify links between the genes that characterise each of the disease subtypes. This bioinformatic approach is implemented in an R package, named *geNetClassifier*, available as an open access tool in Bioconductor. To illustrate the performance of the tool, we applied it to two independent datasets: 250 samples from patients with four major leukemia subtypes analysed using expression arrays; another leukemia dataset analysed with RNA-Seq that includes a subtype also present in the previous set. The results show the selection of key deregulated genes recently reported in the literature and assigned to the leukemia subtypes studied. We also show, using these independent datasets, the selection of similar genes in a network built for the same disease subtype.

Conclusions: The construction of gene networks related to specific disease subtypes that include parameters such as gene-to-gene association, gene disease specificity and gene discriminant power can be very useful to draw gene-disease maps and to unravel the molecular features that characterize specific pathological states. The application of the bioinformatic tool here presented shows a neat way to achieve such molecular characterization of the diseases using genome-wide expression data.

Keywords: gene; expression; expression profile; gene networks; microarray; RNA-Seq; disease; disease classification; cancer; leukemia; acute leukemia

Background

Last decade of experimental work using genomic technologies has provided many data on gene expression profiling of different biological and pathological states [1]. This great effort in biomedical research has lead to a large need for tools and strategies that allow clinicians to translate the genome-wide expression data into useful information, such as transparent and robust signatures to characterize and distinguish multiple pathological subtypes [2]. There are many machine learning and computational procedures that can be applied to build classification systems that allow identifying the type or category of query samples whose class is not-known *a priori* [3, 4, 5]. However, a common problem of these methods is that they often do not reveal any information about the genes that are selected as variables for the classification process [4]. Although obtaining an efficient classifier might seem enough in some cases, there is a clear loss of biological information if the value or power of the chosen genes is not translated into parameters that allow to characterize and rank the genes.

Many clinical and biomedical studies look for the separation between multiple disease subtypes as distinct pathological states, but they are also very interested in finding the specific genes that are altered in each disease subtype. To identify and quantify the power of such 'marking genes' is the only way by which machine learning techniques can bring back biological meaning to this kind of biomedical studies. Moreover, gene products do not work in isolation as 'independent features', but rather interact with others in biomolecular networks to perform specific biological functions [6]. Therefore, together with the identification of the genes that mark a disease, genome-wide studies of related biological states should also provide information about the associations between the affected genes [7].

Following these questions we have developed a bioinformatic approach to provide gene-based analysis and characterization of different diseases and construction of associated gene networks using expression profiles derived from experimental transcriptomic data. The approach integrates established statistical and ma-

*Correspondence: jrivas@usal.es
[1]Cancer Research Center (IMBCC, CSIC/USAL/IBSAL), Campus Miguel de Unamuno s/n, Salamanca, 37007, Spain
Full list of author information is available at the end of the article

chine learning methods into a single tool that allows to **(i)** identify the set of genes that are specifically altered in a disease when a collection of several diseases -or disease subtypes- are studied and compared together using genome-wide expression profiling; **(ii)** obtain a minimum subset of these genes that enable to differentiate each disease subtype from the other; **(iii)** provide information about how relevant each of these genes is for discriminating each studied class; and **(iv)** find associations between the genes based on the analysis of the experimental expression profiles. This tool has been implemented in an R/Bioconductor package named *geNetClassifier* (available at http://www.bioconductor.org/). In order to validate the tool as a whole and prove whether the results it provides have biological and functional meaning, here we present its application to two independent genome-wide expression datasets of human samples isolated from individuals with different subtypes of leukemia: one using high-density oligonucleotide microarrays and another using deep RNA-sequencing.

Results and discussion

Finding genes associated to specific disease subtypes

The human gene landscape can be structured in functionally associated groups of genes which are specific to biological processes or states. Since a disease will normally affect and alter one or several biological processes, we could depict a theoretical multidimensional "gene space" divided in regions that include genes associated to specific pathological states (**Figure 1A**). The identification of these groups of genes is a great scientific endeavour for biomedical research, and some biological databases (e.g. OMIM [8]) have been built following the idea of a "gene-to-disease mapping", as it is known to happen in Mendelian inherited diseases. In this theoretical scenario, the genes that are affected by a given disease can be overlapping with the ones affected by a similar pathological state. This will define genes that can be altered in multiple pathologies, but it will also expect to define genes that are only affected by a specific malignancy when compared with other diseases.

Considering the recognition of such theoretical gene-disease space (**Figure 1A**), we apply expression profiling to find the genes that are altered in one specific disease subtype using differential expression analysis. To do so, we compare each disease category versus all the others using package EBarrays [9], that implements an empirical Bayes method [10]. This provides a *posterior probability* for each gene to be differentially expressed in one of the classes (see Methods). Sorting the genes by their probability allows to build a ranking of the genes ordered by their statistical significance (**Figure 1B**). Since each gene has a probability of differential expression per class, it is assigned to the class in which it has the best ranking. This allows to build non-overlapping gene lists that optimize the specificity and separation between classes. The posterior probability also allows to quantify the association of a gene with a class and identify how many genes are related to each class at a certain significance level.

Constructing gene-based classifiers for multiple diseases

Once the gene rankings have been established, the tool selects from the top of the list the minimum subset of genes required to identify each class. To achieve this, it uses a multiclass implementation [11] of Support Vector Machine (SVM), as a method that has been proven very efficient for classification of gene expression microarray datasets [12, 13, 14]. The SVM is integrated into a wrapper forward selection scheme to test whether a selected subset of genes is actually enough to discriminate the classes [15]. Several **SVM** classifiers are iteratively trained with an increasing number of genes taken from the ranked lists and evaluated through double nested cross-validation. The smallest subset of genes that provides the best performance is selected as feature set (**Figure 1C**) and used to train and build a final classifier that will include all the available samples of the training set.

The classifier built for a given set of compared diseases can be used to query and identify new unlabeled samples. In addition, the classifier is analysed in order to obtain the *discriminant power* of the selected genes (**Figure 1D**). Each gene's *discriminant power* is a quantitative parameter that resembles the value of such gene in class differentiation. Therefore, a high *discriminant power* (either positive or negative, in absolute value) indicates that the gene is useful to mark and identify samples from its assigned class. Full description of this parameter is provided in Methods section.

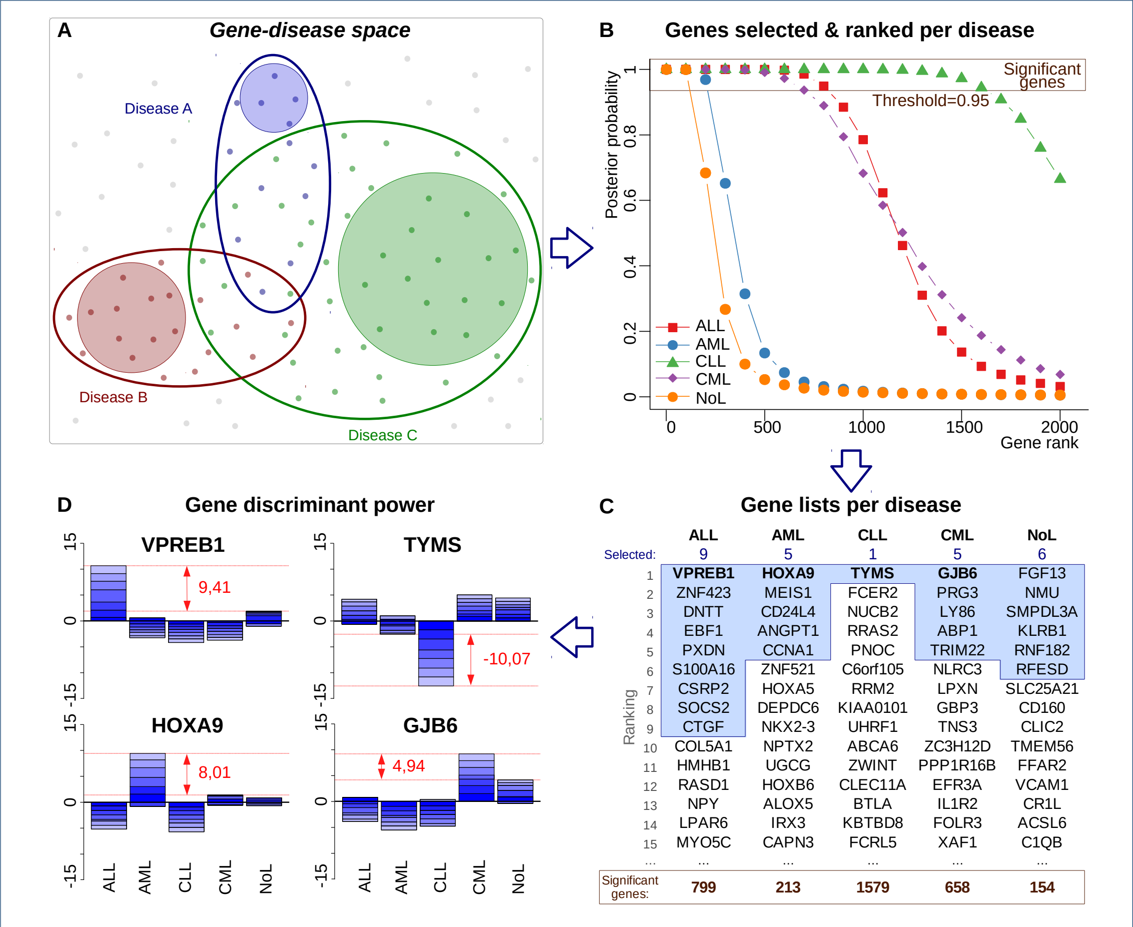
Building networks of genes associated to diseases

To infer possible associations between the genes assigned to each disease, *geNetClassifier* calculates gene-to-gene correlation and mutual information [16] in the expression dataset. This allows to identify possible relations of co-expression between the genes and possible relations of mutual redundancy. The detected associations are integrated in a network that also includes parameters derived from the differential expression analysis and from the classification analysis. Since networks are built for each class, they provide an integrative view of the gene sets associated to each disease in a relational characterized context. Examples of these networks are presented in the case studies in the following sections.

Using *geNetClassifier*: analysis of a leukemia dataset

We have applied *geNetClassifier* to a dataset of genome-wide expression microarrays of samples from leukemia, as a well known disease that allows to test the tool in a real case study and confirm the biological relevance of the results. This dataset includes 50 microarray samples from bone marrow of patients of four major leukemia subtypes (ALL, AML, CLL and CML; described in Methods) plus non-leukemia controls (NoL), making a total of 5 distinct classes.

The first result that *geNetClassiffier* provides is the set of rank-ordered lists of genes selected for each class, being the top genes the ones most significantly associated with each disease (as indicated in **Figure 1C**). The resulting lists of genes-per-disease do not overlap, in this way the method is optimized to find specific markers of each compared disease. The number of genes associated to each disease for a common threshold of significance is quite different from one class to another (e.g. 1027 genes for ALL but only 273 genes for AML). This observation seems to indicate that some diseases can affect more genes than others according



	ALL	AML	CLL	CML	NoL
Selected:	9	5	1	5	6
1	**VPREB1**	**HOXA9**	**TYMS**	**GJB6**	FGF13
2	ZNF423	MEIS1	FCER2	PRG3	NMU
3	DNTT	CD24L4	NUCB2	LY86	SMPDL3A
4	EBF1	ANGPT1	RRAS2	ABP1	KLRB1
5	PXDN	CCNA1	PNOC	TRIM22	RNF182
6	S100A16	ZNF521	C6orf105	NLRC3	RFESD
7	CSRP2	HOXA5	RRM2	LPXN	SLC25A21
8	SOCS2	DEPDC6	KIAA0101	GBP3	CD160
9	CTGF	NKX2-3	UHRF1	TNS3	CLIC2
10	COL5A1	NPTX2	ABCA6	ZC3H12D	TMEM56
11	HMHB1	UGCG	ZWINT	PPP1R16B	FFAR2
12	RASD1	HOXB6	CLEC11A	EFR3A	VCAM1
13	NPY	ALOX5	BTLA	IL1R2	CR1L
14	LPAR6	IRX3	KBTBD8	FOLR3	ACSL6
15	MYO5C	CAPN3	FCRL5	XAF1	C1QB
...	...	...	...	...	...
Significant genes:	799	213	1579	658	154

Figure 1 Description of geNetClassifier main features and results. (A) Scheme representing a gene-disease space for three hypothetical diseases. The color-line ovals would enclose the genes (dots) affected by a given disease. The coloured-background circles would mark the altered genes that are specific for a disease, which are the ones we aim to identify: i.e. genes affected by only one of the compared diseases. **(B)** Graph presenting the posterior probability of the top 2000 genes included in the gene ranking for each of the 4 leukemia subtypes and the non-leukemia samples. The genes are selected and ranked based on their posterior probability for each class. Genes with posterior probability over the threshold (> 0.95) can be considered *significant* candidates to mark each disease subtype. **(C)** Lists presenting the top-15 genes in the ranking of each class. The bottom row shows the total number of significant genes in the whole ranking. The shaded area contains the genes selected as the minimum subset to separate the classes. The number of genes selected per class is shown on top. **(D)** Discriminant power plot of the first ranked gene for each leukemia subtype: *VPREB1* (pre-B lymphocyte 1) for ALL; *HOXA9* (homeobox 9) for AML; *TYMS* (thymidylate synthetase) for CLL; and *GJB6* (gap junction beta-6 30kDa protein) for CML. The red numbers indicate the *discriminat power* values assigned to each gene.

to their comparative changes in the global expression profiles. These sizes do not represent the absolute number of genes each disease affects, but rather the genes that are only affected by each disease in the specific contrast. In any case, this phenomenological consideration supports the proposed hypothesis of a gene-disease space, where different diseases affect different number of genes.

After the classification process the minimun subset of genes that allow the best class separation were selected: 9 genes for ALL, 5 for AML, 1 for CLL, and 5 for CML (blue-shaded boxes in **Figure 1C**; detailed information about these genes is included in **Additional File 1**).

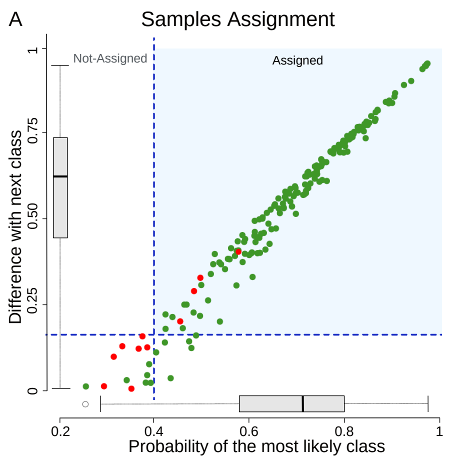
External validation and performance of *geNetClassifier*

Once the classifier for leukemias was built, an external validation was conducted to evaluate the accuracy and performance of the algorithm and to confirm the robustness of the genes selected as markers of the corresponding classes [17].

An external validation consists on querying the classification system with an independent set of samples whose class is *a priori* known. We used a different set of 200 samples of the same five classes (**Figure 2**). Sensitivity, specificity, MCC, global accuracy and global call rate were calculated to evaluate the performance. These statistical parameters were estimated in 10 runs of external validation randomly splitting the available samples.

The external validation could be performed following two different approaches: **(i)** assigning all the samples to their most likely class or (**ii**) leaving doubtful samples as *not-assigned*. (See Methods).

When the *not-assigned* option was selected, the external validation done with 200 leukemia samples provided an average of 4 misclassifications per run (shaded region in **Figure 2A**). All other samples were either correctly assigned or left unclassified (*not-assigned*), resulting in an average global accuracy of 98% and average call rate of 92% (assignment percentage). By contrast, since most samples that would have been incorrectly assigned had a probability under the thresh-



B

External Validation

Confusion matrices (Average percentage of 10 runs)

With Not-Assigned

	NotA	ALL	AML	CLL	CML	NoL
ALL	1	99	0	0	0	0
AML	23	0	70	0	1.25	5.75
CLL	0.50	0	0	99.50	0	0
CML	7.75	0	0.50	0	91.75	0
NoL	7.75	0	0	0	1.25	91

Sensitivity	100	90.9	100	99.5	98.6
Specificity	100	99.9	100	99.3	98.4
MCC	1.00	0.94	1.00	0.98	0.95
Call Rate	99	77	99.5	92.3	92.3

Global	
Accuracy	Call Rate
98.10	92.00

All Assigned

	NotA	ALL	AML	CLL	CML	NoL
ALL	0	99.50	0	0	0	0.50
AML	0	0.50	80.75	0.75	6.25	11.75
CLL	0	0	0	100	0	0
CML	0	0	2.25	0.25	96.75	0.75
NoL	0	0	0	0	2.75	97.25

Sensitivity	99.5	80.8	100	96.8	97.3
Specificity	99.9	99.4	99.8	97.8	96.8
MCC	0.99	0.86	0.99	0.93	0.91
Call Rate	100	100	100	100	100

Global	
Accuracy	Call Rate
94.85	100

Figure 2 External validation of geNetClassifier using a large dataset of leukemia samples. (A) Assignment of 200 leukemia samples in one run of external validation. Green dots correspond to samples in which the most likely class is correct and red dots to samples in which it is incorrect. Samples in the blue background area are assigned to the most likely class; samples under any of the two thresholds (dashed blue lines) are considered doubtful samples and can be left as "Not-Assigned". **(B)** Summary of the results of 10 independent external validations. Average confusion matrices (in percentage values) and statistical parameters: sensitivity, specificity, MCC, call rate (calculated per class), global accuracy and global call rate (calculated globally for all the classes). On the right (labelled *All Assigned*) the statistics are calculated assigning all the samples to their most likely class. On the left (*With Not-Assigned*) the statistics are calculated leaving doubtful samples unassigned.

olds (red dots in **Figure 2A**), the accuracy when all samples were forced to be assigned to their most likely class was 94.85%.

In overall, the external validation for the leukemias showed that the best performance –allowing not assignment– was obtained for ALL and CLL (100% sensitivity and specificity, MCC=1.0), while nk-AML presented the lowest values (90.9% sensitivity, 0.94 MCC and 77% call rate). Difficulties in the identification and classification of nk-AMLs were already described in a large-scale international leukemia study where the rate of misclassification for this specific subtype was 11.4% [18]. In conclusion, the classification accuracy rates provided by *geNetClassifier* confirms that the genes sets selected for each class can be good markers of the analysed disease subtypes.

Genes and networks associated to each leukemia subtype

The gene networks produced for each leukemia subtype are presented in **Figure 3**. The plots include the top-30 genes selected for each class as characteristic markers of each leukemia subtype.

Several of these genes have been already reported as functionally associated to these diseases. For example, in the case of ALL, the gene *VPREB1* -that is the first gene in ALL ranking- encodes a protein that belongs to the immunoglobulin superfamily and is expressed selectively at the early stages of B lymphocytes development (i.e. on the surface of pro-B and early pre-B cells). This gene has already been proposed as a useful marker for the detection of normal and malignant human pre-B lymphocytes [19]. Since all ALL samples included in this study correspond to pre-B-ALL without t(9;22), the selection of *VPREB1* seems quite adequate. Another gene selected to mark ALL is *DNTT*. The protein encoded by *DNTT* is expressed in a restricted population of normal and malignant pre-B and pre-T lymphocytes during early differentiation.

