Supplementary Material for the paper:

GALGO: An R Package For Multivariate Variable Selection Using Genetic Algorithms

Victor Trevino and Francesco Falciani
School of Biosciences, University of Birmingham, Edgbaston, UK.

Abbreviations:
GA – Genetic Algorithms, FS – Forward Selection, BE – Backward Elimination.
DLDA – Diagonal Linear Discriminant Analysis, PAM – Shrunken Centroids, PAMR

– Shrunken Centroids R package, KNN – K-Nearest-Neighbours, SVM – Support Vector Machines, NC – Nearest Centroid, MLHD – Maximum Likelihood Discriminant Functions, RF – Random Forest.

1 Requirements before installing GALGO

- R software for statistical computing v2.0.1 or above.
  - It can be downloaded from http://www.r-project.org/.
- R.oo R package installed
  - Instructions for installing and downloading R.oo package are published by their authors in http://www.maths.lth.se/help/R/R.classes/. In Windows, it can be installed by using the option "Install Packages from CRAN…" available via the R console (go to Packages menu, select "Install Package(s)..", choose a mirror site and then select "R.oo")

2 Installing GALGO

- Download GALGO
  - For Windows: galgo_1.0-10.zip
  - For UNIX: galgo_1.0-10.tar.gz
- Install GALGO
  - In Windows
    - In the R console use the option "Install package(s) from local zip files…" from the "Packages" menu.
    - Select the GALGO zip just downloaded and click "open".
  - In UNIX
    - Log on as root or any user with privileges to write in system directories.
    - Use the command "R CMD INSTALL [galgo .gz file]"

3 GALGO Documentation

- Manual (Manual.pdf)
  - This manual includes a brief introduction to statistical modelling with particular reference to a Genetic Algorithms (GA) strategy for variable selection. It also includes a step by step example with a detailed description of GALGO functionality and on how to use GALGO as a general tool to solve optimization problems.
  - This file is included in the GALGO package. Once GALGO has been loaded, options to open these files should be available from the "Vignettes" menu in the R console.
- Objects, Methods and Functions (Galgo.pdf)
  - This file contains the standard documentation for functions and objects in R in .pdf format.
- This document as a .pdf (GALGO-SupplementaryMaterial.pdf)

- Help system for Objects, Methods and Functions
    - The standard documentation of GALGO in R system can be accessed in the usual way in R, typing "?" followed by the object name or by the method "dot" object. Examples:
        - ?BigBang
        - ?configBB.VarSel
        - ?plot.BigBang
        - ?fowardSelectionModels.BigBang
        - ?confusionMatrix.BigBang
    - The help system can also be explored by HTML typing "help.start()" or by accessing the option "HTML help" in menu Help in R GUI (for Windows). Galgo package may be accessed through search engine or Packages->Galgo.

4 Why developing GALGO?

In the analysis of large datasets, such as data obtained using Functional Genomics Technologies, the selection of gene signatures predictive of sample features (for example disease type) is a difficult problem. Commonly the number of samples is very low (hundreds or dozens) and certain aspects of the samples are known (for example disease type, strain, treatment, etc). One of the most basic problems is the selection of genes whose profile is, in some way, associated to the known sample type, which in turn would allow acquiring more knowledge about the mechanism of action, generating new hypothesis, directing further research, selecting biomarkers, and choosing potential drug targets. In statistics, this association of profiles to known sample types is called "supervised classification" and there are several classification methods that "test" if genes are related to samples phenotype. These methods can be subdivided in univariate and multivariate methods. Univariate methods evaluate each variable (e.g. a gene) at the time for its ability to discriminate between two or more groups of samples. PAMR (Tibshirani *et al.* 2002), GeneSpring (Silicon Genetics, Redwood City, CA), and TNASAS (Vaquerizas *et al.*, 2005) are perhaps the most commonly used software packages by the Functional Genomics community that implement univariate variable selection methods for classification. Univariate variable selection methods use some statistics to identify genes that are differentially expressed between two or more groups of samples and then uses the most differentially expressed to construct a statistical model (Figure 1). These methods have demonstrated to perform well, however, in some cases they can be ineffective regardless of the classification method used. An obvious conceptual limitation of univariate approaches is also the lack of consideration that genes works in the contexts of interconnected pathways and therefore it is their behaviour as a group that may be predictive of the phenotypic variables. Multivariate selection methods (Figure 2) may seem to be more suitable for the analysis of Biological data since variables (such as gene expression values) are tested in combination to identify interactions between genes. However, the extremely large number of models that can be constructed from different combination of thousands of genes cannot be extensively evaluated using available computational resources. An alternative to the extensive analysis of all possible models is the use of search procedures that "explore" the data looking for good, although not optimal, sets of variables. Recently, Markov Chain

Monte Carlo methods and Genetic algorithms have been applied successfully to the analysis of microarray data (Li et al. 2001; Ooi et al. 2003; Sha et al. 2004).

At present, there is no available software package to support the development of statistical models using multivariate variable selection strategies. To address this issue we have developed GALGO, an R package that uses a genetic algorithm search procedure coupled to statistical modelling methods for supervised classification (Figure 2). GALGO is relatively easy to use, can manage parallel searches and has a toolset for the analysis of models. Although GALGO include a number of statistical modelling methodologies to solve classification problems, GALGO can be used as a general tool to solve optimization problems. This requires rewriting the fitness function to specify the criteria for the selection of good variable subset. Because of the functionality that is already available in R, this can be achieved relatively easily. This manual provides a step-by-step tutorial to solve classification problems using microarray data. It also provides examples of the use of GALGO as a general tool to solve optimization problems.

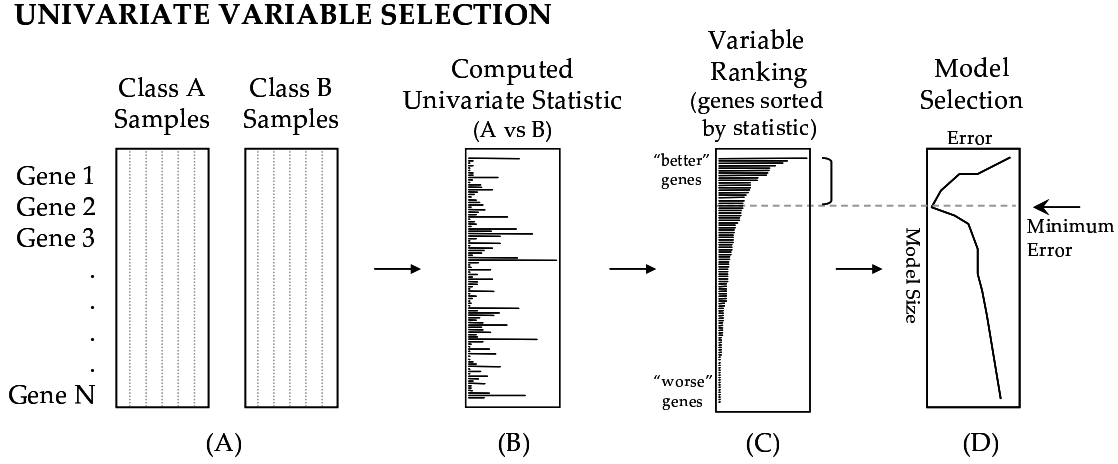

Figure 1 Schematic representation of univariate variable selection. A dataset of two classes of samples (A) is assessed using a univariate test (B) to rank genes by their sole ability to distinguish between classes (C). Then, a forward selection strategy using a classification method is used to detect the number of ranked genes that generates the lowest error (D).

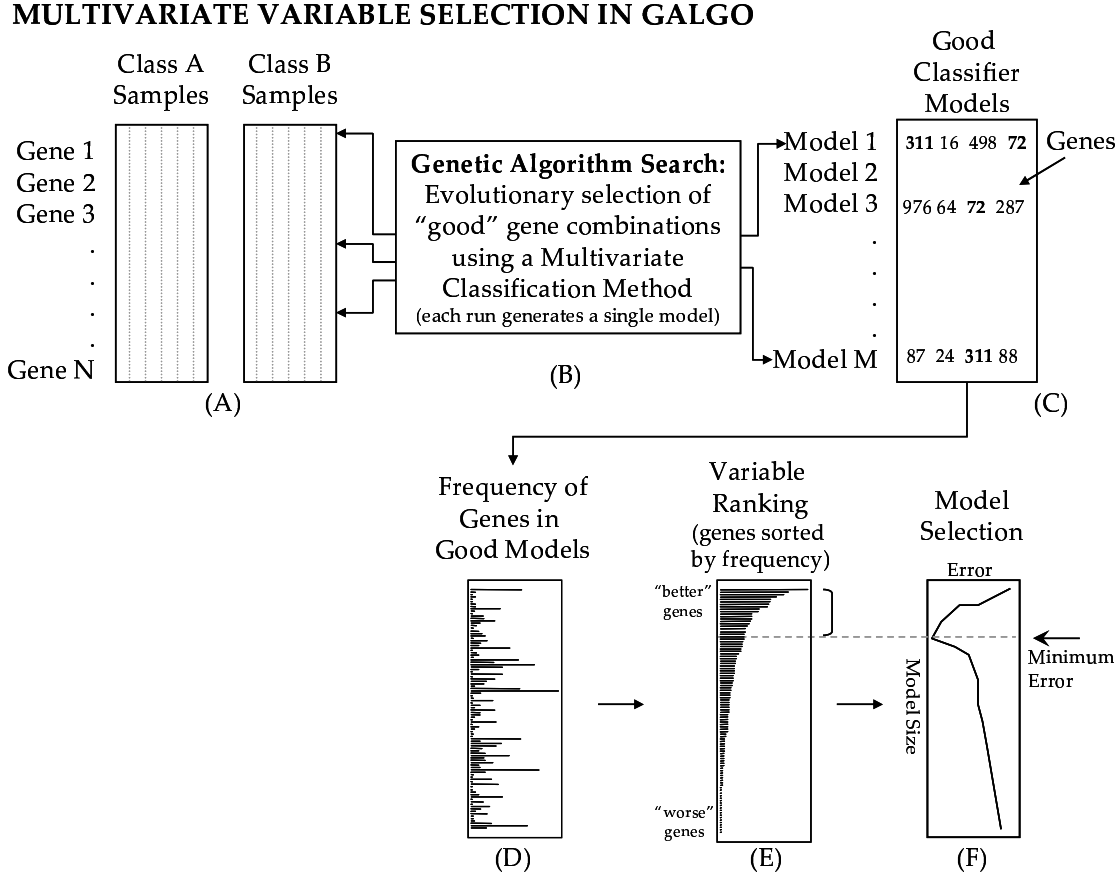

Figure 2 Schematic representation of multivariate variable selection. From a dataset of two classes of samples (A), a genetic algorithm (B) searches and evolves combination of genes (chromosomes representing a multivariate model) that distinguish between classes using a classification method. A number of models are generated performing this procedure several times (C). These models may differ in gene content but with similar high classification accuracy. Genes appearing multiple times in different models suggest these genes are important for the classification problem in a multivariate context. Therefore, the number of times (frequency) a gene appears in a model is computed (D). These frequencies are used to rank genes (E). Then, a forward selection strategy is used to select a representative model that generates the lowest error (F).

5 Application

The GALGO package has been conceived as an implementation of GA in object-oriented paradigm under the R language. R is a statistical programming environment that is platform-independent, robust, freely available, and is widely used for the analysis of functional genomics data. Because of the large collection of statistical functions available in R, GALGO can be also used to find optimal variable subsets that maximize a wide range of user-defined fitness functions. GALGO uses a GA procedure for selecting models with a high fitness value (e.g. classification accuracy) and implements functions for the analysis of the populations of selected models as well as functions to reconstruct and characterize representative summary models (Li et al. 2001). In addition, a function for predicting the class of unknown samples is available.

5.1 What is a Genetic Algorithm?

Genetic Algorithms (GAs) are variable search procedures that are based on the principle of evolution by natural selection. The procedure works by evolving sets of variables (chromosomes) that fit certain criteria from an initial random population via cycles of differential replication, recombination and mutation of the fittest chromosomes. The concept of using *in-silico* evolution for the solution of optimization problems has been introduced by John Holland in 1975 (Holland 1975). Although their application has been reasonably widespread (see Goldberg's book (Goldberg, 1989)), they became very popular only when sufficiently powerful computers became available. What follows is a Step by Step description of the procedure in the context of a classification problem (see Figure 3) for a schematic representation of the procedure, note that we will use stages here to avoid confusion with those steps in the general GALGO pipeline):

Stage 1: The procedure initially creates a number of random variable sets (chromosomes). These variable sets form a population of chromosomes (niche).