In the case of the genes selected for nk-AML, the network shows a cluster of homeobox genes (*HOXA4*, *HOXA5*, *HOXA7*, *HOXA9*, *HOXA10*). The co-expression of these genes detected in the dataset reveals that they are coregulated. *MEIS1* is a transcriptional regulator also included in the homeobox co-expression cluster and selected as one of the genes with best discriminant power for the nk-AML class. Two recent publications have reported that downregulation of *MEIS1* and *HOXA* genes impair proliferation and expansion of acute myeloid leukemia cells [20, 21]. Moreover, *HOXA* has a specific translocation event that has been associated with myeloid leukemogenesis, and overexpression of *HOXA9* has been shown as representitative of nk-AML patients during first diagnosis and if they suffer relapse [22]. These and other reports support the selection of *MEIS1* and *HOXA9* in the gene network that characterizes AML with normal karyotype [23]. Another gene related to AML is *ANGPT1*, that encodes protein angiopoietin 1. Angiopoietins are proteins with important roles in vascular development and angiogenesis which have also been identified as over expressed in bone marrow of AML patients [24].

Finally, the gene network produced for CML includes characteristic genes such as *PRG3*, that encodes for eosinophil major basic protein 2 (MBP2) which is specific of eosinophil granulocytes, a myeloid cell type. Moreover, it has been shown that many molecules essential for tumor cell growth (like polyamines) enter cells via a proteoglycan-dependent pathway that involves PRG3 [25]. All these published reports do not prove that the genes included in the networks for each leukemia subtype are essential for the development of such diseases. However, they give important support to the results and underline the value of the method for creating significant gene sets and gene networks associated to specific disease subtypes.

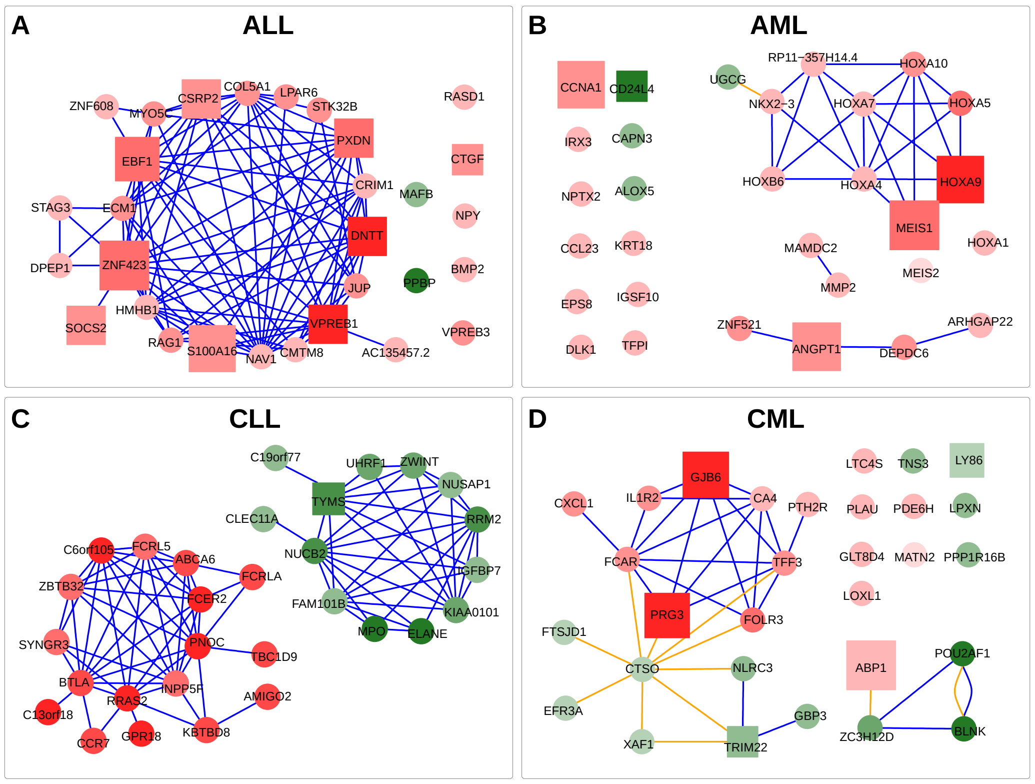


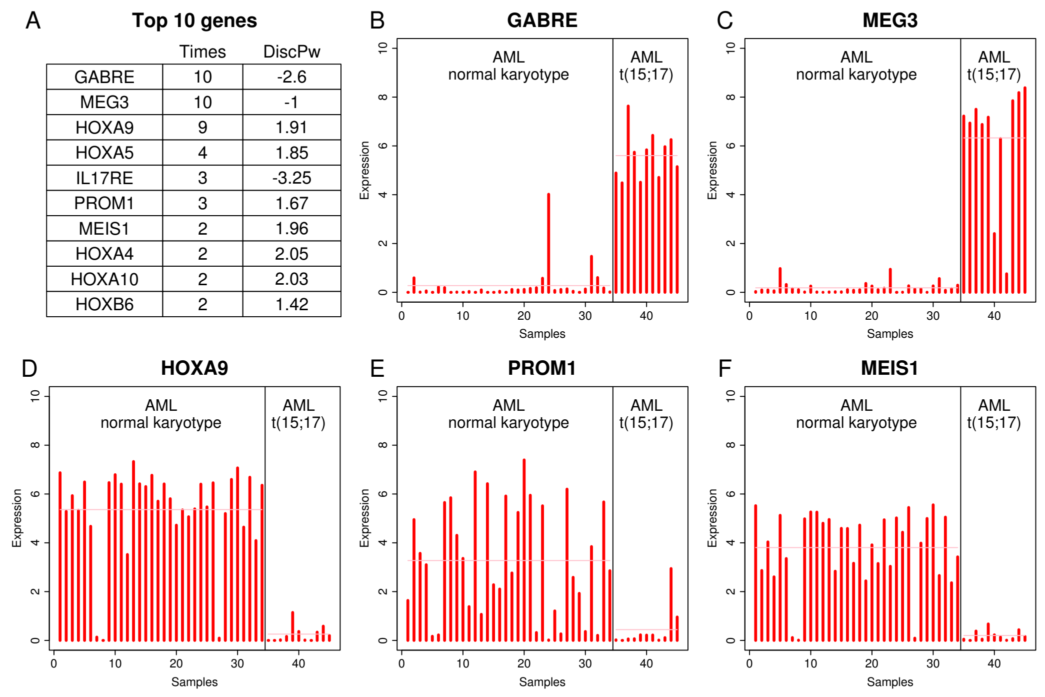
Figure 3 Gene networks produced for each studied leukemia subtype. The networks contain the top-30 ranked genes selected for each class. Genes are coloured in red if they are overexpressed and in green if they are repressed in the given subtype. The intensity of these colours indicate the magnitude of the observed expression difference. Squares correspond to genes that were selected as *minimum subset* to differentiate the classes. The size of these squares is proportional to the *discriminant power* of the gene. The color of the edges indicates the type of relations between the genes: blue meaning co-expression (correlation) and orange mutual information.

Application of *geNetClassifier* to an RNA-Seq dataset

geNetClassifier can be applied to different types of genomic data produced with different platforms. We have also applied it to an RNA-Seq dataset of acute leukemia samples [26] from which we selected 45 samples from patients with two AML subtypes: **(i)** 11 samples of patients with t(15;17) chromosomal translocation characteristic of acute promyelocytic leukemia (APL), and **(ii)** 34 samples of AML patients with normal karyotype and no detected FISH abnormalities (nk-AML). APL is an AML subtype that has good clinical prognosis. Its sensitivity to all-trans retinoic acid (ATRA) allows an efficient treatment unique among leukemias. By contrast, nk-AML is one of the most frequent subtypes of AML (approx. 50%) and usually has a poor clinical prognosis due to the lack of an efficient treatment [26]. Out of these two AML subtypes, nk-AML was also present in the previous microarray dataset analysed. This allows us to investigate the performance of the algorithm studying a common disease subtype in a different context and using a different type of expression data.

geNetClassifier was applied to the RNA-Seq dataset of APLs and nk-AMLs using 8 samples from each class as training samples and then validated with the rest of the samples. We repeated this process 10 times randomly selecting the training samples. The global accuracy obtained in this analysis was 100% with a call rate of 91.38%. The list of genes most frequently selected for classification (**Figure 4A**) included several homeobox genes (HOXA and HOXB) and MEIS1, showing agreement with the results obtained for nk-AML in the microarray analysis. In this way, the expression profiles from these genes in the RNA-Seq dataset are consistent with the results obtained from microarrays, e.g.: genes *HOXA9* and *MEIS1* were down regulated in APL in comparison to nk-AML (**Figure 4D and 4F**). In addition, the network generated for nk-AML selected a set of homeobox genes that form a highly connected co-expression cluster (**Figure 5**). Other genes detected in this analysis, for example MEG3 (**Figure 4C**), showed over-expression in APL versus nk-AML. In fact, it has been reported that MEG3 expression is lost in multiple cancer cell lines of various tissue origins, and it inhibits tumor cell proliferation in vitro. The identification of MEG3 as marker over-expressed in the AML subtype with better prognosis (**Figure 4C**) provides support to the selection of this gene as a discriminant feature between APL and nk-AML.

Finally, to have a better estimation of the global agreement provided by the algorithm in the analysis of the genes assigned to a given disease subtype, we analysed the total overlapping of the genes selected for nk-AML in the arrays dataset and the RNA-Seq dataset. Both platforms included a common set of 16,611 human protein-coding genes. Within this set, the number of significant genes selected for nk-AML were 202 (using posterior probability > 0.95). The RNA-Seq results included 95 of these genes (considering the 10



Top 10 genes

	Times	DiscPw
GABRE	10	-2.6
MEG3	10	-1
HOXA9	9	1.91
HOXA5	4	1.85
IL17RE	3	-3.25
PROM1	3	1.67
MEIS1	2	1.96
HOXA4	2	2.05
HOXA10	2	2.03
HOXB6	2	1.42

Figure 4 Genes selected for the RNA-seq dataset of acute leukemia. (A) The table shows the genes most frequently selected to distinguish between normal karyoptype AML (nk-AML) and AML with t(15;17) (APL). *geNetClassifier* was run ten times with different combinations of samples. The table includes the mean discriminant power within these runs and the number of times that each gene is included in the *minimum subset*. **(B-F)** Expression profiles corresponding to some of the top selected genes: *GABRE*, *MEG3*, *HOXA9*, *PROM1* and *MEIS1*. Each red bar corresponds to the expression signal ($log_2(RPKM + 1)$) in one sample.

runs indicated above), and 76 of them were selected in more than three runs. An overlap of 95 genes corresponds to an odds ratio of 2.17 and to an enrichment p-value < 0.000001 (using hypergeometric test). Therefore, it can be said that the consistency of the method to select genes that mark a specific disease subtype is high.

Comparison of *geNetClassifier* with other methods

Finally, we have evaluated the performance of *geNetClassifier* relative to other gene selection and classification methodologies. We compared *geNetClassifier* with four machine learning methods for feature selection using CMA package [27], which provides a comprehensive collection of various microarray-based classification algorithms (see **Additional File 2**). We have also evaluated the classification procedure of *geNetClassifier* using svb-IMPROVER contest platform [28], which includes a Diagnostic Signature Challenge with several datasets to assess and verify computational approaches that classify clinical samples based on transcriptomics data (see **Additional File 3**). In both cases, the performance of *geNetClassifier* algorithm is within the best methods. However, it should be noted that we could only compare the classification and gene selection procedures. The other features included in our package could not be found integrated in other methods.

Conclusions

Biological annotation of the genes selected and the networks built to mark and separate different pathological states confirm the value of using *geNetClassifier* to analyse multiple disease subtypes based on genome-wide expression profiles. The tool is provided open access in Bioconductor to facilitate the type of studies illustrated in this report.

As a general conclusion, the results using *geNetClassifier* showed a robust selection of gene markers for characterizing disease subtypes and allowed the construction of specific and weighted gene networks associated to each disease subtype. The method can be applied to data derived from different types of technologies (such as microarrays or RNA-Seq) and it is designed to analyse datasets with multiple categories of samples.

Methods

Implementation and availability

geNetClassifier has been developed as an R package following Bioconductor (BioC) standards and technical requisites (www.bioconductor.org). It has attained BioC package submission process and package guidelines to be included in BioC software release. It is freely available, open source and open access. The package includes help pages with usage examples for each specific function. Together with the package, we have written a *vignette* including a detailed tutorial to use the algorithm (**Additional File 4**).

Microarray dataset

The microarray leukemia dataset is a subset of 250 samples collected from the Microarray Innovations in Leukemia (MILE) study [18] available at Gene Expression Omnibus database (www.ncbi.nlm.nih.gov/geo/)

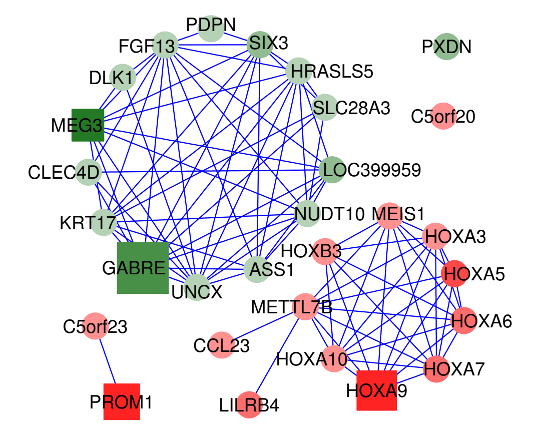


Figure 5 Gene network obtained for AML with the RNA-seq dataset. The network contains the top-30 ranked genes selected after running *geNetClasifier* to analyse the RNA-Seq expression data of normal karyoptype AML (nk-AML) versus AML with t(15;17) (APL) samples. The network shows two clear clusters: one including genes that are up-regulated in nk-AML and another with down-regulated genes. The red cluster includes many homeobox (HOX) genes highly correlated. These genes are characteristic of nk-AMLs and show good agreement with the results obtained with microarrays in spite of being two totally independent datasets.

under series accession number GSE13159. The genome-wide expression signal corresponding to these samples was measured using *Affymetrix* Human Genome U133 Plus 2.0 microarrays. The samples correspond to mononuclear cells isolated by Ficoll density centrifugation from bone marrow of untreated patients with: **(1)** *Acute Lymphoblastic Leukemia* (ALL) subtype *childhood* or *precursor B-cell* (c-ALL/pre-B-ALL) without translocation t(9;22); **(2)** *Acute Myeloid Leukemia* (AML) subtype *normal karyotype* (nk); **(3)** *Chronic Lymphocytic Leukemia* (CLL) subtype *B-cell*; **(4)** *Chronic Myeloid Leukemia* (CML); **(5)** *Non-leukemia* and healthy bone marrow (NoL).

The microarrays were normalized using the algorithm Robust Multi-Array Average (RMA) [29] and applying a gene-centric redefinition of the probes from the *Affymetrix* arrays to Ensembl genes (Ensembl IDs ENSG). This alternative Chip Definition File (CDF) with complete unambiguous mapping of microarray probes to genes is available at GATExplorer (http://bioinfow.dep.usal.es/xgate/) [30].

RNA-Seq dataset

The leukemia dataset analysed with RNA-sequencing corresponds to a subset of samples collected by the Cancer Genome Atlas (TCGA) [26] available at the TCGA data portal (https://tcga-data.nci.nih.gov/). These RNA-Seq data correspond to samples obtained from bone marrow aspirate of patients with AMLs of *de novo* diagnosis. Out of the available samples in TCGA, we selected 45 samples of the following subtypes: **(1)** AML patients with translocation t(15;17) (also called *Acute Promyelocytic Leukemia*, APL) (11 samples); and **(2)** AML patients with normal karyotype and no detected FISH abnormalities (nk-AML) (34 samples). The preprocessed RNA-Seq expression data matrices containing the *reads per kilobase per million mapped reads* (RPKM) were downloaded from the TCGA data portal and were log2 transformed ($log_2(RPKM+1)$) prior to be analysed with *geNetClassifier*.

Statistical methods and algorithm procedures

Gene ranking: To create the gene ranking, *geNetClassifier* uses the function *emfit*, a Parametric Empirical Bayes method, included in package *EBarrays* [9]. This method implements an expectation-maximization (EM) algorithm for gene expression mixture models, which compares the patterns of differential expression across multiple conditions and provides a *posterior probability*. The posterior probability is calculated for each gene-class pair with a *One-versus-Rest* contrast: comparing the samples of one class *versus* all the other samples. In this way, the posterior probability represents how much each gene differentiates a class from the other classes (being 1 the best value, and 0 the worst). The ranking is built, in a first step, by ordering the genes decreasingly by their posterior probability for each class. To resolve ties, the algorithm uses the value of the difference between the signal expression mean for each gene in the given class and the mean in the closest class. In a second step, the ranking procedure assigns each gene to the class in which it has the best ranking. As a result of this process, even if a gene is found associated to several classes during the expression analysis, it will only be on the ranking its best class. In addition, genes that do not show any significant difference between classes are filtered out before building the ranking. Finally, the set of genes considered *significant* in the ranking of each class is determined by a threshold of the posterior probability, which by default is set up to be greater than 0.95
.

Classifier: The classifier included in the algorithm is a multi-class *Support Vector Machine* (SVM) available in R package *e1071* [11]. This package provides a linear kernel implementation that allows the classification of multiple classes by using a *One-versus-One* (OvO) approach, in which all the binary classifications are fitted and the correct class is found based on a voting system.

Gene selection: The gene selection is done through a *wrapper forward* selection scheme based on *8-fold cross-validation*. Each cross-validation iteration starts with the first ranked gene of each class: it trains a temporary internal classifier with these genes, and evaluates its performance. One more gene is added in each step to those classes for which a 'perfect prediction' is not achieved (i.e. in case not all samples are correctly identified). The genes are taken in order from the *gene ranking* of each class until reaching zero error or the maximum number of genes allowed (determined by the arguments *maxGenesTrain* and *continueZeroError*). The error for each of the classifiers and the number of genes used to construct them are saved. Once the cross-validation loop is finished, it selects the minimum number of genes per class which produced the classifier with minimum error. To achieve the best stability in the number of selected genes, the cross-validation is repeated with new samplings as many times as indicated by the user (6 times by default). In each of these iterations, the minor number of genes that provided the smallest error is selected. The final selection is done based on the genes selected in each of the iterations. For each class, the top ranked genes are selected by taking the 'highest number' of genes selected in the cross-validaton iterations, but excluding possible 'outlier numbers' (i.e. selecting trimmed values).

Discriminant power: The *discriminant power* is a parameter calculated based on the *Lagrange coefficients* (alpha) of the *support vectors* for all the genes selected for the classification. Since the multi-class SVM algorithm is a *One-versus-One* implementation, it produces a set of *support vectors* for each binary comparison between classes. For each gene, the *Lagrange coefficients* of all the *support vectors* for each class are added up to give a value per class (represented as piled up bars in **Figure 1D**). The *discriminant power* is then calculated as the difference between the largest value and the closest one (i.e. the distance marked by two red lines in the plots in **Figure 1D**).

Assignment conditions: The whole tool *geNetClassifier* is built considering an *expert decision system* approach, because once the classifier is build it keeps open the possibility of '*do not assign*' when it is not sure about the class of a query sample. To make the assignment decision the probability to assign a sample to a given class should be at least double than the *random probability*, and the difference with the second most likely class should be higher than 0.8 times the *random probability*. If these conditions are not met, the sample is left as *Not-Assigned* (NA). These probability thresholds for assignment conditions are set up by default, but they can be changed by the user.

List of abbreviations used

MCC: Matthews Correlation Coefficient
t(9;22): translocation between chromosomes 9 and 22

Competing interests

The authors declare that they have no competing interests.