Stage 2: Each chromosome in the population is evaluated for its ability to predict the group membership of each sample in the dataset (fitness function). This is achieved by training a statistical model. The GA tests the accuracy of the prediction and assigns a score to each chromosome that is proportional to the accuracy.

Stage 3: When a chromosome has a score higher or equal than a predefined value, this chromosome is selected and the procedure stops; otherwise, the procedure continues to stage 4.

Stage 4: The population of chromosomes is replicated. Chromosomes with a higher fitness score will generate more numerous offspring.

Stage 5: The genetic information contained in the replicated parent chromosomes is combined through genetic crossover. Two randomly selected parent chromosomes are used to create two new chromosomes (Figure 4). This crossover mechanism allows a better exploration of possible solutions recombining good chromosomes.

Stage 6: Mutations are then introduced in the chromosome randomly. These mutations produce that new genes are used in chromosomes.

Stage 7: The process is repeated from stage 2 until an accurate chromosome is obtained. The cycle of replication (stage 4), genetic cross-over (stage 5) and mutations (stage 6) is called generation.

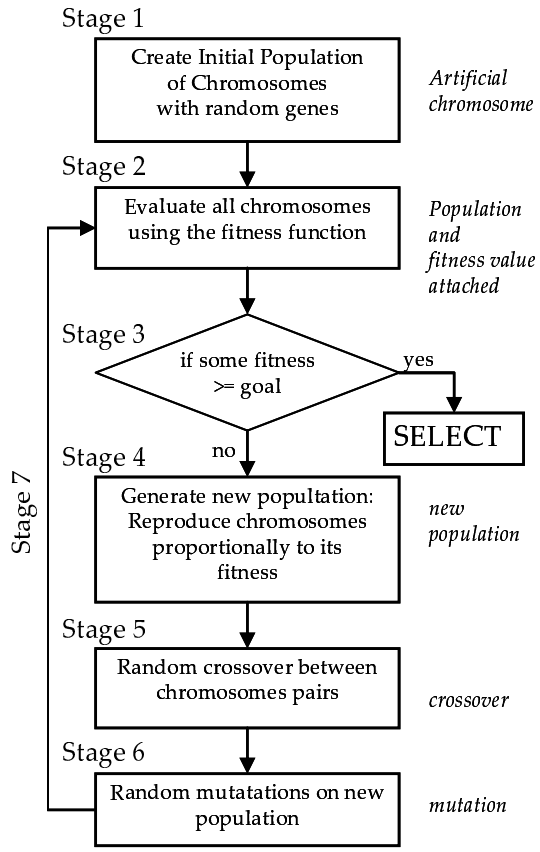

Figure 3 Schematic representation of the GA Procedure.

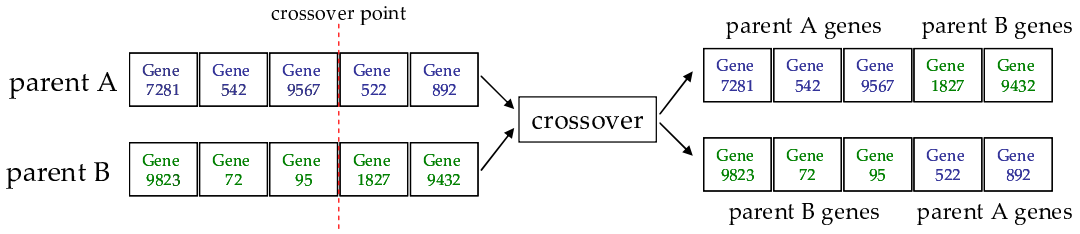

Figure 4 Schematic representation of the Crossover.

5.2 Object-Oriented Design

The GA procedure evaluates collections of variable subsets for their ability to perform a defined task (e.g. supervised classification). It begins from a collection of random sets and, using principles of natural selection, evolves better fitted models until a model of desired accuracy has been found. In the GA terminology variables are defined as genes whereas a subset of n variables that is assessed for its ability to fit a statistical model is called a chromosome. Populations of chromosomes are organized in niches that are independently evolving environments. However, niches have the possibility to occasionally exchange chromosomes with a process called migration. Multiple niches can then be part of a world. The object design of the GALGO

package (illustrated in Figure 5) reflects the structure we described above. In GALGO, Gene object represents a variable whereas the Chromosome object stands for a set of n variables that will be included in the multivariate model, which will be evaluated using the fitness function. A Niche object organizes chromosomes in populations whereas the World object includes several niches. The Galgo object performs the GA evolutionary process and saves the best chromosome as a result. Finally, a BigBang object stores the results of the search for further analysis. These objects have properties that allow users to control the process. We included most common GA operators as Reproduction, Mutation, Crossover, Migration, and Elitism as methods. An important characteristic of GALGO is that the user can add custom defined properties to add new functionality. All objects can be extended and their methods can be overwritten to provide more flexibility. In the manual, available in the supplementary material, we describe an example where the method mutate is overwritten to allow differential mutation rates or variables with defined characteristics.

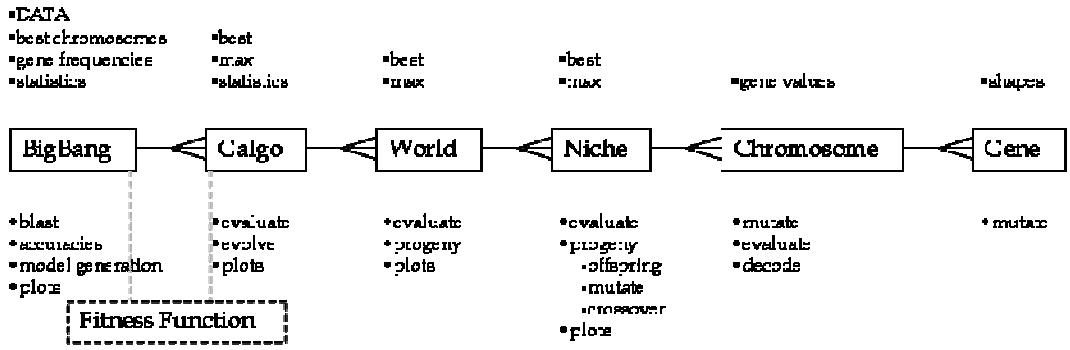

Figure 5 Simplified object-oriented structure of the GALGO package. Boxes represent objects, which are connected by one-to-many relationships hierarchically. Major object properties are marked with solid squares whereas core methods are marked with solid circles. Dashed box represents the fitness function, which are included in GALGO for several classification methods. Dashed lines represent logical connections.