Author's contributions

SA carried out the development of the tool, performed its validation with several expression datasets and its statistical analysis, carried out the implementation of the R package and drafted the manuscript. CF participated in the design of the study, carried out the development of the algorithm and the trials of the methods included. CD participated in the application of the package. BR participated in the validation of the methods and helped in writing the manuscript. FJCL participated in the validation and application of the algorithm with different datasets. JMHR participated in the selection and analyses of the diseases and provided the clinical patient samples. JDLR conceived the study, directed the design and development of the algorithm and wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We acknowledge the funding and grants provided to Dr. J. De Las Rivas group by the Local Government, "Junta de Castilla y Leon" (JCyL, Valladolid, Spain, grants number CSI07A09 and BIO/SA68/13) and the Spanish Government, "Ministerio de Economia y Competitividad" (MINECO, ISCiii, Madrid, Spain, grants number PI09/00843 and PI12/00624). We also acknowledge PhD research grants given to SA, CR and BR by JCyL ("Ayudas a la Contratación de Personal Investigador") provided with the support of the European Social Fund (ESF). We also want to acknowledge Celgene-CITRE (citre@celgene.com) for the support provided to this research (Project Celegene-FICUS 2013). Since performing the work described, CF has become an employee of Celgene Research S.L., part of the Celgene Corporation.

Author details

[1]Cancer Research Center (IMBCC, CSIC/USAL/IBSAL), Campus Miguel de Unamuno s/n, Salamanca, 37007, Spain. [2] Hematology Department, Hospital Universitario de Salamanca (HUS/IBSAL/USAL), Paseo San Vicente 58-182, Salamanca, 37007, Spain. [3] (present address) Celgene Institute for Translational Research Europe (CITRE), Parque Científico y Tecnológico Cartuja 93, c/ Isaac Newton 4, Sevilla, 41092, Spain.

References

1. Culhane, A.C., Schröder, M.S., Sultana, R., Picard, S.C., Martinelli, E.N., Kelly, C., Haibe-Kains, B., Kapushesky, M., St Pierre, A.-A., Flahive, W., Picard, K.C., Gusenleitner, D., Papenhausen, G., O'Connor, N., Correll, M., Quackenbush, J.: GeneSigDB: a manually curated database and resource for analysis of gene expression signatures. Nucleic Acids Res **40**(Database issue), 1060–1066 (2012)
2. Venet, D., Dumont, J.E., Detours, V.: Most Random Gene Expression Signatures Are Significantly Associated with Breast Cancer Outcome. PLoS Comput Biol. **7**, 1002240 (2011)
3. De Ridder, D., De Ridder, J., Reinders, M.J.T.: Pattern recognition in bioinformatics. Brief Bioinform **14**, 633–647 (2013)
4. Larranaga, P.: Machine learning in bioinformatics. Brief Bioinform **7**, 86–112 (2006)
5. Cruz, J., Wishart, D.: Applications of machine learning in cancer prediction and prognosis. Cancer Inform **2**, 59–77 (2006)
6. De Las Rivas, J., Fontanillo, C.: Protein-protein interactions essentials: key concepts to building and analyzing interactome networks. PLoS Comput Biol **6**, 1000807 (2010)
7. Zhang, K., Pirooznia, M., Arabnia, H.R., Yang, J.Y., Wang, L., Luo, Z., Deng, Y.: Genomic signatures and gene networking: challenges and promises. BMC Genomics **12**(Suppl 5), 1 (2011)
8. in Man. OMIM, O.M.I.: http://omim.org
9. Yuan, M., Newton, M., Sarkar, D., Kendziorski, C.: EBarrays: Unified Approach for Simultaneous Gene Clustering and Differential Expression Identification. (2007). http://www.bioconductor.org/packages/release/bioc/html/EBarrays.html
10. Kendziorski, C.M., Newton, M.A., Lan, H., Gould, M.N.: On parametric empirical bayes methods for comparing multiple groups using replicated gene expression profiles. Stat Med **22**, 3899–3914 (2003)
11. Meyer, D.: Support Vector Machines. The Interface to Libsvm in Package E1071. (2001). http://cran.r-project.org/web/packages/e1071/
12. Statnikov, A., Aliferis, C., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics **21**, 631–643 (2005)
13. Statnikov, A., Wang, L., Aliferis, C.F.: A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics **9** (2008)
14. Pirooznia, M., Yang, J.Y., Yang, M.Q., Deng, Y.: A comparative study of different machine learning methods on microarray gene expression data. BMC Genomics **9** (2008)
15. Kohavi, A., Johnb, G.: Wrappers for feature subset selection. Artificial intelligence **97**, 273–324 (1997)
16. Meyer, P.E., Lafitte, F., Bontempi, G.: minet: A r/bioconductor package for inferring large transcriptional networks using mutual information. BMC Bioinformatics **9**, 461 (2008)
17. Ambroise, C., McLachlan, G.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proc Natl Acad Sci U S A **99**, 6562–6566 (2002)
18. Haferlach, T., Kohlmann, A., Wieczorek, L., Basso, G., Kronnie, G.T., Béné, M.-C., De Vos, J., Hernández, J.M., Hofmann, W.-K., Mills, K.I., Gilkes, A., Chiaretti, S., Shurtleff, S.a., Kipps, T.J., Rassenti, L.Z., Yeoh, A.E., Papenhausen, P.R., Liu, W.-M., Williams, P.M., Foà, R.: Clinical utility of microarray-based gene expression profiling in the diagnosis and subclassification of leukemia: report from the International Microarray Innovations in Leukemia Study Group. J Clin Oncol **28**, 2529–2537 (2010)
19. Bauer, S.R., Kudo, A., Melchers, F.: Structure and pre-B lymphocyte restricted expression of the VpreB gene in humans and conservation of its structure in other mammalian species. EMBO J **7**, 111–116 (1988)
20. Orlovsky, K., Kalinkovich, A., Rozovskaia, T., Shezen, E., Itkin, T., Alder, H., Ozer, H.G., Carramusa, L., Avigdor, A., Volinia, S., Buchberg, A., Mazo, A., Kollet, O., Largman, C., Croce, C.M., Nakamura, T., Lapidot, T., Canaani, E.: Down-regulation of homeobox genes MEIS1 and HOXA in MLL-rearranged acute leukemia impairs engraftment and reduces proliferation. Proc Natl Acad Sci USA **108**, 7956–7961 (2011)
21. Woolthuis, C., Han, L., Verkaik-Schakel, R., van Gosliga, D., Kluin, P., Vellenga, E., Schuringa, J., Huls, G.: Downregulation of MEIS1 impairs long-term expansion of CD34+ NPM1-mutated acute myeloid leukemia cells. Leukemia **26**, 848–853 (2012)
22. Grubach, L., Juhl-Christensen, C., Rethmeier, A., Olesen, L.H., Aggerholm, A., Hokland, P., Østergaard, M.: Gene expression profiling of polycomb, hox and meis genes in patients with acute myeloid leukaemia. Eur J Haematol **81**, 112–122 (2008)
23. Bullinger, L., Döhner, K., Bair, E., Fröhling, S., Schlenk, R.F., Tibshirani, R., Ph, D., Döhner, H., Pollack, J.R.: Use of Gene-Expression Profiling to Identify Prognostic Subclasses in Adult Acute Myeloid Leukemia. N Engl J Med **350**, 1605–1616 (2004)
24. Schliemann, C., Bieker, R., Padro, T., Kessler, T., Hintelmann, H., Buchner, T., Berdel, W., Mesters, R.: Expression of angiopoietins and their receptor tie2 in the bone marrow of patients with acute myeloid leukemia. Haematologica **91**, 1203–1211 (2006)
25. Mani, K., Sandgren, S., Lilja, J., Cheng, F., Svensson, K., Persson, L., Belting, M.: HIV-Tat protein transduction domain specifically attenuates growth of polyamine deprived tumor cells. Mol Cancer Ther **6**, 782–788 (2007)
26. Ley, T.J., Miller, C., Ding, L., Raphael, B.J., Mungall, A.J., *et al.*:

Genomic and epigenomic landscapes of adult de novo acute myeloid leukemia. N Engl J Med **368**, 2059–2074 (2013)
27. Slawski, M., Daumer, M., Boulesteix, A.: CMA: a comprehensive Bioconductor package for supervised classification with high dimensional data. BMC Bioinformatics **9**, 439 (2008)
28. Rhrissorrakrai, K., Rice, J.J., Boue, S., Talikka, M., Bilal, E., Martin, F., Meyer, P., Norel, R., Xiang, Y., Stolovitzky, G., Hoeng, J., Peitsch, M.C.: sbv IMPROVER Diagnostic Signature Challenge. Systems Biomedicine **1**, 196–207 (2013)
29. Irizarry, R.a., Hobbs, B., Collin, F., Beazer-Barclay, Y.D., Antonellis, K.J., Scherf, U., Speed, T.P.: Exploration, normalization, and summaries of high density oligonucleotide array probe level data. Biostatistics **4**, 249–264 (2003)
30. Risueño, A., Fontanillo, C., Dinger, M., De Las Rivas, J.: GATExplorer: genomic and transcriptomic explorer; mapping expression probes to gene loci, transcripts, exons and ncRNAs. BMC Bioinformatics **11**, 221 (2010)

Additional Files

Additional file 1: Table S1. Table with data and information about the genes selected by *geNetClassifier* in the analyses of the leukemia microarrays dataset (classes: four leukemia subtypes and control class NoL): **Class**: The category a gene has been assigned to. **Rank**: Position of the gene within the list of genes ranked by significance assigned to a disease. **Posterior probability**: Probability value given by the expectation-maximization algorithm to each gene. This value is used to establish the ranking. In this result all values were very close to 1 (with more than 10 significant digits). Ties are further ranked based on the differential expression. **Expression**: Difference between the mean expression of the gene within its class and the mean expression in the other classes. UP or DOWN indicates whether the gene is overexpressed or repressed in its class compared to the other classes. **Discriminant Power**: Parameter calculated based on the *Lagrange* coefficients of the support vectors of the classifier. Represents the weight that the classifier gives to each gene to differentiate a given class. **Redundancy**: If TRUE, the gene has a high correlation or mutual information with other genes in the list. The threshold to consider a gene redundant can be set through the arguments (by default: correlationsThreshold=0.8 and interactionsThreshold=0.5). **Chosen for classification**: Number of times the gene was chosen for classification (as part of the minimum required subset) in the 5 internal cross-validation loops. Rank mean and rank standard deviation (SD) of the gene in these classifiers. **Cross-validation**: Mean and standard deviation of the rank that the gene has obtained in *geNetClassifier*'s internal cross-validation, including the times it was not selected for classification.

Additional file 2: Table S2. Comparison of *geNetClassifier* gene selection procedure with four other machine learning methods for gene selection (i.e. feature selection): Limma, F-test, Boosting and Random Forest.
The comparison has been done on the dataset of 250 leukemia samples, using R/Bioc package CMA that provides a comprehensive collection of various microarray-based classification algorithms [27].

Additional file 3: File S3. Evaluation of the performance of *geNetClassifier* classification procedure in the sbv-IMPROVER contest platform (https://sbvimprover.com/), which includes a Diagnostic Signature Challenge to assess and verify computational approaches that classify clinical samples based on transcriptomics data [28].
The performance has been evaluated using the dataset from IMPROVER that includes four classes corresponding to lung cancer subtypes.

Additional file 4: File S4. *geNetClassifier vignette*
Vignette including a tutorial with executable examples and description of all the methods. This *vignette* is available in Bioconductor:
`http://www.bioconductor.org/packages/release/bioc/vignettes/geNetClassifier/inst/doc/geNetClassifier-vignette.pdf`

Additional file 1: File S1.

Class	Gene	Rank	Posterior Probability	Expression		Discriminant Power		Redundant	Chosen for classification			Cross-Validation	
				Mean difference	UP/DW	Value	Class		Times	Rank mean	Rank SD	Rank mean	Rank SD
ALL	**VPREB1**	1	1	6.33	UP	9.41	ALL	0	5	1	0	1	0
	ZNF423	2	1	5.09	UP	13.24	ALL	1	4	2.75	1.5	3	1.41
	DNTT	3	1	6.89	UP	8.97	ALL	1	5	2.8	0.45	2.8	0.45
	EBF1	4	1	5.41	UP	10.51	ALL	1	2	3	1.41	3.8	1.48
	PXDN	5	1	5.03	UP	8.65	ALL	1	2	5	1.41	5.2	0.84
	S100A16	6	1	4.34	UP	12.38	ALL	1	2	5.5	0.71	5.4	1.14
	CSRP2	7	1	4.04	UP	8.78	ALL	1	1	7	0	7.8	1.3
	SOCS2	8	1	4.53	UP	8.69	ALL	0	1	8	0	10.8	3.27
	CTGF	9	1	3.61	UP	5.55	ALL	0	0	NA	0	14.8	10.03
AML	**HOXA9**	1	1	4.43	UP	8.01	AML	0	5	1.2	0.45	1.2	0.45
	MEIS1	2	1	3.27	UP	10.31	AML	1	4	2.5	1	3	1.41
	CD24L4	3	1	-4.49	DOWN	-5.73	AML	0	2	3.5	3.54	3.8	2.17
	ANGPT1	4	1	2.74	UP	9.21	AML	0	2	4.5	0.71	4.8	1.3
	CCNA1	5	1	2.55	UP	8.24	AML	0	4	4	1.83	5.4	3.51
CLL	**TYMS**	1	1	-5.51	DOWN	-10.07	CLL	0	4	1.25	0.5	1.8	1.3
CML	**GJB6**	1	1	5.25	UP	4.94	CML	0	5	2.2	1.79	2.2	1.79
	PRG3	2	1	4.97	UP	4.09	CML	1	3	3	1	92.4	166.45
	LY86	3	1	-2.2	DOWN	-5.56	CML	0	1	2	0	39.6	21.9
	ABP1	4	1	2.51	UP	8.47	CML	0	4	3	2.16	5	4.85
	TRIM22	5	1	-2.67	DOWN	-9.05	CML	0	1	5	0	35.8	18.27
NOL	**FGF13**	1	1	2.69	UP	3.78	NoL	0	5	1.2	0.45	1.2	0.45
	NMU	2	1	1.96	UP	4.1	NoL	0	2	2.5	0.71	9	6.44
	SMPDL3A	3	1	1.95	UP	5.07	NoL	0	3	7	3.46	13.8	10.83
	KLRB1	4	1	2.23	UP	3.39	NoL	0	2	6.5	4.95	22.2	16.51
	RNF182	5	1	1.84	UP	1.06	NoL	0	3	2.33	1.53	5.6	4.72
	RFESD	6	1	2.36	UP	2.94	NoL	0	4	4.25	1.5	5.8	3.7

Additional file 2: File S2.

Comparison of five multi-class feature selection (i.e. "gene selection") methods. (Done using svmCMA function from CMA R package)

Data: expression microarrays from 250 leukemia samples. 50 samples are used for training (10 per class); 200 for testing as external validation (40 per class).

The classification is done always using SVM. To be comparable with the other methods, geNetClassifier is forced to assign all samples to a class.

SUM of the 10 confusion matrices for 10 runs: METHOD Boosting

10 runs	
Accuracy	**CallRate**
94.0	100
92.0	100
92.5	100
89.5	100
97.5	100
91.5	100
89.5	100
95.5	100
94.0	100
89.5	100

Always Assign	**ALL**	**AML**	**CLL**	**CML**	**NoL**
ALL	394	12	0	2	0
AML	4	332	0	4	4
CLL	0	4	400	0	0
CML	1	13	0	358	29
NoL	1	39	0	36	367

	Sensitivity	**Specificity**	**MCC**	**CallRate**
ALL	96.651	99.627	96.94	100
AML	96.744	95.948	87.179	100
CLL	99.024	100	99.384	100
CML	89.343	97.384	86.752	100
NoL	83.285	97.895	84.003	100

Global	
Accuracy	**CallRate**
92.55	100

SUM of the 10 confusion matrices for 10 runs: METHOD Limma

10 runs	
Accuracy	**CallRate**
92.0	100
95.5	100
99.5	100
96.0	100
95.5	100
92.5	100
95.0	100
93.5	100
91.5	100
97.0	100

Always Assign	**ALL**	**AML**	**CLL**	**CML**	**NoL**
ALL	395	0	0	0	0
AML	0	376	0	2	0
CLL	0	0	400	0	0
CML	1	4	0	358	33
NoL	4	20	0	40	367

	Sensitivity	**Specificity**	**MCC**	**CallRate**
ALL	100	99.692	99.215	100
AML	99.443	98.538	95.883	100
CLL	100	100	100	100
CML	91.087	97.420	87.726	100
NoL	85.708	97.940	85.585	100

Global	
Accuracy	**CallRate**
94.8	100

SUM of the 10 confusion matrices for 10 runs: METHOD Random Forest

10 runs	
Accuracy	**CallRate**
98.5	100
95.5	100
98.0	100
96.0	100
92.5	100
97.5	100
97.0	100
98.0	100
97.5	100
96.0	100

Always Assign	**ALL**	**AML**	**CLL**	**CML**	**NoL**
ALL	398	0	0	0	0
AML	0	376	0	2	1
CLL	0	2	400	0	0
CML	0	3	0	380	20
NoL	2	19	0	18	379

	Sensitivity	**Specificity**	**MCC**	**CallRate**
ALL	100	99.876	99.686	100
AML	99.232	98.528	95.752	100
CLL	99.512	100	99.692	100
CML	94.531	98.769	93.389	100
NoL	91.132	98.688	91.012	100

Global	
Accuracy	**CallRate**
96.65	100

SUM of the 10 confusion matrices for 10 runs: METHOD F-test

10 runs	
Accuracy	**CallRate**
96.0	100
96.5	100
96.0	100
96.0	100
94.5	100
92.5	100
94.5	100
92.5	100
93.5	100
97.0	100

Always Assign	**ALL**	**AML**	**CLL**	**CML**	**NoL**
ALL	395	1	0	0	0
AML	0	381	1	0	7
CLL	0	0	399	0	0
CML	0	4	0	351	21
NoL	5	14	0	49	372

	Sensitivity	**Specificity**	**MCC**	**CallRate**
ALL	99.756	99.691	99.061	100
AML	98.122	98.836	95.816	100
CLL	100	99.938	99.843	100
CML	93.528	97.006	88.291	100
NoL	84.880	98.215	85.842	100

Global	
Accuracy	**CallRate**
94.9	100

SUM of the 10 confusion matrices for 10 runs: METHOD geNetClassifier (for this comparison forced to "always assign")

10 runs	
Accuracy	**CallRate**
95.5	100
93.0	100
96.0	100
96.0	100
94.5	100
96.0	100
97.0	100
94.5	100
90.5	100
95.5	100

Always Assign	**ALL**	**AML**	**CLL**	**CML**	**NoL**
ALL	398	0	0	0	2
AML	2	323	3	25	47
CLL	0	0	400	0	0
CML	0	9	1	387	3
NoL	0	0	0	11	389

	Sensitivity	**Specificity**	**MCC**	**CallRate**
ALL	99.500	99.880	99.380	100
AML	80.750	99.440	86.200	100
CLL	100	99.750	99.380	100
CML	96.750	97.750	92.560	100
NoL	97.250	96.750	90.690	100

Global	
Accuracy	**CallRate**
94.85	100

Additional file 3: File S3.

Evaluation of the performance of ***geNetClassifier*** classification procedure in the **sbv-IMPROVER** contest platform (https://sbvimprover.com/challenge-1), which includes a **Diagnostic Signature Challenge** to assess and verify computational approaches that classify clinical samples based on transcriptomics data.

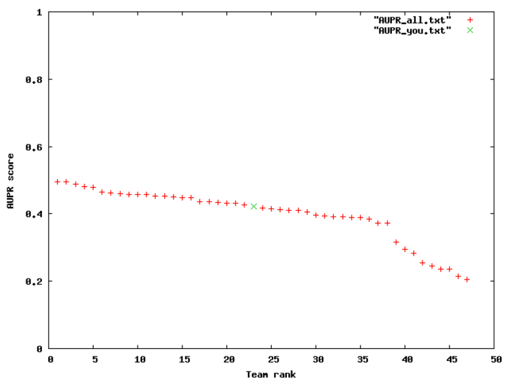
The performance of the algorithm ***geNetClassifier*** has been evaluated using a data-set that has multiple classes (a data-set of **lung cancer** included in **IMPROVER**). We show below the results corresponding to the performance measured with three parameters: **(i) AUPR**, that computes the precision-recall curve for each class, from which the Area Under the Precision-Recall curve is extracted (Precision is a measure of specificity whereas Recall is a measure of completeness); **(ii) BCM**, Belief Confusion Matrix, that is a matrix whose element {i,j} is the average confidence that a sample belonging to class i is in class j (Each prediction has its own belief confusion matrix. The perfect belief confusion matrix is the identity matrix); **(iii) CCEM**, Correct Class Enrichment Metric, that is computed adding the confidence of the samples whose classes were correctly predicted and subtract the confidence of the subjects whose classes were incorrectly predicted (In other words, this is a measure of enrichment of the correctly classified samples. The final value is normalized to be between 0 and 1).