5.3 Analysis Strategy

Figure 6 shows a flowchart summarizing a typical analysis pipeline that can be implemented using the GALGO package. For simplicity the process has been represented in four steps. The first step consists in specifying the input data, the error estimation strategy, the statistical modeling technique, the fitness function and the parameters for the GA procedure. In the second step the GA procedure searches for and collects models that have a high value of the fitness function (e.g. classification accuracy, evaluated using a cross-validation procedure). In the third step the population of models selected in step 2 is analyzed for its variable composition and classification accuracy. In the fourth step a forward selection strategy is used to develop and test a statistical model that is representative of the model population.

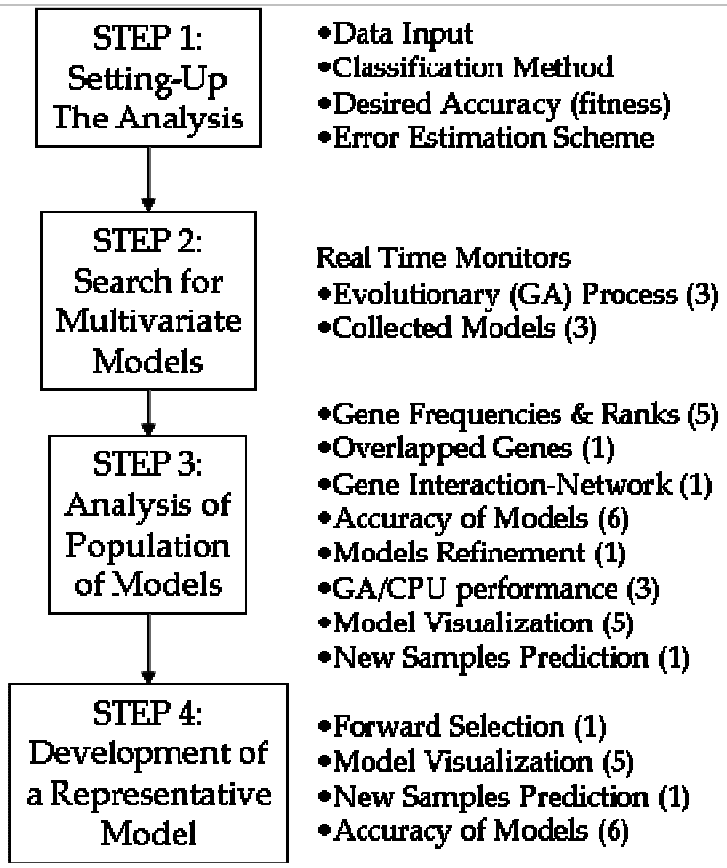

Figure 6 Schematic representation of a typical application of GALGO package. The figure describes a general analysis pipeline that can be implemented in GALGO. A summary of functionality in each step is listed in the right side of the scheme. The number of individual functions is shown in brackets.

5.4 Performance

Developing statistical models using a bona fide multivariate variable selection strategy is a very computer intensive procedure and depends on the particularities of the dataset. The GA procedure is a very efficient method for developing multivariate models. For example, a single evolutionary cycle can select a highly predictive model in seconds. However, it is advisable to sample a large number of solutions to represent the solution space that can be explored with this procedure. Typically, it is necessary to collect between 200 and 1000 chromosomes before observing some degree of convergence. In a typical classification problem with a microarray dataset we may require several hours of computation to collect a sufficient number of chromosomes. Table 1 summarizes the performance of GALGO in three different classification problems of increasing complexity. In this example we have collected 500 chromosomes. In order to increase its performance, GALGO has functions that allow the parallelization of the search process on different CPUs (see manual).

Problem	*Samples*	*Chromosome Size*	*Average accuracy*	*Running time*
2-class	143	5	1.0	0h 58m 48s
5-class	233	5	0.989	2h 37m 49s
7-class	327	7	0.926	8h 29m 3s

Table 1 – Performance of GALGO.

6 An Example of Supervised Classification Using GALGO

This section describes a typical application of GALGO in biomarker discovery using large scale expression profiling data. The aim of this analysis is to identify gene sets that are predictive of disease type in a panel of leukaemia patients (see section 6.1). This tutorial will describe the main basic functionality implemented in GALGO. A complete description of the functionality available in GALGO can be found in the Manual and in the software documentation.

6.1 Dataset

The dataset used in this analysis is derived from the work of Yeoh *et al.* (2002). The dataset represents the gene expression profiles of five groups of patients (EMLLA[i], Hyp+50, MLL, T, and TEL including 27, 64, 20, 43, and 79 samples respectively). The original dataset comprising 12,600 genes have been processed to eliminate the most invariant genes. The standard deviation and difference between maximum and minimum expression value were calculated for each gene. The genes were then ranked by both values. The genes that had any of these values in the top 15% of the ranked lists were selected for further analysis. The dataset after filtering contained the expression values for 2,435 genes.

6.2 Step 1 – Setting-Up the Analysis

In the GALGO package we have included a data-frame object (ALL) that contains the normalized gene expression values. The object is a matrix in which rows are genes and columns are samples. The identity of the samples is defined in a different object called (ALL.classes). Both objects are loaded using the function data (name object).

In R type:

```
> library(galgo)
> data(ALL)
> data(ALL.classes)
```

Data from an external text file can be loaded within the wrapper function (see manual for details).

The wrapper function "configBB.VarSel" is used to specify the data, the parameters for the GA search, the classification method, the error estimation method, and any other user-defined parameter. This function builds a BigBang object that contains the data and the values of all parameters and will eventually store the results of the analysis.

To set up the GA search type in R:

```
> bb.nc <- configBB.VarSel(
data=ALL,
classes=ALL.classes,
classification.method="nearcent",
chromosomeSize=5,
maxSolutions=300,
goalFitness = 0.90,
main="ALL-Tutorial",
saveVariable="bb.nc",
saveFrequency=30,
saveFile="bb.nc.Rdata")
```

The code above configure a BigBang Object that will store 300 chromosomes (*maxSolutions=300*) which will contain 5 genes (*chromosomeSize=5*) that correspond to models developed using a nearest centroid classifier (*classification.method="nearcent"*) with a classification accuracy of at least 90% (*goalFitness=0.9*). The other parameters define the name of the saved object that is created (*saveVariable="bb.nc"*), the frequency of saving the results in a file (*saveFrequency=30*) and the name of the file where the results are saved (*saveFile="bb.nc.Rdata"*).

In defining the BigBang Object GALGO pre-process the dataset creating two subsets of data that are used respectively for the selection of the chromosomes (training data) and for the final error estimation of the selected chromosomes (test). In **BOX 1** we give a brief explanation of the options that GALGO offers to estimate the classification accuracy. Further information is available in the Manual (see section 3).

The wrapper function *configBB.VarSel* can also be used to configure additional functions. These are explained in dept in the package Manual. A brief description of the full list of parameters that can be defined within the wrapping function can be obtained typing:

```
> ?configBB.VarSel
```

6.3 Step 2 - Evolving Models/Chromosomes

Once the BigBang and Galgo objects are configured properly, we are ready to start the GA procedure for collecting chromosomes associated to good predictive models for tumour class. This is achieved by calling the method "blast".

In R type:

```
> blast(bb.nc)
```

This command starts the GA search and will continue until the desire number of chromosomes is collected. The entire procedure can take from minutes to hours depending on the degree of difficulty of the classification problem, on the classification method, and on the GA search parameters.

The default configuration in the wrapping function displays the state of the BigBang and Galgo objects in the command line including the approximated remaining time.

This is an example of the text output for one GA cycle (61 generations):

```
[e] Starting: Fitness Goal=0.9, Generations=(10 : 200)
[e]     Elapsed Time    Generation      Fitness %Fit    [Next Generations]
[e]     0h 0m 0s        (m)     0       0.64103 71.23%  +++++++...+.........
[e]     0h 0m 6s                20      0.87179 96.87%  ....................
[e]     0h 0m 14s               40      0.87179 96.87%  .....+..+.....+.+...
[e]     0h 0m 22s               60      0.92308 102.56% +
[e]     0h 0m 22s       ***     61      0.92308 102.56% FINISH: 2164 1612...
[Bb]    300     299     Sol Ok  0.92308 102.56% 61      22.16s  3722s
4054s   14 (0h 0m 14s )
```

The last line (starting with "[Bb]") corresponds to the current collection of the BigBang object. This line shows respectively the number of evolutions (300 in this case), the number of evolutions that have reached the goal fitness (299), the status of the last evolution cycle (Sol Ok – the goal fitness was reached), the fitness value of the best chromosome from the last evolution (0.92408) along with it percentage relative to the goal fitness (102.56%), the number of generations required (61), the process time spent in last evolution (22.16 seconds), the cumulative process time spent in all evolutions (3,722 seconds), the cumulative real time (4,054 seconds, which considers the time spent by saving the object and other operative system delays), and the remaining time needed to collect the previously specified number of chromosomes (14 seconds).

Lines starting with "[e]" represent the output of the evolutionary process (the genetic algorithm search). The first line of each evolution shows the goal fitness and the configured minimum and maximum number of generations. Successive lines show, in columns, the elapsed time, the current number of generation (by default refreshed every 20 generations) and the current best fitness along with the percentage relative to the goal fitness. The last column summarize the behaviour of next generations, "+" means that maximum fitness of the current population has increased, "-" means that it has decreased, and "." means that it has not changed. "G" appears occasionally when the fitness goal has been reach before the minimum number of generations.

The default configuration would show three plots summarizing the characteristics of the population of selected chromosomes (see Figure 7). The topmost plot shows the frequency (vertical axis) of each gene (horizontal axis) in the Chromosome population. The default settings display the top most frequent 50 genes colour-coded on the basis of their frequency rank. The middle plot shows the stability of the rank of the top 50 genes over the number of different search cycles (see section 0). The plot at

the bottom displays the distribution of the number of generations required by the GA process to reach a solution.