These parameters indicate, as shown in the tables below, that ***geNetClassifier*** is within the best methods, performing as the **6th best** out of 47 different methods submitted to the **Diagnostic Signature Challenge** when it is applied using the option of "**not-assignment**"; and as the **7th best** in the rank of 47 methods when it is used forced to assign always a query sample to a class ("**all assigned**").

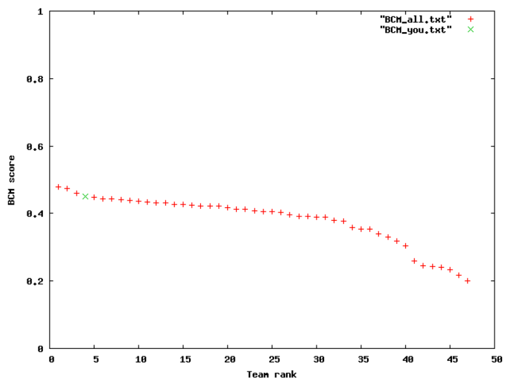
Plots that present the results of AUPR, BCM and CCEM corresponding to the performance of ***geNetClassifier*** using the option of "**not-assignment**".

The RESULTS TABLE placed after these plots presents the values of these parameters and the rank for the top-15 methods.

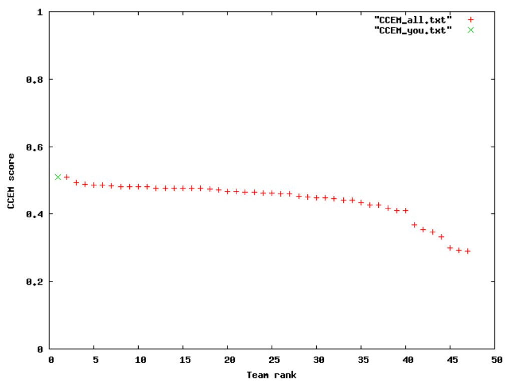
AUPR



BCM



CCEM



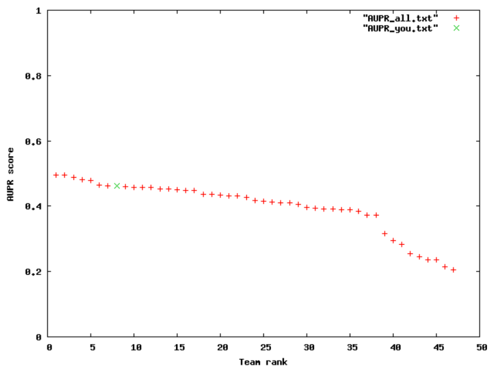
RESULTS TABLE

Team	AUPR	BCM	CCEM	Rank-sum	Rank
Team036	0.458	0.479	0.509	13	1
Team221	0.454	0.459	0.492	18	2
Team114	0.464	0.443	0.483	20	3
Team063	0.489	0.427	0.489	21	4
Team161	0.496	0.431	0.481	23	5
you	0.421	0.452	0.510	28	6
Team273	0.462	0.431	0.480	29	7
Team227	0.428	0.474	0.480	35	8
Team080	0.447	0.423	0.482	41	9
Team187	0.461	0.440	0.459	43	10
Team245	0.480	0.403	0.477	43	10
Team122	0.481	0.413	0.468	45	12
Team290	0.458	0.408	0.476	45	12
Team132	0.448	0.392	0.487	48	14
Team297	0.496	0.378	0.476	49	15

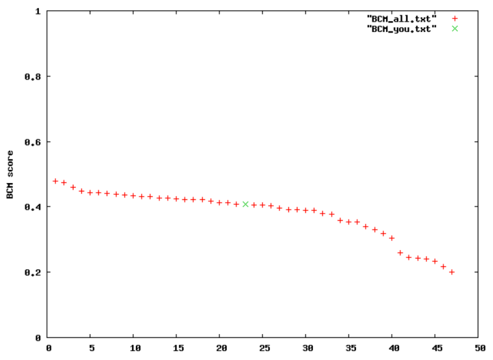
AUPR

Plots that present the results of AUPR, BCM and CCEM corresponding to the performance of ***geNetClassifier*** using the option of "**all assigned**".

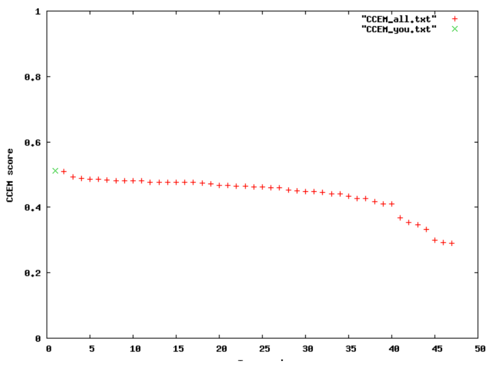
The RESULTS TABLE placed after these plots presents the values of these parameters and the rank for the top-15 methods.

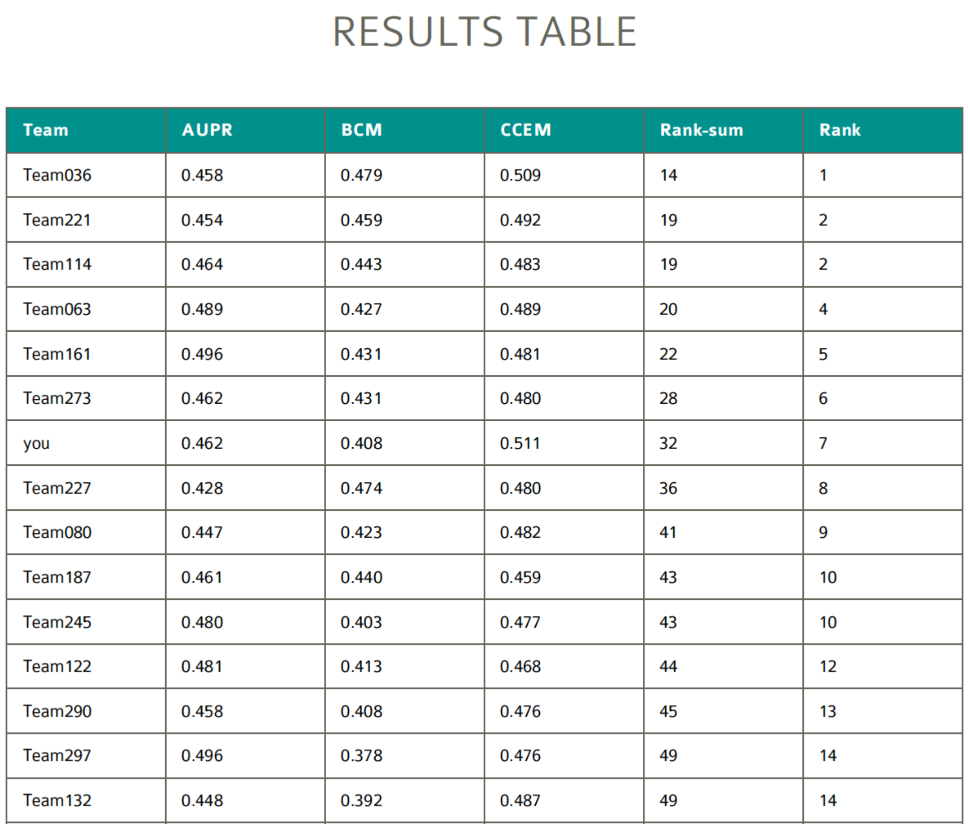


BCM



CCEM





RESULTS TABLE

Team	AUPR	BCM	CCEM	Rank-sum	Rank
Team036	0.458	0.479	0.509	14	1
Team221	0.454	0.459	0.492	19	2
Team114	0.464	0.443	0.483	19	2
Team063	0.489	0.427	0.489	20	4
Team161	0.496	0.431	0.481	22	5
Team273	0.462	0.431	0.480	28	6
you	0.462	0.408	0.511	32	7
Team227	0.428	0.474	0.480	36	8
Team080	0.447	0.423	0.482	41	9
Team187	0.461	0.440	0.459	43	10
Team245	0.480	0.403	0.477	43	10
Team122	0.481	0.413	0.468	44	12
Team290	0.458	0.408	0.476	45	13
Team297	0.496	0.378	0.476	49	14
Team132	0.448	0.392	0.487	49	14

geNetClassifier classify multiple diseases and build associated gene networks using gene expression profiles

Sara Aibar, Celia Fontanillo, Conrad Droste, and Javier De Las Rivas

Bioinformatics and Functional Genomics Group
Centro de Investigacion del Cancer (CiC-IBMCC, CSIC/USAL)
Salamanca - Spain

October 24, 2014

Version: 1.6

Contents

1 Introduction to *geNetClassifier*

geNetClassifier is an algorithm designed to build transparent classifiers and the associated gene networks based on genome-wide expression data.

geNetClassifier() is also the name of the main function in the package. This function takes as input the *expressionSet* or expression matrix of the studied samples and the classes the samples belong to (i.e. the diseases or disease subtypes). Once the data are analyzed, geNetClassifier() provides: **(i)** ranked gene sets (or gene signatures) that identify each class; **(ii)** a multiple-class classifier; and **(iii)** gene networks associated to each class.

- Gene ranking: The genes, probesets, or any other variables that are input in the *expressionSet* are considered *features* for the classification. These features are analyzed by *geNetClassifier*, and ranked according to the class they best identify, in order to select the optimum set for training the classifier. This ranking is returned by *geNetClassifier()* as well as the parameters calculated for gene selection.
- Classifier: *geNetClassifier()* also returns a multi-class SVM-based classifier, which can be queried later on; the genes (features) chosen for classification; their discriminant power (a parameter that measures the importance that the classifier internally gives to each gene); and, optionally, the classifier's generalization error and statistics about the selected genes.
- Network: The mutual-information (interactions) and the co-expression (correlations) between the genes are also calculated and analyzed by the algorithm. These allow to estimate the degree of association between the variables and they are used to generate a gene network for each class. These networks can be plotted, providing a integrated overview of the genes that characterized each disease (i.e. each class).

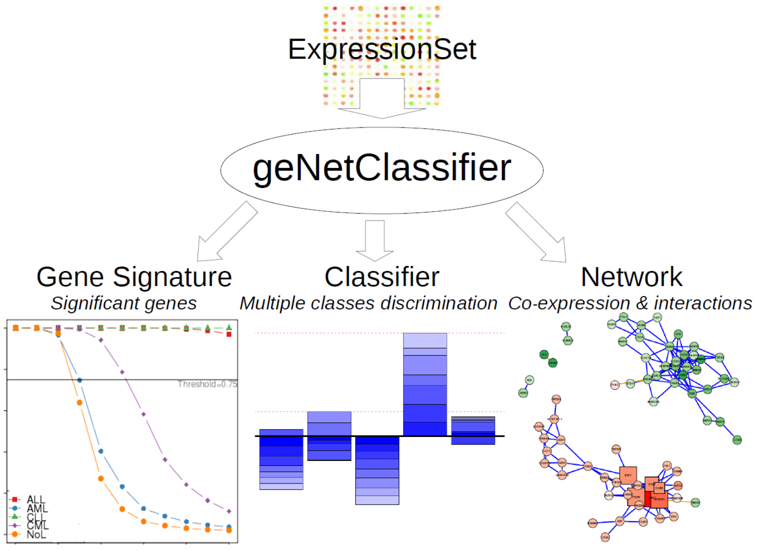


Figure 1. Taking an *expressionSet* as input, *geNetClassifier()* returns a gene signature for each class, a classifier to discriminate the classes, and gene networks associated to each class. The package also includes several analytic and visualizing tools to explore these results.

Examples of use

The algorithm shows a robust performance applied to patient-based gene expression datasets that study disease subtypes or disease classes. In this vignette, we show its performance for a leukemia dataset that includes 60 microarray samples from bone marrow of patients with four major leukemia subtypes (ALL, AML, CLL, CML) and no-leukemia controls (NoL). The results outperform a previously published classification analysis of these data [1].

The method is designed to be applied to the analysis and classification of different disease subtypes. Therefore, in the R package and this vignette, all the explanations and examples are disease-oriented. However, *geNetClassifier* can be applied to the classification of any other type of biological states, pathological or not.

Methods

The algorithm *geNetClassifier()* integrates several existing machine learning and statistical methods. The *feature ranking* is achieved based on a Parametric Empirical Bayes method (PEB). Double-nested internal cross-validation (CV) [2] is used for the *feature selection* process and to estimate the *generalization error* of the classifier. The machine learning method implemented in the classifier is a multi-class Support Vector Machine (SVM) [3]. The gene *networks* are built calculating the relations derived from gene to gene co-expression analysis (by default, *Pearson correlation*) and the interactions derived from gene mutual information analysis (using *minet* package) [4]. More details about these methods are available in the appropriate sections.

Queries

geNetClassifier includes a *query* function that allows either validation of the classifiers using external independent samples of known class (section 4) or classification of new samples whose class is unknown (section 5). This function facilitates the application of the classification algorithm as a predictor for new samples, and it is designed to resemble expert behavior by allowing *NotAssigned* (NA) instances when it is not sure about the class labelling. In order to assign a sample to a class, the algorithm requires a minimum certainty (i.e. probability), leaving it unassigned in case it does not achieve a clear call to a single class. These probability thresholds can be tuned to achieve a more or less stringent assignment. By following this procedure, the algorithm emulates human experts in the decision-making.

2 Install the package and example data

To install *geNetClassifier* from *Bioconductor*:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("geNetClassifier")
```

To follow the examples presented in this *Vignette*, we also need to install a sample dataset called *leukemiasEset*:

```
> biocLite("leukemiasEset")
```

This dataset contains an *expresssionSet* built with 60 gene expression microarrays (HG-U133 plus 2.0 from *Affymetrix*) hybridized with mRNA extracted from bone marrow biopsies of patients of the 4 major types of leukemia (ALL, AML, CLL and CML) and from non-leukemia controls (NoL). These data was produced by the Microarray Innovations in LEukemia (MILE) research project [1] and are available at GEO, under accession number GSE13159. The selected samples are labeled keeping their source GEO IDs.

To have an overview of this *ExpressionSet* and its available info:

```
> library(leukemiasEset)
> data(leukemiasEset)
> leukemiasEset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 20172 features, 60 samples
  element names: exprs, se.exprs
protocolData
  sampleNames: GSM330151.CEL GSM330153.CEL ... GSM331677.CEL (60 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: GSM330151.CEL GSM330153.CEL ... GSM331677.CEL (60 total)
  varLabels: Project Tissue ... Subtype (5 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: genemapperhgu133plus2

> summary(leukemiasEset$LeukemiaType)

ALL AML CLL CML NoL
 12  12  12  12  12

> pData(leukemiasEset)
```

For further information/help about this *ExpressionSet*:

```
> ?leukemiasEset
```

CEL files were preprocessed using an alternative Chip Description File (CDF), which allows mapping the expression directly to genes (Ensembl IDs ENSG) instead of *Affymetrix* probesets. This alternative CDF, with redefines gene-based annotation files for the *Affymetrix* expression microarrays, can be found in *GATExplorer* (a bioinformatic web platform that integrates a gene loci browser with nucleotide level re-mapping of the oligo *probes* from *Affymetrix* expression microarrays to genes: mRNAs and ncRNAs)[5].

To translate these Ensembl gene IDs into Gene Symbols for easier reading, the optional argument *geneLabels* from *geNetClassifier* can be used. This option allows to extend the annotation and labelling of the genes by providing a table that contains the gene symbol and other characteristics of the genes in the *expresssionSet*. This option can be used with any annotation (i.e. Bioconductor's *org.Hs.eg.db* package) as long as it is provided in the correct format. However, for increased consistency between versions, when using *GATExplorer* CDF, we recomend to also use *GATExplorer* annotation files. Annotation files with the Gene Symbol corresponding to each Ensembl gene ID can be found at: `http://bioinfow.dep.usal.es/xgate/mapping/mapping.php?content=annotationfiles`. The one used in this example is the *Human Genes* R annotation file. A subset of this file was saved into the object *geneSymbols* for easier use in the examples:

```
> data(geneSymbols)
> head(geneSymbols)
```

This annotation file provides further information which can be used to filter the genes. i.e. To consider only protein-coding genes for the construction of *geNetClassifier*, use the following filter:

```
> load("genes-human-annotation.R")
> leukEset_protCoding <- leukemiasEset[featureNames(leukemiasEset)
+ %in% rownames(genes.human.Annotation[genes.human.Annotation$biotype
+ %in% "protein_coding",]),]
> dim(leukemiasEset)
> dim(leukEset_protCoding)
```

Please note that *geNetClassifier* is designed to work with genes. In case the expression data is not summarized into genes (i.e. it uses the default *probesets*) *geNetClassifier* can still be used but those probesets/features will still be called *genes*.

3 Main function of the package: *geNetClassifier()*

geNetClassifier() is the main function of the package. It builds the classifier and the gene network associated to each class, and also returns the genes ranking and further information about the selected genes.

The workflow internally followed by *geNetClassifier()* includes the following steps:

1.- Filtering data and calculating the genes ranking.
2.- Calculating correlations between genes.
3.- Calculating interactions between genes.
Optional - Filter of redundant genes from the ranking (see arguments *removeCorrelations* and *removeInteractions*).

4.- Construction of the classifier: Selects of a subset of genes to train the classifier through 8-fold cross-validation. The selected genes are used to train the classifier with the complete set of samples.
5.- Estimation of performance: calculates the *generalization error* of the classifier and the statistics about the genes adding an 5-fold *cross-validation* around the construction of the classifier (*nested cross-validation*).
6.- Construction of the gene networks: a gene network is built for each one of the classes using the pairwise gene-to-gene correlations and interactions.
7.- Writing and saving the results including a series of plots for visualization.

The following sections show: how to load the package and the data (sec. 3.1); how to run the algorithm (sec. 3.2); an overview of the results and returned data (sec. 3.3): the genes ranking (sec. 3.4), the classifier (sec. 3.5) and the gene networks (sec. 3.6).

3.1 Loading the package and data

In order to have *geNetClassifier* functions available, the first step is to load the package:

```
> library(geNetClassifier)
```

To list all available tutorials for this package, or to open this *Vignette* you can use:

```
> # List available vignettes for package geNetClassifier:
> vignette(package="geNetClassifier")
> # Open vignette named "geNetClassifier-vignette":
> vignette("geNetClassifier-vignette")
```

To list all the available functions and objects included in *geNetClassifier* use the function *objects()*. Typing its name with a question mark (?) before any function, will show its help file. Through this tutorial, we will see how to use the main ones:

```
> objects("package:geNetClassifier")
```

```
 [1] "calculateGenesRanking"         "externalValidation.probMatrix"
 [3] "externalValidation.stats"      "gClasses"
 [5] "genesDetails"                  "geNetClassifier"
 [7] "getEdges"                      "getNodes"
 [9] "getNumEdges"                   "getNumNodes"
[11] "getRanking"                    "getSubNetwork"
[13] "getTopRanking"                 "initialize"
[15] "network2txt"                   "numGenes"
[17] "numSignificantGenes"           "overview"
[19] "plotAssignments"               "plotDiscriminantPower"
[21] "plotExpressionProfiles"        "plot.GenesNetwork"
[23] "plot.GenesRanking"             "plot.GeNetClassifierReturn"
[25] "plotNetwork"                   "queryGeNetClassifier"
[27] "querySummary"                  "setProperties"
[29] "show"
```

```
> ?geNetClassifier
```

After the package is loaded, you can proceed to analyze your data. In this vignette we use *leukemiasEset*: 60 microarrays from bone marrow from patients of the 4 major types of leukemia (ALL, AML, CLL, CML) and from healthy non-leukemia controls (NoL).(For installation and further information regarding *leukemiasEset* data package see Section 2).