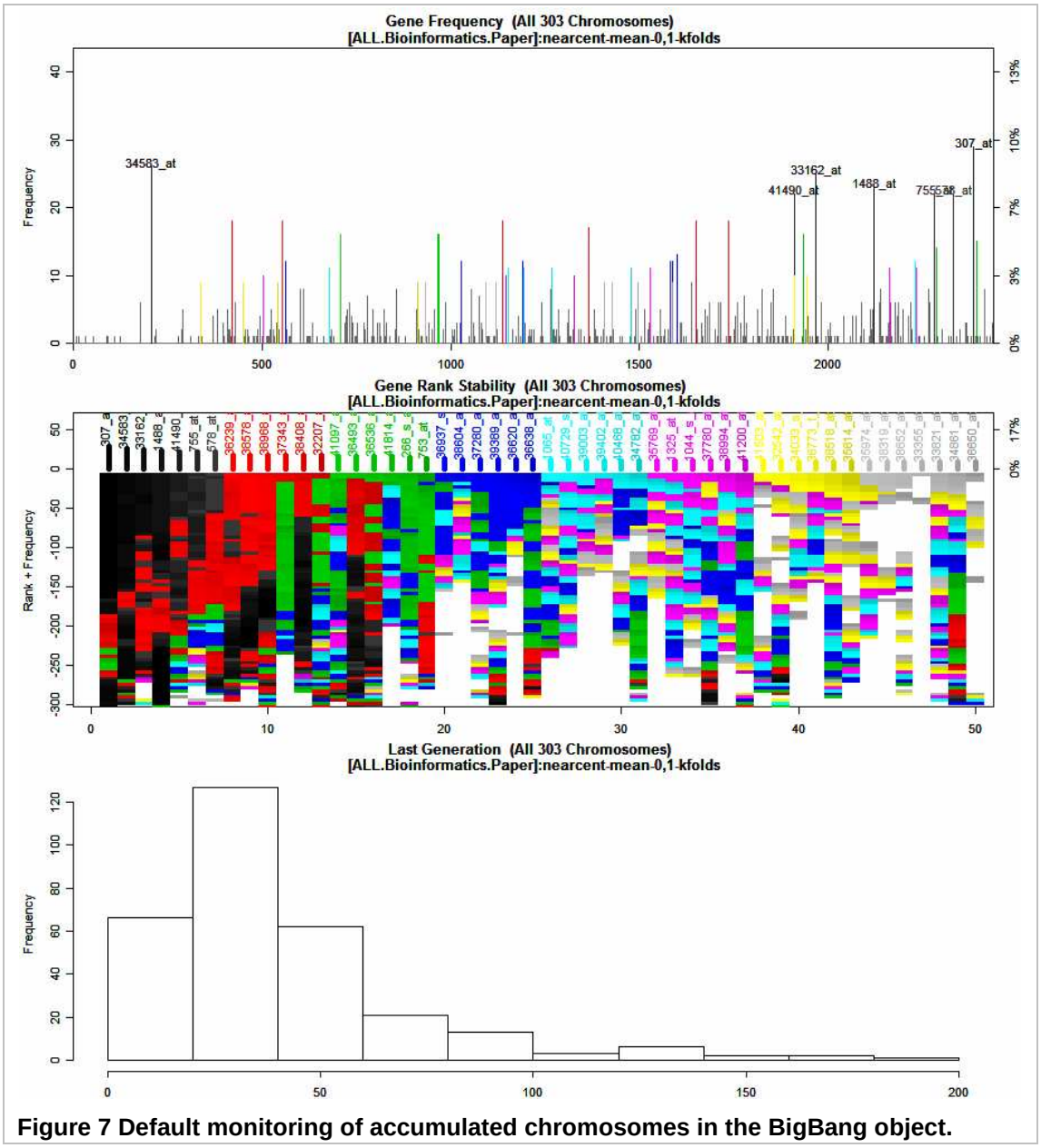

Figure 7 Default monitoring of accumulated chromosomes in the BigBang object.

The blast method terminates either when all the requested chromosomes have been found or if the process is interrupted (by typing the ctrl-c keys in Linux or esc in windows). It is recommended to break the process to perform a preliminary analysis after the initial 100-200 chromosomes are selected. The process can be resumed by typing the blast command again. The result of the last evolution might be lost but the accumulated results should remain be intact. Resuming the process will have the effect of restarting the Galgo object as in any cycle. The possibility to interrupt the process is very useful for initial exploratory analysis since the most updated results can be analysed and can be saved anyway using the saveObject method. Instead of interrupting the process, you can open a new R console and benefit from the use of progressive saving strategy that updates the current object called “bb.nc” into a file named “bb.bc.Rdata” once at least 30 solutions have been reached (controlled by

saveVariable, saveFile, and saveFrequency parameters respectively). To do this, a previously saved object can be loaded in GALGO using the loadObject method in a new R console window:

```
> library(galgo)
#change directory to yours
> loadObject(“bb.nc.Rdata”)
```

Once the file is loaded, the loadObject method displays a summary of the loaded variables and their classes and you can proceed to the analysis step.

GALGO can also summarise the population of evolving chromosomes in real time. The code below shows the modifications to the definition of the BigBang Object that are required to activate this function (marked in red).

```
> x11()
> x11()
> bb.nc <- configBB.VarSel(
data=ALL,
classes=ALL.classes,
classification.method=”nearcent”,
chromosomeSize=5,
maxSolutions=300,
goalFitness = 0.90,
main="ALL-Tutorial",
saveVariable=”bb.nc”,
saveFrequency=30,
saveFile=”bb.nc.Rdata”,
callBackFuncGALGO=plot,
callBackFuncBB=function(...){dev.set(2);plot(...);dev.set
(3); })
```

The topmost panel in Figure 8 shows the gene composition of the evolving chromosomes. The middle plot shows the evolution of the fitness relative to the goal in the course of generations. The plot at the bottom shows the gene composition of the maximum chromosome across generations.

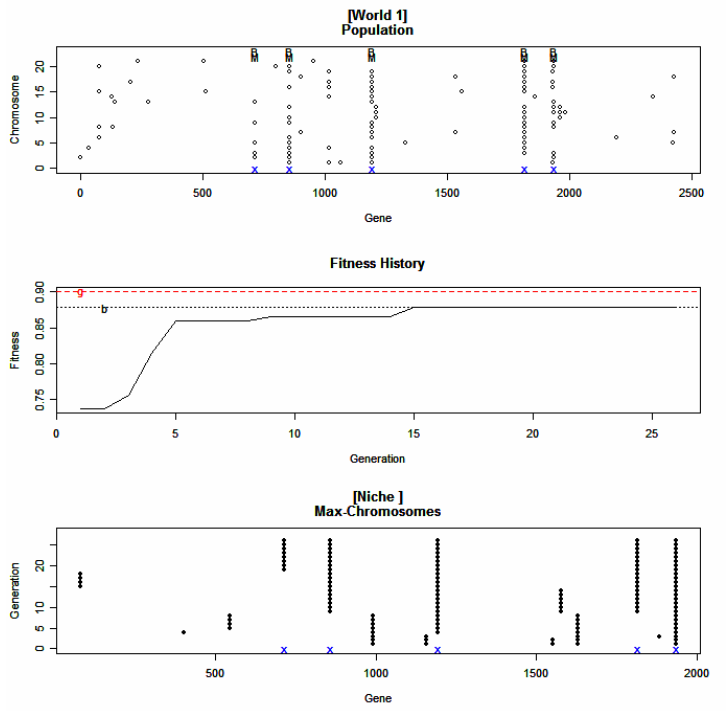

Figure 8 Real-time monitoring of the Genetic Algorithm search. The horizontal axis of the top and bottom plots display unranked gene indexes. The vertical axis of the top panel is displaying the chromosome index whereas the vertical axis of the bottom panel is displaying the generation number. In the middle plot the horizontal axis is displaying the generation whereas the vertical axis is displaying the fitness value.

6.4 Step 3 - Analysis and refinement of Chromosome Populations

6.4.1 Are we getting solutions?

The first question we have to answer is whether we are getting acceptable solutions. By default, configBB.VarSel configures the BigBang object to save all chromosomes even if they didn't reach the goalFitness value. The reason is that we need to assess the success of the configured GA search under all searches, not only in those that reach solutions. We can analyze the success of the configured GA search by looking at the evolution of the fitness value across generations, using the code below.

```
> plot(bb.nc, type="fitness")
```

Figure 9 shows that in average, we are reaching a solution in generation 40. The blue lines show the average fitness for all chromosomes. The cyan line traces those that have not reached a goal in a given generation.

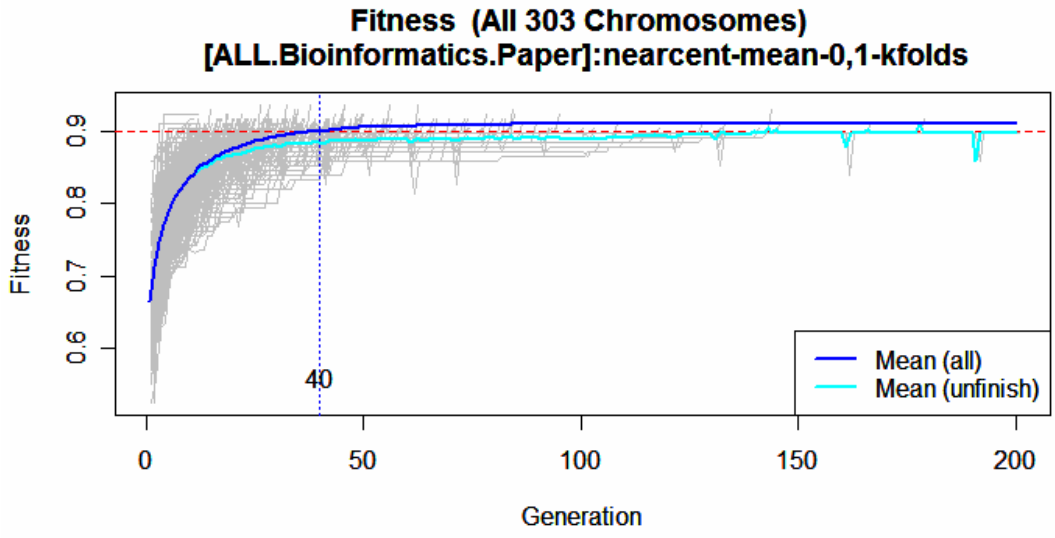

Figure 9 Evolution of the maximum fitness across generations in 303 independent searches.

It is possible to produce a plot that display the evolution of the fitness value for all cycles that have lead to a "fit" chromosome by typing:

```
> par(mfrow=c(2,1))
> plot(bb.nc, type="fitness", filter="solutions")
```

Similarly, we can plot the evolution of maximum fitness value of cycles that did not lead to the fitness goal.

```
> plot(bb.nc, type="fitness", filter="nosolutions")
```

The "filter" parameter is general and can be used in most functions in GALGO.

6.4.2 What is the overall accuracy of the population of selected models?

Once the chromosomes have been selected we need to asses the classification accuracy of the corresponding models using one of the three Strategies that we describe in **BOX 1**. The default configuration will estimate the accuracy of the models using Strategy 3 (Figure 11C) as described in **BOX 1**.

Use the following command to plot the overall accuracy.

```
> plot(bb.nc, type="confusion")
```

The output of this function is shown in Figure 10. The horizontal axis represents the individual samples grouped according to the disease class whereas the vertical axis represents the predicted classes. The barcharts represent the percentage of models that classify each sample in a given class. For example, samples in second column (marked in red) belong to the HYP+50 class. These are, on average, correctly

classified 85.6% of the times. However, on average, they are "wrongly" classified 2.5% of the times as EMLLA[i], 5.4% of the times as MLL, 1.5% as T, and 5% as TEL. The plot also report the value of sensitivity and specificity of the prediction. These are measures of the overall prediction per class. The sensitivity of the prediction for a given class k is defined as the proportion of samples in k that are correctly classified. The specificity for a given class k is defined as the number of true negatives divided by the sum of true negatives and false positives.

To obtain the confusion matrix, specificity, and sensitivity measures in a numeric format use the following code.

```
> cpm <- classPredictionMatrix(bb.nc)
> cm <- confusionMatrix(bb.nc, cpm)
> sec <- sensitivityClass(bb.nc, cm)
> spc <- specificityClass(bb.nc, cm)
```

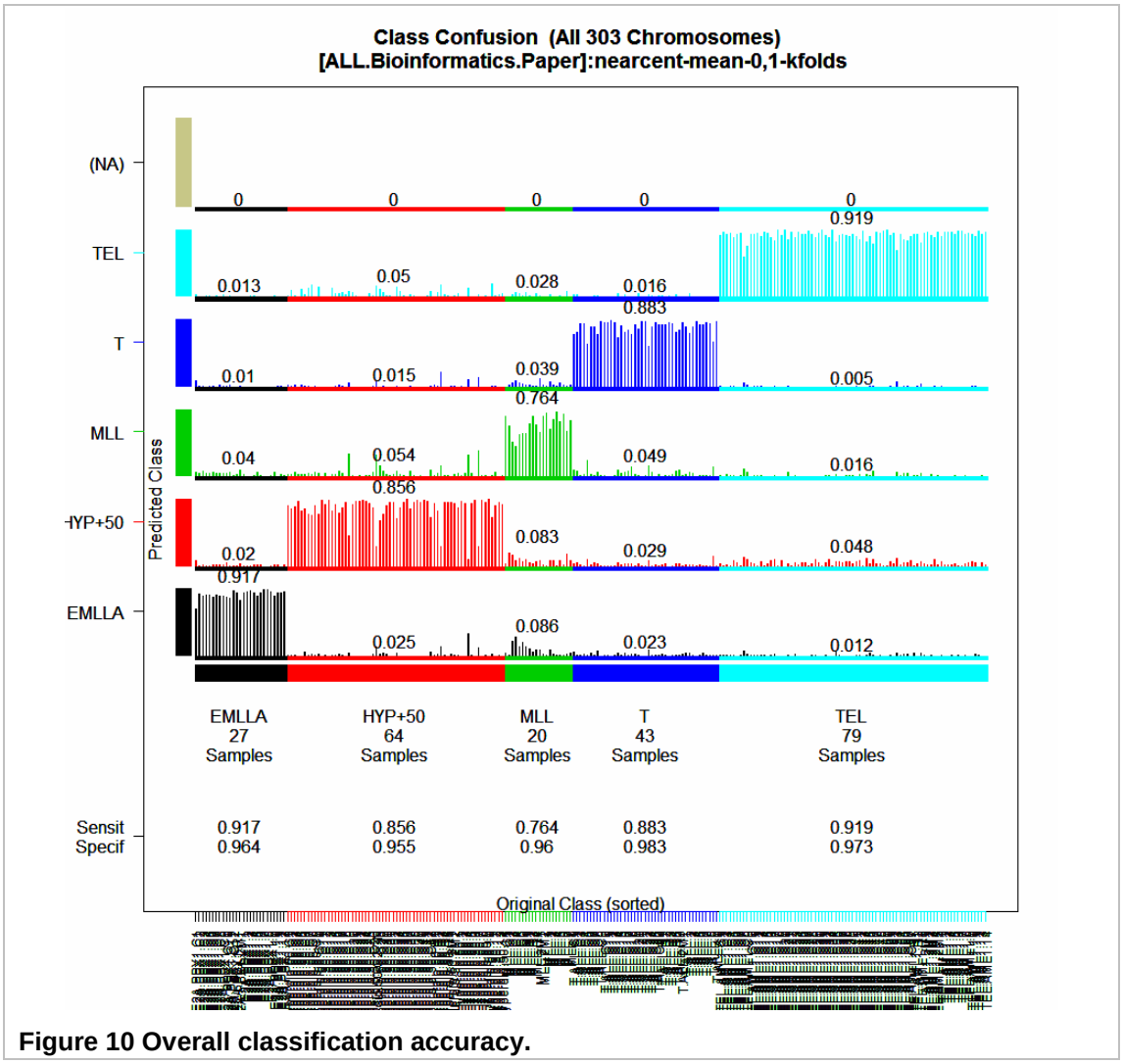

Figure 10 Overall classification accuracy.