```
> library(leukemiasEset)
> data(leukemiasEset)
```

In *leukemiasEset* there are 60 samples: 12 of each class (ALL, AML, CLL, CML and NoL). We will select 10 samples from each class to execute *geNetClassifier()*, and leave 2 for external validation of the resulting classifier. In this way, it makes a total of 50 samples for the *training* and 10 samples for the *validation*.

```
> trainSamples <- c(1:10, 13:22, 25:34, 37:46, 49:58)
> summary(leukemiasEset$LeukemiaType[trainSamples])

ALL AML CLL CML NoL
 10  10  10  10  10
```

3.2 Run *geNetClassifier()*

The essential input elements that *geNetClassifier* needs are:

1.- An *expressionSet*: R object defined in Bioconductor that contains a genome-wide expression matrix with data for multiple samples; see *?ExpressionSet*. Note that since the ranking is built though package *EBarrays*, the data in the expression set should be normalized intensity values (positive and on raw scale, not on a logarithmic scale).

2.- The *sampleLabels*: a vector with the class name of each sample or the *ExpressionSet phenoData* object containing this information. Note that to run *geNetClassifier* it is highly recommended to have the **same number of samples in each class**. A balanced number of samples allows an even exploration of each class and provides better classification.

The algorithm input also includes many other arguments that allow to personalize the execution or modify some of the parameters internally used. All of them have a default value and there is no need to modify them. In the following step we will see examples on how to use the main ones. Information about them can be found using the help options (i.e. *?geNetClassifier*). This is the full list of arguments with their default values:

```
geNetClassifier(eset, sampleLabels, plotsName=NULL, buildClassifier=TRUE,
estimateGError=FALSE, calculateNetwork=TRUE, labelsOrder=NULL,
geneLabels=NULL, numGenesNetworkPlot=100, minGenesTrain=1,
maxGenesTrain=100, continueZeroError=FALSE, numIters=6, lpThreshold=0.95,
numDecimals=3, removeCorrelations=FALSE, correlationsThreshold=0.8,
correlationMethod="pearson", removeInteractions=FALSE, interactionsThreshold=0.5,
skipInteractions=FALSE, minProbAssignCoeff=1, minDiffAssignCoeff=0.8,
IQRfilterPercentage=0, precalcGenesNetwork=NULL, precalcGenesRanking=NULL,
returnAllGenesRanking=TRUE, verbose=TRUE)
```

The execution time will depend on the computer and the size of the dataset. To avoid waiting now for the construction of a new classifier to continue this tutorial, a pre-executed example is included in the package:

```
> data(leukemiasClassifier)
```

This classifier was built running the following code:

```
> leukemiasClassifier <- geNetClassifier(leukEset_protCoding[,trainSamples],
+ sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
+ estimateGError=TRUE, geneLabels=geneSymbols)
```

These are some examples of standard use:

- The fastest execution would be training the classifier exploring a reduced number of genes (by default *maxGenesTrain=100*). In order ot skip calculating the network within the genes, set *calculateNetwork=FALSE*. However, since the correlations are relatively fast to calculate, we recommend keeping *calculateNetwork=TRUE*, and set *skipInteractions=TRUE* instead.

  ```
  > leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
  + sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
  + skipInteractions=TRUE, maxGenesTrain=20, geneLabels=geneSymbols)
  ```

- The default execution (*buildClassifier=TRUE*, *calculateNetwork=TRUE*) only requires the *expressionSet* and the *sampleLabels*. Providing *plotsName* is also recommended in order to produce the plots:

  ```
  > leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
  + sampleLabels="LeukemiaType", plotsName="leukemiasClassifier")
  ```

- In order to also estimate the classifier's performance, set *estimateGError=TRUE*. This option will take longer to execute

  ```
  > leukemiasClassifier <- geNetClassifier(eset=leukemiasEset[,trainSamples],
  + sampleLabels="LeukemiaType", plotsName="leukemiasClassifier",
  + estimateGError=TRUE)
  ```

Some of the parameters allow to provide extra information for an easier reading of the results:
- *labelsOrder* allows to show and plot the classes in a specific order (i.e. *labelsOrder=c('ALL', 'CLL', 'AML', 'CML', 'NoL')*)
- *geneLabels* can be used to add a label to the genes to show in the outputs instead of the *featureNames* from the *ExpressionSet*.
In the example, the genes were labeled with the gene symbols provided by *GATExplorer* gene-based probe mapping (*geneLabels=geneSymbols*), as it was indicated in section 3.1.

After running *geNetClassifier()*, we recommend to save the output:

```
> getwd()
> save(leukemiasClassifier, file="leukemiasClassifier.RData")
```

3.3 Overview of the data returned by *geNetClassifier()*

The main results that *leukemiasClassifier()* provides are: the **genes ranking** (sec. 3.4), the **classifier** (sec.3.5) and the **gene networks** (sec. 3.6). All this information is returned by *geNetClassifier()* in an object of class *GeNetClassifierReturn*. This objec tontains several slots which can be seen with the function names():

```
> names(leukemiasClassifier)

[1] "call"                 "classifier"
[3] "classificationGenes" "generalizationError"
[5] "genesRanking"         "genesRankingType"
[7] "genesNetwork"         "genesNetworkType"
```

The slot **@call** contains the R sentence that was used to execute *geNetClassifier()*. It is the only slot that will always be returned by *geNetClassifier()*, the presence and contents of the other components returned by the algorithm will depend on the arguments used to run it.

```
> leukemiasClassifier@call

geNetClassifier(eset = leukEset_protCoding[, trainSamples], sampleLabels = "LeukemiaT
    plotsName = "leukemiasClassifier", buildClassifier = TRUE,
    estimateGError = TRUE, calculateNetwork = TRUE, geneLabels = geneSymbols)
```

All the outputs and returned components are explained in detail in the following sections:

- @**genesRanking** in section 3.4
- @**classifier** and @**classificationGenes** in section 3.5
- @**generalizationError** in section 3.5.2
- @**genesNetwork** in section 3.6
- The **plots** are explained in section 6

A general view of the output can be seen by just typing the assigned name:

```
> leukemiasClassifier

R object summary:
Classifier trained with 50 samples.
Total number of genes included in the classifier: 26.
Number of genes per class:
ALL AML CLL CML NoL
  9   5   1   5   6
For classificationGenes details: genesDetails(EXAMPLE@classificationGenes)

Generalization error and gene stats calculated through 5-fold cross-validation:
[1] "accuracy"                 "sensitivitySpecificity"
[3] "confMatrix"               "probMatrix"
[5] "querySummary"             "classificationGenes.stats"
```

```
[7] "classificationGenes.num"

The ranking of all genes contains (genes per class):
 ALL  AML  CLL  CML  NoL
2342 3023 2824 2539 3049

The networks calculated for the topGenes genes of each class contain:
                     ALL AML   CLL  CML  NoL
Number of genes     1027 400  1916  949  400
Number of relations 1942 296 18506 6540 1993

Available slots in this R object:
[1] "call"                "classifier"          "classificationGenes"
[4] "generalizationError" "genesRanking"        "genesRankingType"
[7] "genesNetwork"        "genesNetworkType"
To see an overview of all available slots type "overview(EXAMPLE)"
```

3.4 Return I: Genes ranking

The first step of *geNetClassifier* algorithm is to determine a ranking of genes for each class based in the analysis of the expression signal. To create this ranking, it uses the function *emfit*, a Parametric Empirical Bayes method [6], included in package *EBarrays* [7]. This method implements an expectation-maximization (EM) algorithm for gene expression mixture models, which compares the patterns of differential expression across multiple conditions and provides a *posterior probability*.

The posterior probability is calculated for each gene-class pair, and represents how much each gene differentiates a class from the other classes; being 1 the best value, and 0 the worst. In this way, the posterior probability allows to find the genes that show significant differential expression when comparing the samples of one class *versus* all the other samples (One-versus-Rest comparison).

A first version of the ranking is built by ordering the genes decreasingly by their posterior probability for each class. To resolve the ties, *geNetClassifier* uses the expression difference between the mean for each gene in the given class and the mean in the closest class. In addition, the genes with a posterior probability greater or equal to 0.95 for the 'no difference' -the genes that do not show any difference between classes- are filtered out before proceeding into further steps.

The final version of the ranking is built assigning each gene to the class in which it has the best ranking. In this way the separation between classes is optimized, and the method will choose first the genes that best differentiate any of the classes. As a result of this process, even if a gene is found associated to several classes during the expression analysis, **each gene can only be on the ranking of one class**.

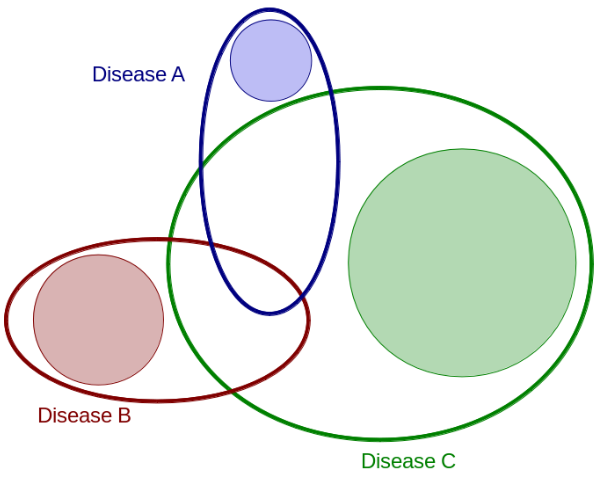


Figure 2. Scheme representing the overlap between the sets of genes that each disease may affect. *geNetClassifier* explores all the genes that affect each disease (**ovals**) and selects as significant, the genes that are unique (differentially expressed) to each disease (**coloured circles**).

The genes ranking obtained for each class is used for the gene selection in the classification procedure and it is also provided as an output of *geNetClassifier()* in the slot: ...@genesRanking.

```
> leukemiasClassifier@genesRanking

Top ranked genes for the classes:  ALL AML CLL CML NoL
      ALL       AML      CLL        CML      NoL
 [1,] "VPREB1"  "HOXA9"  "TYMS"     "GJB6"   "FGF13"
 [2,] "ZNF423"  "MEIS1"  "FCER2"    "PRG3"   "NMU"
 [3,] "DNTT"    "CD24L4" "NUCB2"    "LY86"   "SMPDL3A"
 [4,] "EBF1"    "ANGPT1" "RRAS2"    "ABP1"   "KLRB1"
 [5,] "PXDN"    "CCNA1"  "PNOC"     "TRIM22" "RNF182"
 [6,] "S100A16" "ZNF521" "C6orf105" "NLRC3"  "RFESD"
 [7,] "CSRP2"   "HOXA5"  "RRM2"     "LPXN"   "SLC25A21"
 [8,] "SOCS2"   "DEPDC6" "KIAA0101" "GBP3"   "CD160"
 [9,] "CTGF"    "NKX2-3" "UHRF1"    "TNS3"   "CLIC2"
[10,] "COL5A1"  "NPTX2"  "ABCA6"    "ZC3H12D" "TMEM56"
...

Number of ranked significant genes (posterior probability over threshold):
         ALL AML CLL CML NoL
         1027 273 1916 949 191
To see the whole ranking (3049 rows) use: getRanking(...)
Details of the top X ranked genes of each class: genesDetails(..., nGenes=X)
```

This ranking an object of class *GenesRanking*. This class provides some utility functions which will help working with the information contained in the object. The total number of genes in the ranking for each class can be queried using the function *numGenes()*. These numbers include all the genes that have some ability to distinguish between classes, although only the top ones are really significant.

```
> numGenes(leukemiasClassifier@genesRanking)

 ALL  AML  CLL  CML  NoL
2342 3023 2824 2539 3049
```

With *getTopRanking()* a subset of the ranking containing only the given number of top genes can be obtained. Since the returned object is also a *GenesRanking* object, no information is lost and other functions (i.e. *genesDetails()*) can be used afterwards.

```
> subRanking <- getTopRanking(leukemiasClassifier@genesRanking, 10)
```

In order to retrieve the whole ranking in the form of a matrix (i.e. to print the full version or get a subset of it), the function *getRanking()* can be used. This function provides the option to show the ranking with the gene IDs or the gene Labels.

```
> getRanking(subRanking)

$geneLabels
      ALL       AML      CLL        CML       NoL
 [1,] "VPREB1"  "HOXA9"  "TYMS"     "GJB6"    "FGF13"
 [2,] "ZNF423"  "MEIS1"  "FCER2"    "PRG3"    "NMU"
 [3,] "DNTT"    "CD24L4" "NUCB2"    "LY86"    "SMPDL3A"
 [4,] "EBF1"    "ANGPT1" "RRAS2"    "ABP1"    "KLRB1"
 [5,] "PXDN"    "CCNA1"  "PNOC"     "TRIM22"  "RNF182"
 [6,] "S100A16" "ZNF521" "C6orf105" "NLRC3"   "RFESD"
 [7,] "CSRP2"   "HOXA5"  "RRM2"     "LPXN"    "SLC25A21"
 [8,] "SOCS2"   "DEPDC6" "KIAA0101" "GBP3"    "CD160"
 [9,] "CTGF"    "NKX2-3" "UHRF1"    "TNS3"    "CLIC2"
[10,] "COL5A1"  "NPTX2"  "ABCA6"    "ZC3H12D" "TMEM56"
```

```
> getRanking(subRanking, showGeneID=TRUE)$geneID[,1:4]

      ALL               AML               CLL               CML
 [1,] "ENSG00000169575" "ENSG00000078399" "ENSG00000176890" "ENSG00000121742"
 [2,] "ENSG00000102935" "ENSG00000143995" "ENSG00000104921" "ENSG00000156575"
 [3,] "ENSG00000107447" "ENSG00000185275" "ENSG00000070081" "ENSG00000112799"
 [4,] "ENSG00000164330" "ENSG00000154188" "ENSG00000133818" "ENSG00000002726"
 [5,] "ENSG00000130508" "ENSG00000133101" "ENSG00000168081" "ENSG00000132274"
 [6,] "ENSG00000188643" "ENSG00000198795" "ENSG00000111863" "ENSG00000167984"
 [7,] "ENSG00000175183" "ENSG00000106004" "ENSG00000171848" "ENSG00000110031"
 [8,] "ENSG00000120833" "ENSG00000155792" "ENSG00000166803" "ENSG00000117226"
 [9,] "ENSG00000118523" "ENSG00000119919" "ENSG00000034063" "ENSG00000136205"
[10,] "ENSG00000130635" "ENSG00000106236" "ENSG00000154262" "ENSG00000178199"
```

The function *genesDetails()* allows to show all the available info of the genes in the ranking.

```
> genesDetails(subRanking)$AML

                GeneName ranking class postProb exprsMeanDiff exprsUpDw
ENSG00000078399    HOXA9       1   AML        1        4.4362        UP
ENSG00000143995    MEIS1       2   AML        1        3.2785        UP
ENSG00000185275   CD24L4       3   AML        1       -4.4926      DOWN
ENSG00000154188   ANGPT1       4   AML        1        2.7427        UP
ENSG00000133101    CCNA1       5   AML        1        2.5558        UP
ENSG00000198795   ZNF521       6   AML        1        2.5697        UP
ENSG00000106004    HOXA5       7   AML        1        3.1729        UP
ENSG00000155792   DEPDC6       8   AML        1        2.4803        UP
ENSG00000119919   NKX2-3       9   AML        1        2.1962        UP
ENSG00000106236    NPTX2      10   AML        1        2.0582        UP
                isRedundant
ENSG00000078399       FALSE
ENSG00000143995        TRUE
ENSG00000185275       FALSE
ENSG00000154188       FALSE
ENSG00000133101       FALSE
ENSG00000198795        TRUE
ENSG00000106004        TRUE
ENSG00000155792        TRUE
ENSG00000119919       FALSE
ENSG00000106236       FALSE
```

NOTE: If the console splits the table into several lines, try:

```
> options(width=200)
```

By default, the *rownames* are the ID included in the *expressionSet*: in our case the ENSEMBL IDs. The *GeneName* column has been added by setting the argument *geneLabels=geneSymbols* (see sec. 3.2).

To see the description of the content of this table write: *?genesDetails*.

More details about *GenesRanking* class is available at: *?GenesRanking*.

3.4.1 Significant genes

The set of genes considered *significant* for each of the classes is determined by a common threshold for the posterior probability (by default *lpThreshold=0.95*). This common threshold provides a way to quantify the size of the *gene signature* assigned to each disease (as always: compared to the other diseases in the study). In this way, the algorithm provides a framework to compare biological states, i.e. the biological or pathological conditions represented in the samples.

plotSignificantGenes() provides a plot of the distribution of the posterior probabilities of the genes within the rankings for each class:

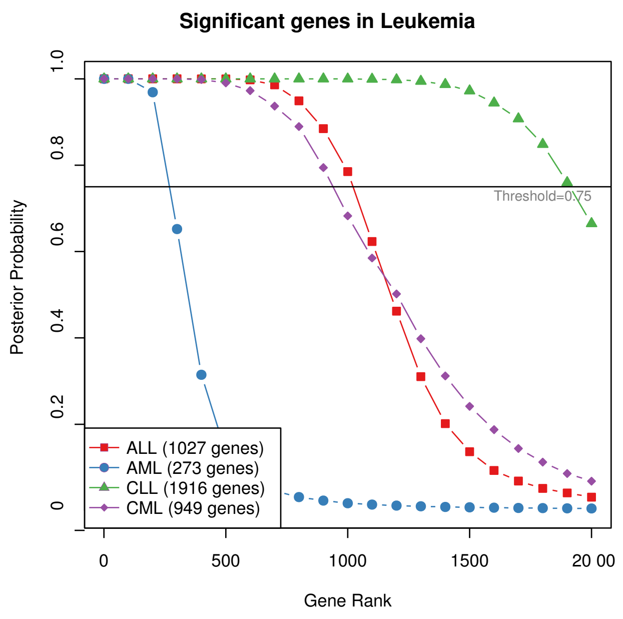


Figure 3. Plot of the posterior probabilities of the genes of 4 leukemia classes, ordering the genes according to their rank.

This example shows the big differences in size of the gene sets assigned to a disease: at lpThreshold 0.95 CLL has been assigned 2028 genes, while AML only 308 genes. The biological interpretation of this observation will depend on the specific study. Larger gene signatures may be an indication of more *systemic* diseases (i.e. a disease affect more genes than another), but it may also be an indication of the relative differences between the diseases in the study (i.e. one of the diseases affects different genes than the others). In any case, the results provided by *geNetClassifier* may help to unravel disease sub-types differences based on the gene signatures.

numSignificantGenes() provides the number of significant genes, the number of genes with posterior probability over the threshold:

```
> numSignificantGenes(leukemiasClassifier@genesRanking)
```

```
 ALL  AML  CLL  CML  NoL
1027  273 1916  949  191
```

The plot of the posterior probability (*plotSignificantGenes()*) is the default plot for objects of class *GenesRanking*. (More details in section 6.1).

```
> plot(leukemiasClassifier@genesRanking)
```

In both functions, the threshold can be modified through *lpThreshold*:

```
> plot(leukemiasClassifier@genesRanking,
+ numGenesPlot=3000, lpThreshold=0.80)
```

3.5 Return II: Classifier

The information regarding the classifier is saved into the slots: @classifier, @classifcation-Genes and @generalizationError.

The ***@classifier*** slot contains the SVM classifier that can later be used to make queries. The SVM method included in the algorithm is a linear kernel implementation from R package *e1071*. This implementation allows multi-class classification by using a One-versus-One (OvO) approach, in which all the binary classifications are fitted and the correct class is found based on a voting system.

```
> leukemiasClassifier@classifier

$SVMclassifier

Call:
svm.default(x = t(esetFilteredDataFrame[buildGenesVector, trainSamples]),
    y = sampleLabels[trainSamples], kernel = "linear", probability = T,
    C = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.03846154

Number of Support Vectors:  29
```

@classificationGenes contains the final genes selected to build the classifier. Since *@classificationGenes* is an object of class *GenesRanking*, functions such as *numGenes()* or *genesDetails()* can be used to explore it.

```
> leukemiasClassifier@classificationGenes

Top ranked genes for the classes:  ALL AML CLL CML NoL
      ALL       AML      CLL    CML      NoL
 [1,] "VPREB1"  "HOXA9"  "TYMS" "GJB6"   "FGF13"
 [2,] "ZNF423"  "MEIS1"  NA     "PRG3"   "NMU"
 [3,] "DNTT"    "CD24L4" NA     "LY86"   "SMPDL3A"
 [4,] "EBF1"    "ANGPT1" NA     "ABP1"   "KLRB1"
 [5,] "PXDN"    "CCNA1"  NA     "TRIM22" "RNF182"
 [6,] "S100A16" NA       NA     NA       "RFESD"
 [7,] "CSRP2"   NA       NA     NA       NA
 [8,] "SOCS2"   NA       NA     NA       NA
 [9,] "CTGF"    NA       NA     NA       NA

Details of the top X ranked genes of each class: genesDetails(..., nGenes=X)
```

```
> numGenes(leukemiasClassifier@classificationGenes)

ALL AML CLL CML NoL
  9   5   1   5   6

> genesDetails(leukemiasClassifier@classificationGenes)$ALL

                GeneName ranking gERankMean class postProb exprsMeanDiff
ENSG00000169575   VPREB1       1        1.0   ALL        1        6.3307
ENSG00000102935   ZNF423       2        3.0   ALL        1        5.0980
ENSG00000107447     DNTT       3        2.8   ALL        1        6.8948
ENSG00000164330     EBF1       4        3.8   ALL        1        5.4171
ENSG00000130508     PXDN       5        5.2   ALL        1        5.0387
ENSG00000188643  S100A16       6        5.4   ALL        1        4.3434
ENSG00000175183    CSRP2       7        7.8   ALL        1        4.0479
ENSG00000120833    SOCS2       8       10.8   ALL        1        4.5383
ENSG00000118523     CTGF       9       14.8   ALL        1        3.6167
                exprsUpDw discriminantPower discrPwClass isRedundant
ENSG00000169575        UP          9.416945          ALL       FALSE
ENSG00000102935        UP         13.240579          ALL        TRUE
ENSG00000107447        UP          8.978735          ALL        TRUE
ENSG00000164330        UP         10.515557          ALL        TRUE
ENSG00000130508        UP          8.657167          ALL        TRUE
ENSG00000188643        UP         12.385161          ALL        TRUE
ENSG00000175183        UP          8.782649          ALL        TRUE
ENSG00000120833        UP          8.697958          ALL       FALSE
ENSG00000118523        UP          5.551344          ALL       FALSE
```

Note that besides the common information about the genes provided by the genes ranking (sec. 3.4), the classification genes also have information about the **discriminant power** of the genes (sec. 6.3).

For details on the *gene selection procedure* (sec. 3.5.1) and the *estimation of performance and generalization error procedure* (slot @***generalization***) (sec. 3.5.2), see the next two sections.

3.5.1 Gene selection procedure

The optimum number of genes to train the classifier is selected by evaluating the classifiers trained with increasing number of genes. This is done using several iterations of 8-fold cross-validation. Each cross-validation iteration starts with the first ranked gene of each class: it trains an internal classifier with these genes, and evaluates its performance.

One more gene is added in each step to those classes for which a 'perfect prediction' is not achieved (i.e. not all samples correctly identified). The genes are taken in order from the *genes ranking* of each class until any of the classes reaches gets to the maximum number of genes (*maxGenesTrain=100*) or until zero error is reached (*continueZeroError=FALSE*). The error for each of the classifiers and the number of genes used to construct them are saved. Once the cross-validation loop is finished, it saves the minimum number of genes per class which produced the classifier with minimum error.

To achieve the best stability in the number of selected genes, the cross-validation is not run just once, but it is repeated several times with new samplings. This process is repeated as many times as indicated by the optional parameter *numIters* (6 by default). In each of these iterations, the minor number of genes that provided the smallest error is selected.

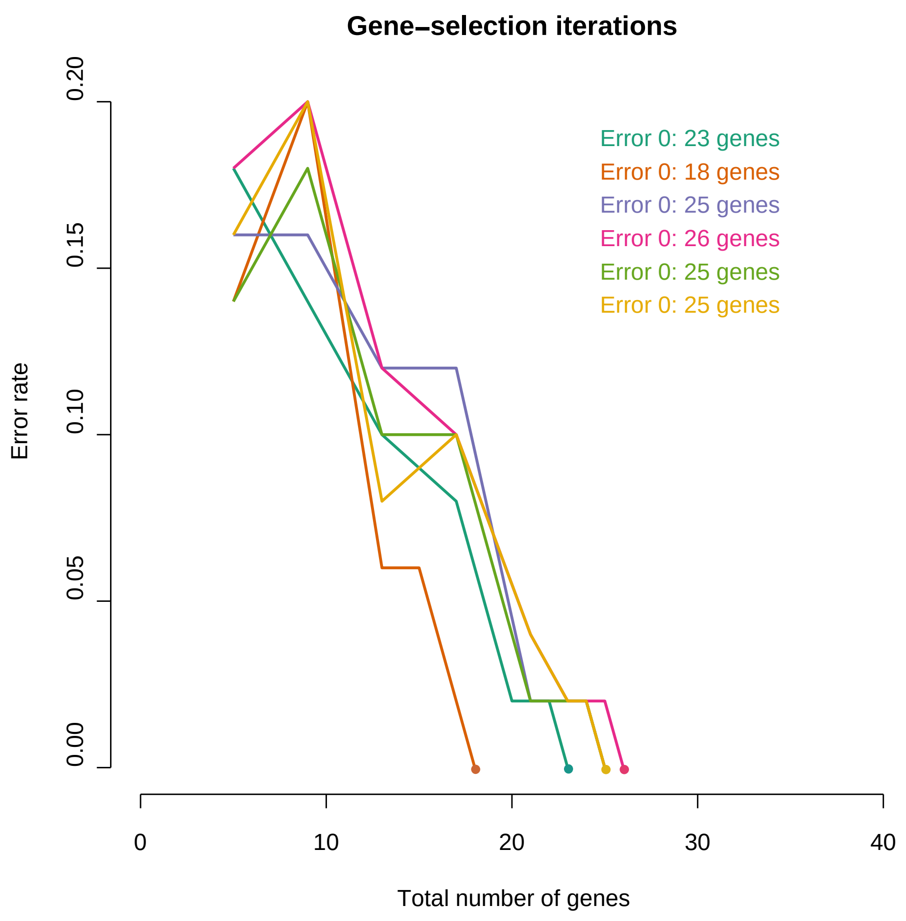


Figure 4. Plot of the gene-selection iterations. Each line represents an iteration and the error rates observed for each number of genes (starting at 5, one per class). The algorithm runs until exploring a maximum number of genes in any class (*maxGeneTrain=100*) or until zero error is reached (*continueZeroError=FALSE*). In each iteration the minimum number of genes with minimum error is selected.

The final selection is done based on the genes selected in each of the iterations. For each class, the top ranked genes are selected by taking the highest number of genes –excluding outliers– selected in the cross-validaton iterations. This allows to identify a stable number of genes, while accounting for the diffences in sampling.

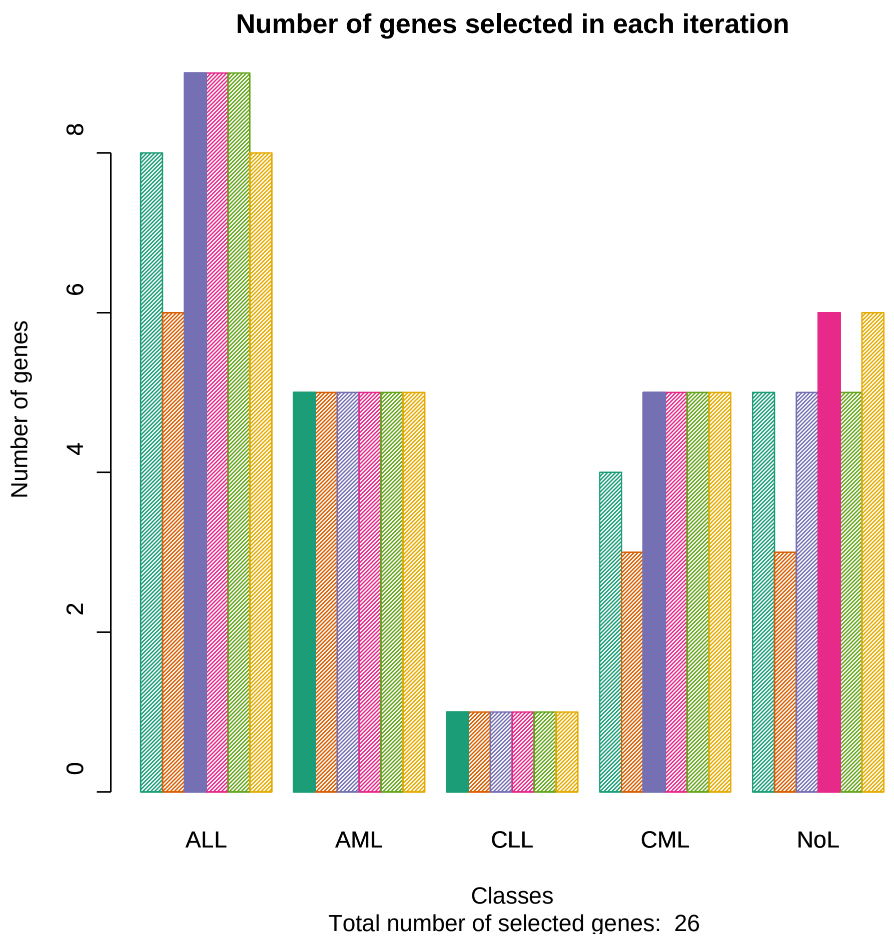


Figure 5. Plot of the number of genes selected in each iteration. The bars represent the number of genes with minimum error rates in each iteration. Each color represents an iteration. The filled bar is the final number of genes of each class selected to train the classifier.

Figures 4 and 5 show the gene selection for the leukemia's example.

3.5.2 Estimation of performance and generalization error procedure

The estimation of the *generalization error* (GE) of the classification algorithm is an option that can be included using the parameter *estimateGError=TRUE*. When this option is chosen, an independent validation is simulated by adding a second loop of cross-validation (CV) around the construction of the classifier. In each iteration of this loop, a few samples are left out of the *training* and used as *test* samples. This step allows to estimate and provide statistics and metrics regarding the quality of the classifier and the genes selected for classification. The parameters measured for the classifier are the following:

- **Sensitivity**: Proportion of samples from a given class which were correctly identified. In statistical terms it is the rate of true positives (TP). *Sensitivity* relates to the ability of the test to identify positive results.

$$Sensitivity = \frac{TP}{TP + FN} = TruePositiveRate$$

- **Specificity**: Proportion of samples assigned to a given class which really belonged to the class. In statistical terms it is the rate of true negatives (TN). *Specificity* relates to the ability of the test to identify negative results.

$$Specificity = \frac{TN}{TN + FP} = TrueNegativeRate$$

Note: In order to truly evaluate the classification, both sensitivity and specificity need to be taken into account. For example, 100% sensitivity for AML will be achieved by assigning all AML samples to AML. In the same way, 100% specificity will be achieved by not assigning any sample from other class to AML. Therefore, the classification will only be reliable if both -sensitivity and specificity- are optimized, by identifying all samples from one class while not having samples from another classes miss-classified.

- **Matthews Correlation Coefficient** (MCC): It is a measure which takes into account both true and false positives and negatives. It is generally regarded as a balanced measure of performance. In machine learning it is used as a measure of the quality of binary classifications.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **Global Accuracy**: Proportion of true results within the assigned samples.

- **Call Rate per class and Global Call Rate**: Proportion of *assigned* samples within a class or in the whole prediction.

$$CallRate = \frac{Assigned}{Assigned + NotAssigned}$$

The results about the estimation of performance and the generalization error are saved in the slot: *@generalizationError*

```
> leukemiasClassifier@generalizationError

Estimated accuracy, sensitivity and specificity for the classifier:
       Accuracy CallRate
Global      100       90
    Sensitivity Specificity MCC CallRate
ALL         100         100 100       90
```

```
AML         100          100 100       70
CLL         100          100 100      100
CML         100          100 100      100
NoL         100          100 100       90

To see all available statistics type "overview(EXAMPLE@generalizationError)"
```

To see all the available info gathered during estimation of performance use the *overview()* function:

```
> overview(leukemiasClassifier@generalizationError)
```

This object contains all the information regarding estimation of performance in different slots: *@accuracy*, *@sensitivitySpecificity*, *@confMatrix*, *@probMatrix*, *@querySummary.*

The slot *...@confMatrix* contains the confusion matrix. A confusion matrix is a table used to quickly visualize and evaluate the performance of a classification algorithm. The rows represent the real class of the samples, while the columns represent the class to which the samples were assigned. Therefore, the correctly assigned samples are in the diagonal.

```
> leukemiasClassifier@generalizationError@confMatrix

          prediction
testLabels ALL AML CLL CML NoL NotAssigned
       ALL   9   0   0   0   0           1
       AML   0   7   0   0   0           3
       CLL   0   0  10   0   0           0
       CML   0   0   0  10   0           0
       NoL   0   0   0   0   9           1
```

The slot *...@probMatrix* presents the probabilities of assignment to each class that are calculated during the 5-fold cross-validation. This *probability matrix* provides a good estimation of how easy or difficult is to assign each sample to its class. It also provides an indication about the likelihood to confuse one class with others:

```
> leukemiasClassifier@generalizationError@probMatrix

      ALL   AML   CLL   CML   NoL
ALL 0.697 0.060 0.073 0.067 0.102
AML 0.058 0.770 0.083 0.044 0.045
CLL 0.088 0.094 0.673 0.064 0.080
CML 0.055 0.107 0.064 0.633 0.141
NoL 0.073 0.072 0.055 0.145 0.654
```

The slot *...@classificationGenes.stats* includes calculations about the number of times that each gene was selected for classification in the 5-fold cross-validation executions:
- *timesChosen*, number of times that each gene is chosen for classification in the 5 CV.
- *chosenRankMean*, average rank of the gene only within the CV loops in which the gene was chosen for classification.
- *chosenRankSD*, standard deviation of the gene rank only within the CV loops in which the gene was chosen for classification.

- *geRankMean*, average rank of the gene in the 5 CV loops performed during the generalization error estimation.
- *geRankSD*, standard deviation of the rank of the gene in the 5 CV loops performed during the generalization error estimation.

```
> leukemiasClassifier@generalizationError@classificationGenes.stats$CLL
```

```
                timesChosen chosenRankMean chosenRankSD gERankMean gERankSD
ENSG00000176890           4           1.25         0.50        1.8     1.30
ENSG00000070081           2           1.50         0.71        2.4     0.89
ENSG00000104921           1           1.00         0.00        2.8     1.48
```

The slot *...@classificationGenes.num* includes calculations about the number of genes selected for each class in the 5 runs of the 5-fold cross-validation applied for the estimation of performance. These numbers allow to explore the number of genes that are used per class. However, the proper calculation of the final *number of genes* selected for each class in the classifier is done with the other 8-fold cross-validation which includes all the available samples (as indicated in section 3.5.1).

```
> leukemiasClassifier@generalizationError@classificationGenes.num
```

```
      ALL AML CLL CML NoL
CV 1:   6   7   1   3  10
CV 2:   3   2   1   8   6
CV 3:   9   2   2   6   5
CV 4:   2  16   2   9  16
CV 5:   3   5   1   8  10
```

3.6 Return III: Gene networks

Together to the classifier and the genes ranking, the third major result that the algorithm *geNetClassifier* produces are the gene networks associated to each class.

The gene networks for each class are built based on association parameters between genes. These association parameters are gene to gene co-expression calculated using a correlation coefficient (*Pearson* by default) and gene to gene interactions derived from *mutual information* (MI) analysis (*mi.empirical* entropy estimator from the R package minet [4]); both calculated along all the samples of each class of the studied dataset.

The *correlations* and *interactions* also allow to find possible redundancy between the genes as features in the classification procedure. Such redundancy can be tested by producing comparative classifiers that include or not the associated genes. Usually, classifiers without redundant genes need less features for classification.

The *...@genesNetwork* slot contains the list of networks.

```
> leukemiasClassifier@genesNetwork
```

```
$ALL
Attribute summary of the GenesNetwork:
```

```
Number of nodes (genes): [1] 1027
Number of edges (relationships): [1] 1942

$AML
Attribute summary of the GenesNetwork:
Number of nodes (genes): [1] 400
Number of edges (relationships): [1] 296

$CLL
Attribute summary of the GenesNetwork:
Number of nodes (genes): [1] 1916
Number of edges (relationships): [1] 18506

$CML
Attribute summary of the GenesNetwork:
Number of nodes (genes): [1] 949
Number of edges (relationships): [1] 6540

$NoL
Attribute summary of the GenesNetwork:
Number of nodes (genes): [1] 400
Number of edges (relationships): [1] 1993
```

```
> overview(leukemiasClassifier@genesNetwork$AML)
```

```
getNodes(...)[1:10]:
 [1] "ENSG00000078399" "ENSG00000143995" "ENSG00000185275" "ENSG00000154188"
 [5] "ENSG00000133101" "ENSG00000198795" "ENSG00000106004" "ENSG00000155792"
 [9] "ENSG00000119919" "ENSG00000106236"
... (400 nodes)

getEdges(...)[1:5,]:
     gene1             class1 gene2             class2 relation
[1,] "ENSG00000078399" "AML"  "ENSG00000143995" "AML"  "Correlation - pearson"
[2,] "ENSG00000154188" "AML"  "ENSG00000198795" "AML"  "Correlation - pearson"
[3,] "ENSG00000078399" "AML"  "ENSG00000106004" "AML"  "Correlation - pearson"
[4,] "ENSG00000154188" "AML"  "ENSG00000155792" "AML"  "Correlation - pearson"
[5,] "ENSG00000119919" "AML"  "ENSG00000108511" "AML"  "Correlation - pearson"
     value
[1,] "0.922460476283629"
[2,] "0.804443836092871"
[3,] "0.836149615702043"
[4,] "0.815177435058601"
[5,] "0.940367679337551"
... (296 edges)
```

Each of the networks in this list is an object of the class *GenesNetwork*. This class offers some functions to retrieve and count the edges and nodes, and also to subset the network (*getSubNetwork()*). Note that *getNodes()* includes all possible nodes even if they are no linked by edges.

```
> getNumEdges(leukemiasClassifier@genesNetwork$AML)

[1] 296

> getNumNodes(leukemiasClassifier@genesNetwork$AML)

[1] 400

> getEdges(leukemiasClassifier@genesNetwork$AML)[1:5,]

     gene1             class1 gene2             class2 relation
[1,] "ENSG00000078399" "AML"  "ENSG00000143995" "AML"  "Correlation - pearson"
[2,] "ENSG00000154188" "AML"  "ENSG00000198795" "AML"  "Correlation - pearson"
[3,] "ENSG00000078399" "AML"  "ENSG00000106004" "AML"  "Correlation - pearson"
[4,] "ENSG00000154188" "AML"  "ENSG00000155792" "AML"  "Correlation - pearson"
[5,] "ENSG00000119919" "AML"  "ENSG00000108511" "AML"  "Correlation - pearson"
     value
[1,] "0.922460476283629"
[2,] "0.804443836092871"
[3,] "0.836149615702043"
[4,] "0.815177435058601"
[5,] "0.940367679337551"

> getNodes(leukemiasClassifier@genesNetwork$AML)[1:12]

 [1] "ENSG00000078399" "ENSG00000143995" "ENSG00000185275" "ENSG00000154188"
 [5] "ENSG00000133101" "ENSG00000198795" "ENSG00000106004" "ENSG00000155792"
 [9] "ENSG00000119919" "ENSG00000106236" "ENSG00000148154" "ENSG00000108511"
```

The function *network2txt()* allows to save or export the networks as text files. This function produces two text files: one with the information about the *nodes* and another with the information about the *edges*. They are flat text files (.txt). In the case of the *edges* file, it includes the nodes that interact (gene1 – gene2), the type of link (correlation or interaction) and the value of such relation.

```
> network2txt(leukemiasClassifier@genesNetwork, filePrefix="leukemiasNetwork")
```

To produce just the files with the information about the *edges*:

```
> geneNtwsInfo <- lapply(leukemiasClassifier@genesNetwork,
+     function(x) write.table(getEdges(x),
+     file=paste("leukemiaNtw_",getEdges(x)[1,"class1"],".txt",sep="")))
```

These flat text files allow to export the networks to external software (e.g. *Cytoscape*, http://www.cytoscape.org).

The networks can also be exported using direct R connectors (e.g. RCytoscape) with the igraph objects returned by the function *plotNetwork* (sec. 6.4).

For more information see the class help *?GenesNetwork*.

4 External validation: query with new samples of known class

Once a classifier is built for a group of diseases or disease subtypes, it can be queried with new samples to know their class. However, before proceeding with samples whose class is unknown, an external validation is normally performed. An external validation consists on querying the classifier with several samples whose class is *a priori* known, in order to see if the classification is done correctly. As indicated in section 3.5.2, if the number of known samples is limited (as it is usually the case) to avoid leaving a sub-set of known samples out of the training, *geNetClassifier()* provides the *generalization error* option, which will simulate an external validation by using cross-validation. Despite this possibility, it is clear that using external samples (totally independent to the classifier built) is the best option to validate its performance.

In this section, we will proceed with an example of external validation with the leukemia's classifier. In *leukemiasEset*, the class of all the available samples is known *a priori.* Since we had 60 samples in the initial leukemia dataset and only 50 were used to train the classifier, the 10 remaining can be used for external validation.

The first step is to select the 10 samples that were not used for training:

```
> testSamples <- c(1:60)[-trainSamples]
> testSamples

 [1] 11 12 23 24 35 36 47 48 59 60
```

The classifier is then be asked about the class of these 10 samples using *queryGeNetClassifier()*:

```
> queryResult <- queryGeNetClassifier(leukemiasClassifier,
+ leukemiasEset[,testSamples])
```

This query will return the class that each sample has been assigned to, which will be saved into $class. It also returns the probabilities of assignment of each sample to each class in $probabilities.

```
> queryResult$class

GSM330195.CEL GSM330201.CEL GSM330611.CEL GSM330612.CEL GSM331037.CEL
          ALL           ALL           AML           AML           CLL
GSM331048.CEL GSM331392.CEL GSM331393.CEL GSM331675.CEL GSM331677.CEL
          CLL           CML           CML           NoL           NoL
Levels: ALL AML CLL CML NoL

> queryResult$probabilities

    GSM330195.CEL GSM330201.CEL GSM330611.CEL GSM330612.CEL GSM331037.CEL
ALL    0.82480476    0.72132949    0.04584317    0.03853380    0.04233982
AML    0.04145204    0.05669690    0.68053161    0.84706650    0.09093176
CLL    0.02591494    0.03200663    0.09622283    0.02028114    0.75041107
CML    0.04409894    0.08325862    0.08198307    0.07359096    0.04732196
```

```
NoL    0.06372931    0.10670835    0.09541932    0.02052760    0.06899539
    GSM331048.CEL GSM331392.CEL GSM331393.CEL GSM331675.CEL GSM331677.CEL
ALL    0.04115917    0.04569346    0.02645151    0.05492039    0.02213885
AML    0.09742257    0.17443163    0.02549073    0.05510842    0.04907441
CLL    0.71914364    0.13181179    0.03923288    0.09748276    0.01016039
CML    0.07715866    0.56354463    0.87901701    0.04714128    0.03225649
NoL    0.06511596    0.08451848    0.02980787    0.74534715    0.88636986
```

Since the real class of the samples is known, we can create a confusion matrix. Note: For using this matrix as input in upcoming functions the real classes should be placed as row names (*rownames*) and the predicted classes (assigned by the classifier) as column names (*colnames*).

```
> confusionMatrix <- table(leukemiasEset[,testSamples]$LeukemiaType,
+ queryResult$class)
```

Once we have executed the query, *externalValidation.stats()* can be used to calculate the parameters to evaluate the classifier (Section 3.5.2).

```
> externalValidation.stats(confusionMatrix)

$byClass
    Sensitivity Specificity MCC CallRate
ALL         100         100 100      100
AML         100         100 100      100
CLL         100         100 100      100
CML         100         100 100      100
NoL         100         100 100      100

$global
       Accuracy CallRate
Global      100      100

$confMatrix
    ALL AML CLL CML NoL NotAssigned
ALL   2   0   0   0   0           0
AML   0   2   0   0   0           0
CLL   0   0   2   0   0           0
CML   0   0   0   2   0           0
NoL   0   0   0   0   2           0
```

The class to class assignment probability matrix, that gives support to the confusion matrix, can be also created for the external validation analysis:

```
> externalValidation.probMatrix(queryResult,
+ leukemiasEset[,testSamples]$LeukemiaType, numDecimals=3)

      ALL   AML   CLL   CML   NoL
ALL 0.773 0.049 0.029 0.064 0.085
AML 0.042 0.764 0.058 0.078 0.058
CLL 0.042 0.094 0.735 0.062 0.067
CML 0.036 0.100 0.086 0.721 0.057
NoL 0.039 0.052 0.054 0.040 0.816
```

4.1 Assignment conditions

queryGeNetClassifier() includes an expert-like approach to decide if a sample is assigned to a class: instead of directly assigning a sample to the class with the highest probability, it takes into account the probability of belonging to the class and the probability of the closest class before taking the final decision.

By default, the probability to assign a sample to a given class should be at least double than the *random probability*, and the difference with the next likely class should also be higher than 0.8 times the *random probability*. For example, to assign a sample in a 5 class classifier, the highest probability should be at least 40% (2 x 0.20 = 0.40) and the probability of belonging to the closest class should be at least 16% lower than the highest (0.8 x 0.20 = 0.16). This implies that if a sample's probability to belong to one class is 55% and to belong to another class is 40%, since the the difference is lower than 16%, it is not clear enough, and it will be left as a *NotAssigned* (NA). This feature allows modulation of the assignment to resembles expert decision-making.

To allow adapting these conditions, *queryGeNetClassifier()* includes two coefficients that determine the *minimum probability for assignment* (minProbAssignCoeff), and the *minimum difference between the of the first and the second classes* (minDiffAssignCoeff). If these two coefficients are set up to 0 all samples will be assigned to the most likely class and therefore no samples will be left as *NotAssigned*.

```
> queryResult_AssignAll <- queryGeNetClassifier(leukemiasClassifier,
+     leukemiasEset[,testSamples], minProbAssignCoeff=0, minDiffAssignCoeff=0)
> which(queryResult_AssignAll$class=="NotAssigned")

integer(0)
```

On the contrary, the thresholds can be raised to increase the the certainty of the assignments: i.e. by setting the coefficients to 1.5 and 1, the minimum probability to be assigned is 0.6 (**1.5** x 2 x 0.20) and the minimum difference between first and second class probabilities is 0.2 (**1** x 0.20).

```
> queryResult_AssignLess <- queryGeNetClassifier(leukemiasClassifier,
+     leukemiasEset[,testSamples], minProbAssignCoeff=1.5, minDiffAssignCoeff=1)
> queryResult_AssignLess$class

GSM330195.CEL GSM330201.CEL GSM330611.CEL GSM330612.CEL GSM331037.CEL
          ALL           ALL           AML           AML           CLL
GSM331048.CEL GSM331392.CEL GSM331393.CEL GSM331675.CEL GSM331677.CEL
          CLL   NotAssigned           CML           NoL           NoL
Levels: ALL AML CLL CML NoL NotAssigned
```

In this case, these samples were left as *NotAssigned*:

```
> t(queryResult_AssignLess$probabilities[,
+     queryResult_AssignLess$class=="NotAssigned", drop=FALSE])

                     ALL       AML       CLL       CML        NoL
GSM331392.CEL 0.04569346 0.1744316 0.1318118 0.5635446 0.08451848
```

To help understanding how these thresholds behave for a specific dataset, if *geNetClassifier()* is executed with *estimateGError=TRUE*, it generates a plot presenting the assignment probabilities for each sample. This plot shows the probability of the most likely class *versus* the probability difference with next likely class for each sample. Therefore, it allows to view the effects of the 2 coefficients (*minProbAssignCoeff* and *minDiffAssignCoeff*) in the assignment.

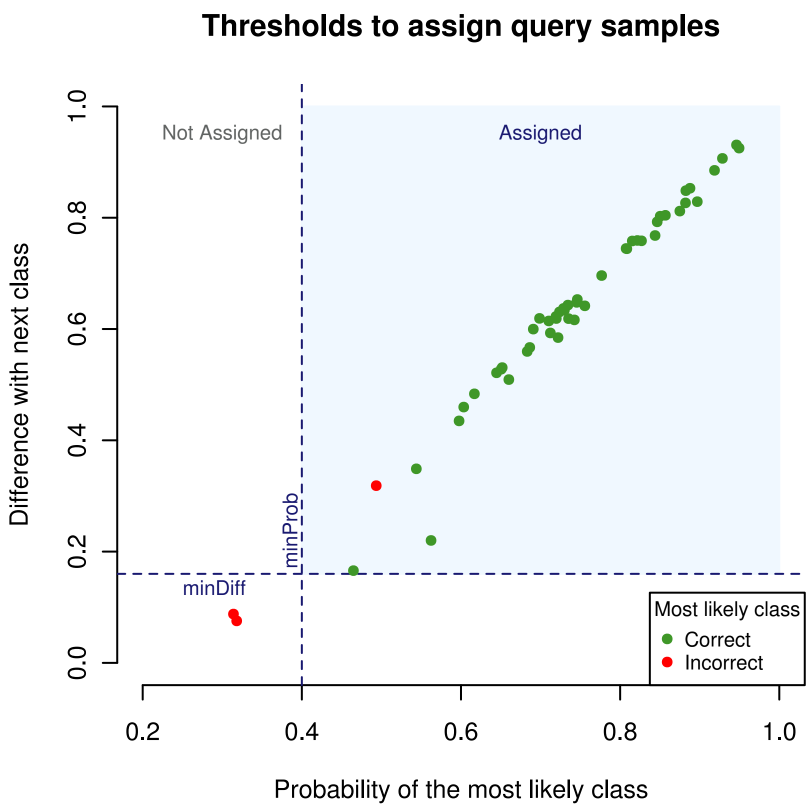


Figure 6. Assignment probabilities plot: It shows for each sample the probability of its most likely class *versus* the difference in probability with the next likely class. **Green** dots indicate that the probability of the most likely class is the correct class. **Red** dots indicate that the probability of the most likely class is not the correct class and, if assigned, such sample would have been missclassified. Dotted lines represent the chosen thresholds. The green area between them shows the samples that are actually assigned, those out of the green area are left as *NotAssigned*.

The plot in Figure 6 was obtained through the execution of *geNetClassifier()* with the leukemia's dataset. It shows that there are several samples under the assignment thresholds: these samples are left as *NotAssigned*. Out of these not assigned samples, the highest probability of some of them was to the real class (green), but some others was to an incorrect class (red). If the classifier had assigned the samples in red, it would have been an incorrect assignment.

5 Sample classification: query with new samples of unknown class

Once a classifier is built for a group of diseases or biological states, we can take external samples from new patients or new studies to query the classifier and know their class type.

Since we had 60 samples in the initial leukemia dataset and only 50 were used in the classifier, the 10 not used for training can be used as new samples to query the classifier and find out their class. In this case we will consider that the class of these samples is unknown.

```
> testSamples <- c(1:60)[-trainSamples]
```

queryGeNetClassifier() can then be used to ask the classifier about the class of the new samples.

```
> queryResult_AsUnkown <- queryGeNetClassifier(leukemiasClassifier,
+ leukemiasEset[,testSamples])
```

In the field *$class* of the return, we can see the class that each sample has been assigned to.

```
> names(queryResult_AsUnkown)
```

```
[1] "call"          "class"         "probabilities"
```

```
> queryResult_AsUnkown$class
```

```
GSM330195.CEL GSM330201.CEL GSM330611.CEL GSM330612.CEL GSM331037.CEL
          ALL           ALL           AML           AML           CLL
GSM331048.CEL GSM331392.CEL GSM331393.CEL GSM331675.CEL GSM331677.CEL
          CLL           CML           CML           NoL           NoL
Levels: ALL AML CLL CML NoL
```

If there were samples that had not been assigned to any class, they would be marked as*NotAssigned*. In the field *$probabilities*, we could see the probability of each sample to belong to each class. All these steps are very similar to the ones describes in section 4.1.

```
> t(queryResult_AsUnkown$probabilities[ ,
+ queryResult$class=="NotAssigned"])
```

```
     ALL AML CLL CML NoL
```

The function *querySummary()* provides a summary of the results by counting the number of samples that were assigned to each class and with which probabilities. It is a good way to have an overview of the classification results. In this case, the 100% *call rate* indicates that all samples have been assigned.

```
> querySummary(queryResult_AsUnkown, numDecimals=3)
```

```
$callRate
[1] 100

$assigned
    Count MinProb MaxProb  Mean    SD
ALL     2   0.721   0.825 0.773 0.073
AML     2   0.681   0.847 0.764 0.118
CLL     2   0.719   0.750 0.735 0.022
CML     2   0.564   0.879 0.721 0.223
NoL     2   0.745   0.886 0.816 0.100

$notAssigned
[1] "All samples have been assigned."
```

6 Functions to plot the results

6.1 Plot Ranked Significant Genes: *plot(...@genesRaking)*

As indicated in section 3.4.1, the default plot of a *genesRanking* can be obtained through the plot() function. This plot represents the gene rank obtained for each class *versus* the posterior probability of the genes.

```
> plot(leukemiasClassifier@genesRanking)
```

Some of the parameters to personalize this plot are:

- *lpThreshold* to set the value of the posterior probability threshold (marked as an horizontal line in the plot)
- *numGenesPlot* to determine the maximum number of genes that will be plot

```
> plot(leukemiasClassifier@genesRanking, numGenesPlot=3000,
+ plotTitle="5 classes: ALL, AML, CLL, CML, NoL", lpThreshold=0.80)
```

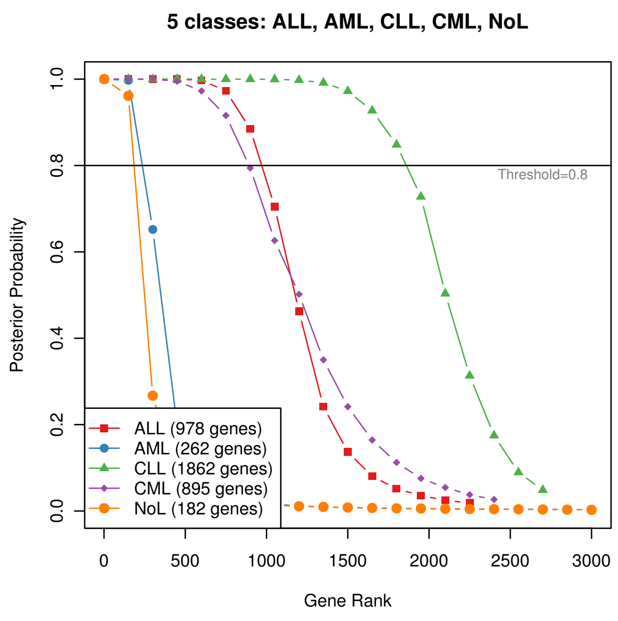


Figure 7. Plot of the posterior probabilities of the genes of 4 leukemia classes and the non-leukemia controls, ordering the genes according to their rank and setting the *lpThreshold* at 0.80.

calculateGenesRanking() allows to calculate (and plot) the ranking for a given data set without building the classifier:

```
> ranking <- calculateGenesRanking(leukemiasEset[,trainSamples],
+ "LeukemiaType")
```

6.2 Plot Gene Expression Profiles: *plotExpressionProfiles()*

The function *plotExpressionProfiles()* generates an overview of the expression profile of each gene along all the samples contained in the studied dataset. The plot will be saved as a PDF if *fileName* is indicated. The parameter *geneLabels* can be used to show a different name to the one included in the expression matrix (i.e. gene symbol instead of ENSEMBL ID or *Affymetrix* ID).

To plot the expression of 4 specific genes across the samples included in the leukemia's set:

```
> data(geneSymbols)
> topGenes <- getRanking(
+ getTopRanking(leukemiasClassifier@classificationGenes,numGenesClass=1),
+ showGeneID=TRUE)$geneID
> plotExpressionProfiles(leukemiasEset, topGenes[,c("ALL","AML"), drop=FALSE],
+     sampleLabels="LeukemiaType", geneLabels=geneSymbols)
```

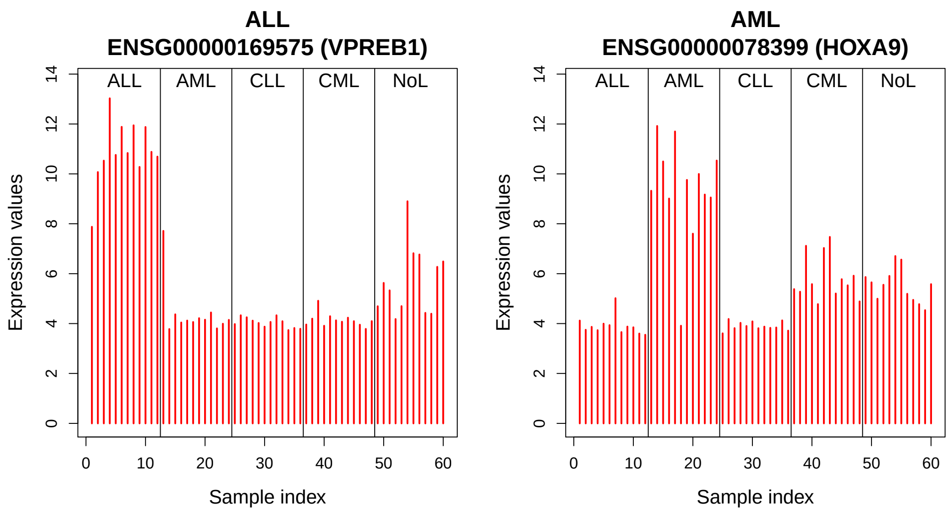


Figure 8. Plot of the expression profiles across 60 samples of 2 genes.

If a *geNetClassifierReturn* object is provided instead of a list of genes, it will plot the expression of all the genes used for training the classifier:

```
> plotExpressionProfiles(leukemiasEset[,trainSamples], leukemiasClassifier,
+     sampleLabels="LeukemiaType", fileName="leukExprs_trainSamples.pdf")
```

To plot the expression of all the genes chosen for classification for a specific class, for example AML:

```
> classGenes <- getRanking(leukemiasClassifier@classificationGenes,
+     showGeneID=TRUE)$geneID[,"AML"]
> plotExpressionProfiles(leukemiasEset, genes=classGenes,
+     sampleLabels="LeukemiaType", geneLabels=geneSymbols, fileName="AML_genes.pdf"
```

These plots can be modified in several ways, for example coloring specific samples or classes, or plotting the expression as boxplot

- Coloring specific samples or classes:

```
> plotExpressionProfiles(leukemiasEset, genes=topGenes[,3, drop=FALSE],
+                        sampleLabels="LeukemiaType",
+                        showMean=TRUE, identify=FALSE,
+                        sampleColors=c("grey","red")
+                        [(sampleNames(leukemiasEset)%in% c("GSM331386.CEL","GSM331
```

```
> plotExpressionProfiles(leukemiasEset, genes=topGenes[,3, drop=FALSE],
+                        sampleLabels="LeukemiaType",
+                        showMean=TRUE, identify=FALSE,
+                        classColors=c("red","red", "blue","red","red"))
```

- Plotting the expression as boxplot (grouped by classes):

```
> plotExpressionProfiles(leukemiasEset, genes=topGenes[,3, drop=FALSE],
+                        sampleLabels="LeukemiaType",
+                        type="boxplot", geneLabels=geneSymbols, sameScale=FALSE)
```

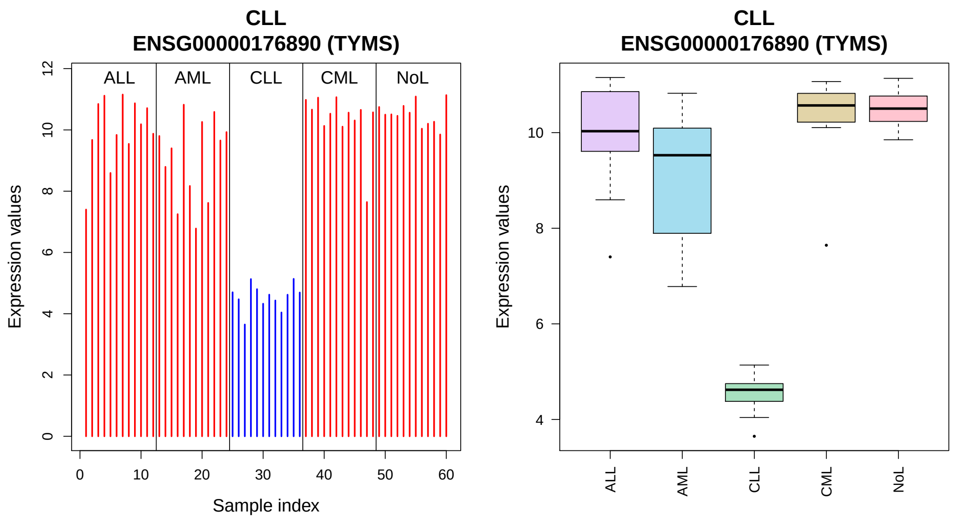


Figure 9. Two different versions of expression plot.

See *?plotExpressionProfiles* for more details.

6.3 Plot Genes Discriminant Power: *plotDiscriminantPower()*

The *discriminant power* is a parameter derived from the classifier's *support vectors* which resembles the power of each gene to mark the difference between classes.

The multi-class SVM algorithm (One-versus-One, OvO) produces a set of *support vectors* for each binary comparison between classes. Such *support vectors* include the *Lagrange coefficients* (alpha) for all the genes selected for the classification. Therefore, we can assign to each gene the sum of the *Lagrange coefficients* of all the *support vectors* of each class (represented as piled up bars in the plot). The *discriminant power* is then calculated as the difference between the value of the largest class and the closest (the distance marked by two red lines in the plot). In conclusion, the *discriminant power* is a parameter that allows the characterization of the genes based in their capacity to separate different classes (i.e. different diseases or diseases subtypes compared).

The *discriminant power* is calculated for each gene included in the classifier (the *@classificationGenes*) when it is built *geNetClassifier()*). The *plotDiscriminantPower()* function is included in the package to generate a graphic representation of the *discriminant power*.

```
> plotDiscriminantPower(leukemiasClassifier,
+ classificationGenes="ENSG00000169575")
```

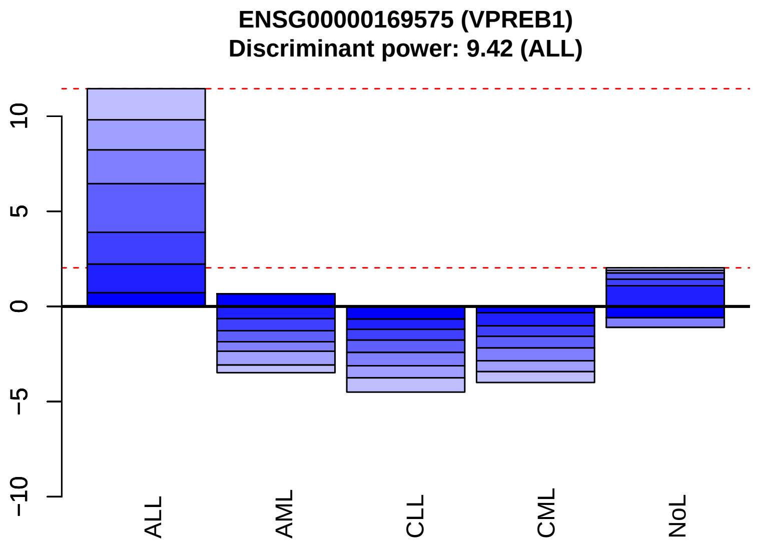


Figure 10. Plot of the discriminant power of gene VPREB1 (ENSG00000169575). The plot shows that this gene identifies class ALL and the closest class is NoL.

The next example shows the discriminant power of the top genes of a class. In order to plot more than 20 genes, or to save the plots as PDF, provide a *fileName*.

```
> discPowerTable <- plotDiscriminantPower(leukemiasClassifier,
+ classificationGenes=getRanking(leukemiasClassifier@classificationGenes,
+ showGeneID=TRUE)$geneID[1:4,"AML",drop=FALSE], returnTable=TRUE)
```

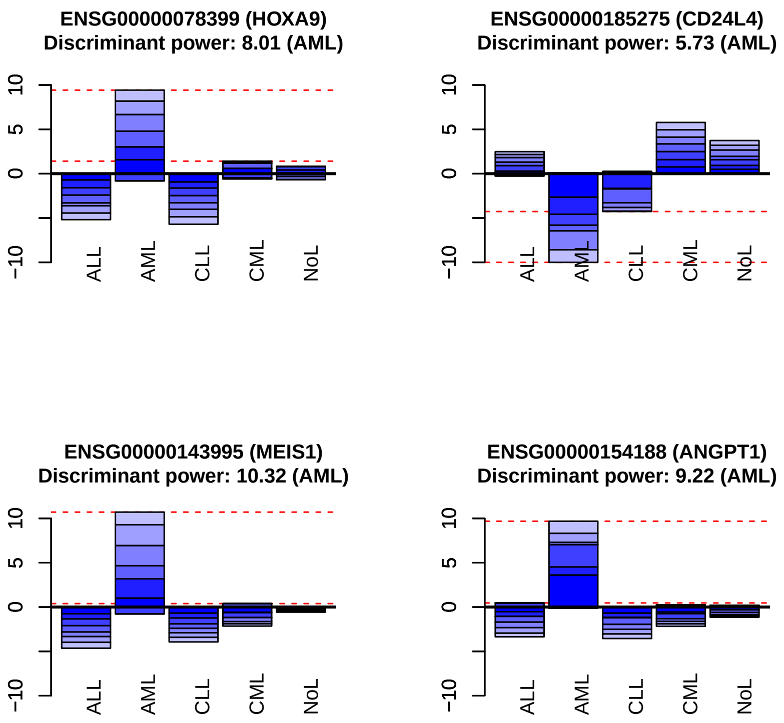


Figure 11. Plot of the discriminant power of the 4 genes that best discriminate AML class from the other classes. The figures indicate that MEIS1 (ENSG00000143995) presents the highest discriminant power. This gene encodes a homeobox protein that has been involved in myeloid leukemia. A high discriminant power can help to identify gene markers.

Some of the options to personalize the plot are *classNames* to provide a different name for the classes and textitgeneLabels to provide a alias for the genes. As usual, more details about the function are available at *?plotDiscriminantPower*.

6.4 Plot Gene Networks: *plotNetwork()*

The package also includes some functions to manipulate the networks produced by *geNet-Classifier()* (i.e. select part of a network and personalize the plots).

Step 1: Select a network or sub-network.
getSubNetwork() allows to select sub-networks. i.e. the sub-network containing only the classification genes:

```
> clGenesSubNet <- getSubNetwork(leukemiasClassifier@genesNetwork,
+ leukemiasClassifier@classificationGenes)
```

Step 2: Get the info of the genes to plot.
genesDetails() provides the available information about the genes. This information can be shown in the network: The gene name will be the node label. The expression of the gene will be shown with the node color, and the discriminant power will determine its size. In case the network includes genes selected for classification and genes which were not selected, the genes selected for classification will be plot as squares and the not selected as circles (only available for PDF plot, not on the dynamic view). For more details see the network legend in figure 14.

```
> clGenesInfo <- genesDetails(leukemiasClassifier@classificationGenes)
```

Step 3: Plot the network.
The network plots can be produced either using R interactive view (*tkplot* from *igraph*) or plotted as saved PDF files. Use *plotType="pdf"* to save the network as a static PDF file. This option is recommended to produce an overview of several networks. To produce interactive networks skip this argument. Iteractive plotscan be exported as a *postscript* files (.eps).

Some plot examples:
Network of ALL classification genes:

```
> plotNetwork(genesNetwork=clGenesSubNet$ALL, genesInfo=clGenesInfo)
```

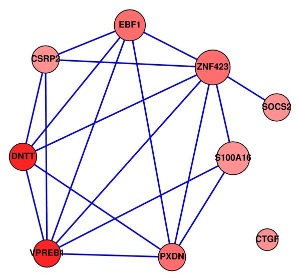


Figure 12. Gene network obtained for class ALL including the 9 classification genes selected for this disease.

Only connected nodes from ALL classification genes network:

```
> plotNetwork(genesNetwork=clGenesSubNet$ALL, genesInfo=clGenesInfo,
+ plotAllNodesNetwork=FALSE, plotOnlyConnectedNodesNetwork=TRUE)
```

AML network of the top 30 genes from the ranking (calculated as co-expression and mutual information):

```
> top30g <- getRanking(leukemiasClassifier@genesRanking,
+ showGeneID=TRUE)$geneID[1:30,]
> top30gSubNet <- getSubNetwork(leukemiasClassifier@genesNetwork, top30g)
> top30gInfo <- lapply(genesDetails(leukemiasClassifier@genesRanking),
+ function(x) x[1:30,])
> plotNetwork(genesNetwork=top30gSubNet$AML, genesInfo=top30gInfo$AML)
```

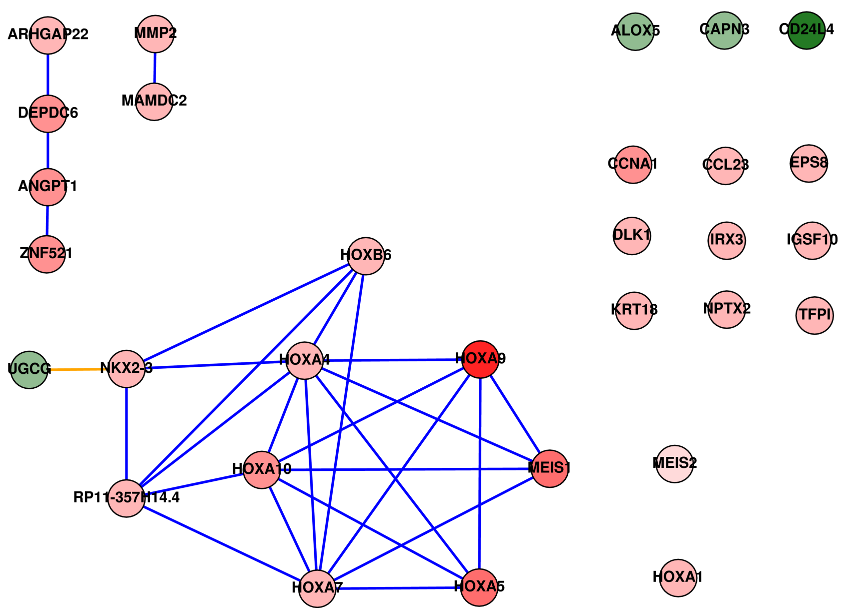


Figure 13. Gene network obtained for class AML including the top 30 genes selected from the gene ranking of this disease.

Network of the top 100 genes from AML ranking.
A preview of this network is automatically plotted for every class by *geNetClassifier()* if *plotsName* is provided.

```
> top100gRanking <- getTopRanking(leukemiasClassifier@genesRanking,
+ numGenes=100)
> top100gSubNet <- getSubNetwork(leukemiasClassifier@genesNetwork,
+ getRanking(top100gRanking, showGeneID=TRUE)$geneID)
> plotNetwork(genesNetwork=top100gSubNet,
+ classificationGenes=leukemiasClassifier@classificationGenes,
+ genesRanking=top100gRanking, plotAllNodesNetwork=TRUE,
+ plotOnlyConnectedNodesNetwork=TRUE, labelSize=0.4,
+ plotType="pdf", fileName="leukemiasNetwork")
```

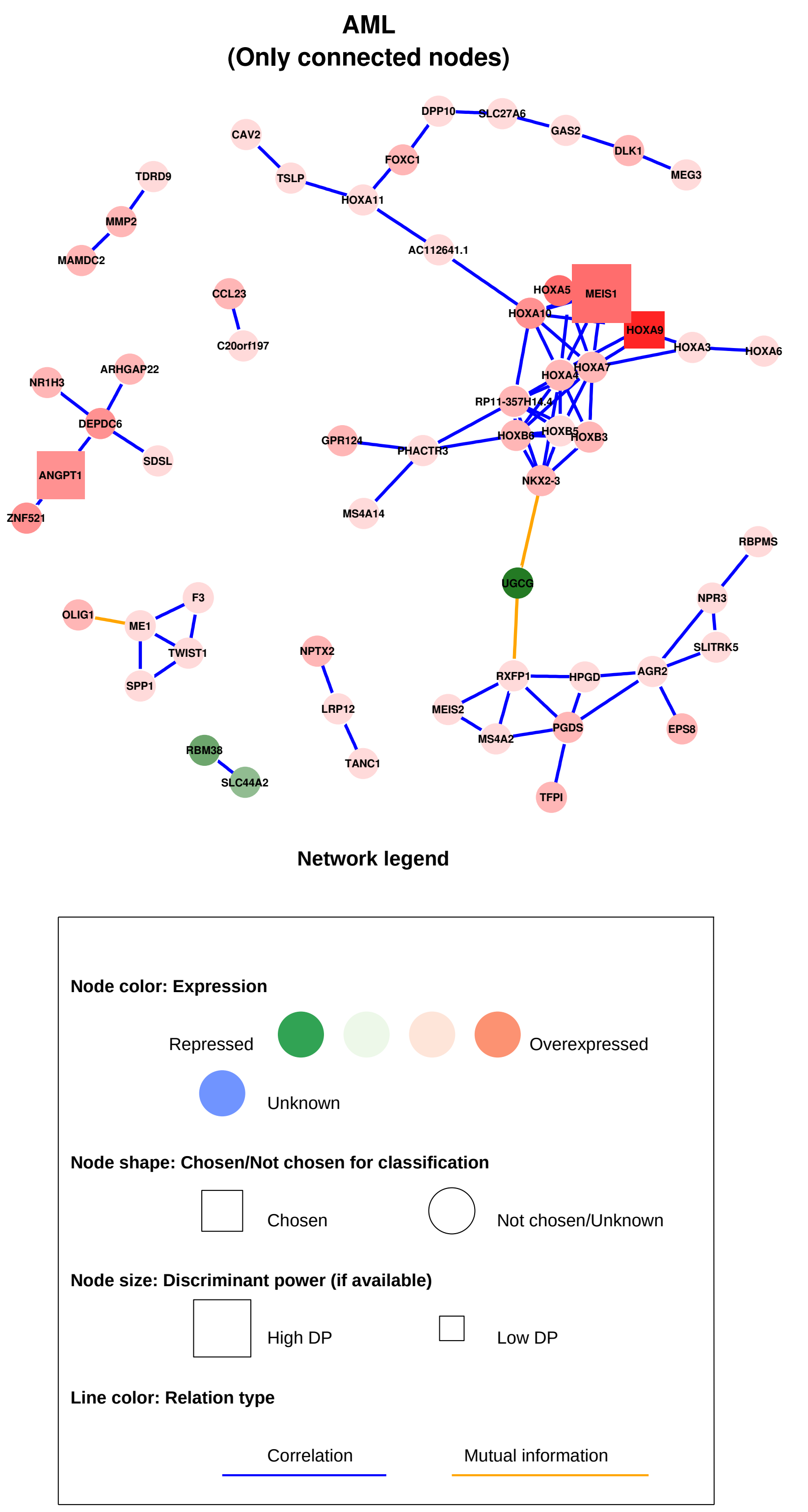


Figure 14. Gene network obtained for class AML selecting the 100 top genes from the gene ranking of this disease, but presenting only the connected nodes. The figure also includes the network legend indicating the meaning of the shapes and colors given to the nodes and edges.

Acknowledgements

This work was supported by Instituto de Salud Carlos III and by a grant from the Junta de Castilla y Leon and the European Social Fund to S.A and C.D.

References

[1] Haferlach T, Kohlmann A, Wieczorek L, Basso G, Kronnie GT, Bene MC, De Vos J, Hernandez JM, Hofmann WK, Mills KI, Gilkes A, Chiaretti S, Shurtleff SA, Kipps TJ, Rassenti LZ, Yeoh AE, Papenhausen PR, Liu WM, Williams PM, Foa R (2010). *Clinical utility of microarray-based gene expression profiling in the diagnosis and subclassification of leukemia: report from the International Microarray Innovations in Leukemia Study Group.* J Clin Oncol. 28: 2529-2537.

[2] Barrier A, Lemoine A, Boelle PY, Tse C, Brault D, Chiappini F, Breittschneider J, Lacaine F, Houry S, Huguier M, Van der Laan MJ, Speed T, Debuire B, Flahault A, Dudoit S (2005) *Colon cancer prognosis prediction by gene expression profiling.* Oncogene. 24: 6155-6164.

[3] Meyer D, Leischa F, Hornik K (2005). *The supportvector machine under test.* Neurocomputing. 55: 169-186

[4] Meyer PE, Lafitte F, Bontempi G (2008). *minet: A R/Bioconductor package for inferring large transcriptional networks using mutual information.* BMC Bioinformatics. 9: 461.

[5] Risueno A, Fontanillo C, Dinger ME, De Las Rivas J (2010). *GATExplorer: genomic and transcriptomic explorer; mapping expression probes to gene loci, transcripts, exons and ncRNAs.* BMC Bioinformatics. 11: 221.

[6] Morris C (1983). *Parametric empirical Bayes inference: theory and applications.* JASA. 78: 47-65.

[7] Kendziorski CM, Newton MA, Lan H, Gould MN (2003). *On parametric empirical Bayes methods for comparing multiple groups using replicated gene expression profiles.* Statistics in Medicine 22: 3899-3914.

[8] Benjamini Y, Hochberg Y (1995). *Controlling the false discovery rate: A practical and powerful approach to multiple testing.* J. Roy. Statist. Soc. Ser. B. 57: 289-300.