BOX 1: Error estimation Strategies in GALGO

There are several methods to estimate Classification accuracy. These are all based on the fundamental principle that a correct estimate of accuracy must be performed on a set of samples that has not been used to develop the model itself.
Classical approaches involve splitting data in training and test sets. The training set is used to estimate the parameters of the model whereas the test set is left aside and it is used to asses the accuracy of the model itself. This approach is considered the most appropriate when a large number of samples is available. However, when the number of samples is relatively small, as it is the case of a typical microarray experiment, the test set could be too small to estimate the classification accuracy with acceptable precision.

In order to estimate the accuracy with small datasets it is possible to use a different statistical technique called *cross-validation*. The dataset is split in *k* different training and test sets. The classification accuracy is then defined as the average of the classification accuracies calculated, by default, on the test sets for each of the *k* splits. GALGO uses a technique called bootstrapping (Efron *et al.,* 1993) to generate the splits. Within GALGO we can use three main strategies for estimating classification accuracy. In the first strategy a simple cross-validation or resubstitution error strategy is used to compute the value of the fitness function that guide chromosome selection in the GA procedure. The classification accuracy of the selected chromosome is defined as the fitness value (Figure 11A). The second strategy (Figure 11B) is a classic Training and Test procedure where the accuracy is estimated on the test data.

In the GA process, the value of the fitness function is estimated by cross validation on the training data. Other approaches, such as .632 bootstrap (Efron *et al.*, 1993), combine training and test accuracies, which can be specified as error weights through the parameter *classification.test.error = c(.368, .632)* for training and test respectively. The third strategy is to select the Chromosomes as in the second strategy and to compute the classification accuracy of the selected chromosomes as the average of the classification accuracy estimated on *k* data splits as exemplified in Figure 11C.
GALGO defines the initial split (common to both strategies) as Split 1.

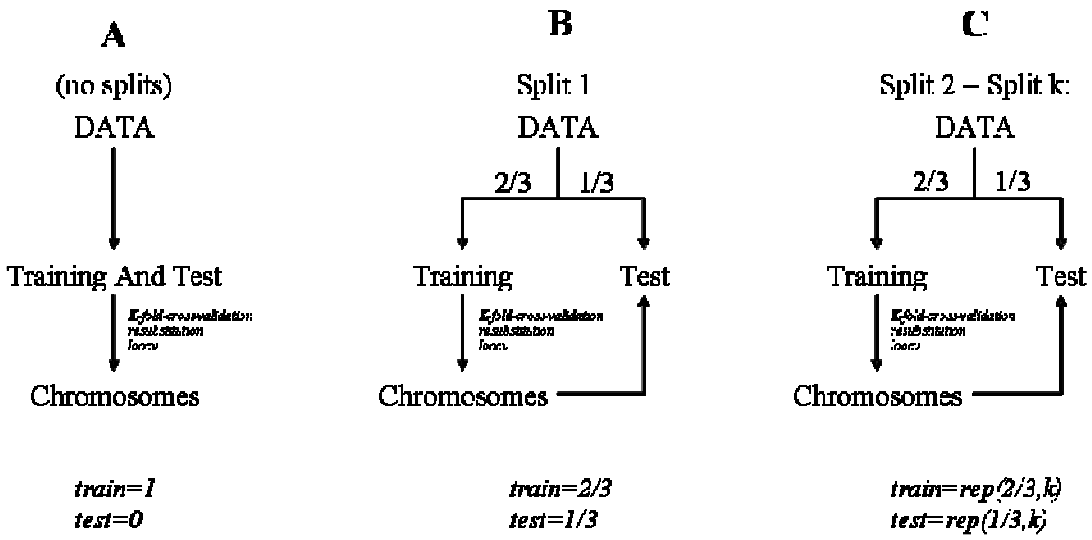

Figure 11 Schematic Representation of the Estimation of Classification Accuracy. (A) Strategy 1, using all data as training and test. (B) Strategy 2, classical training and test. (C) Strategy 3, *k* repetitions of the strategy 2. The respective values of the parameters, *train* and *test,* needed to perform each strategy is shown at the bottom of the schema.

6.4.3 Is the rank of the genes stable?

Stochastic searches (such as GA) are very efficient methods to identify solutions to an optimization problem (e.g. classification). However they are exploring only a small portion of the total model space. The starting point of any GA search is a random population. Different searches therefore are likely to provide different solutions. In order to extensively cover the space of models that can be explored it is necessary to collect a large number of chromosomes. GALGO offers a diagnostic tool to determine when the GA searches reach some degree of convergence. Our approach is based on the analysis of the frequency that each gene appears in the chromosome population. As chromosomes are selected the frequency of each gene in the population will change until no new solutions are found. Therefore monitoring the stability of gene ranks (based on their frequency) offers the possibility to visualize model convergence.

To produce the rank stability plot type:

```
> plot(bb.nc, type="generankstability")
```

By default, the most frequent 50 genes are shown in 8 different colours with about 6 or 7 genes per colour (Figure 12). Horizontal axis in Figure 12 shows the genes ordered by rank. Vertical axis shows the gene frequency (in the top part of the y axis) and the colour coded rank of each gene in previous evolutions. Consequently, for a given gene, changes in ranks are marked by different colours (below the frequency). Figure 12 shows that the first 7 black genes have been stable at least during the last 50 solutions whereas some red genes have recently swap from green. Thus, red and green genes are not yet stable; this is because 303 chromosomes are not enough to stabilize these genes. Probably, 1000 chromosomes would generate more stable results, however, the more chromosomes the better. For comparison, Figure 13 shows the result for the same run used here but using 1000 chromosomes, which exhibit more in ranks. Another property is that top genes are being stabilized in order; first black genes, then red, green and so on. For longer runs comparisons, see further sections.

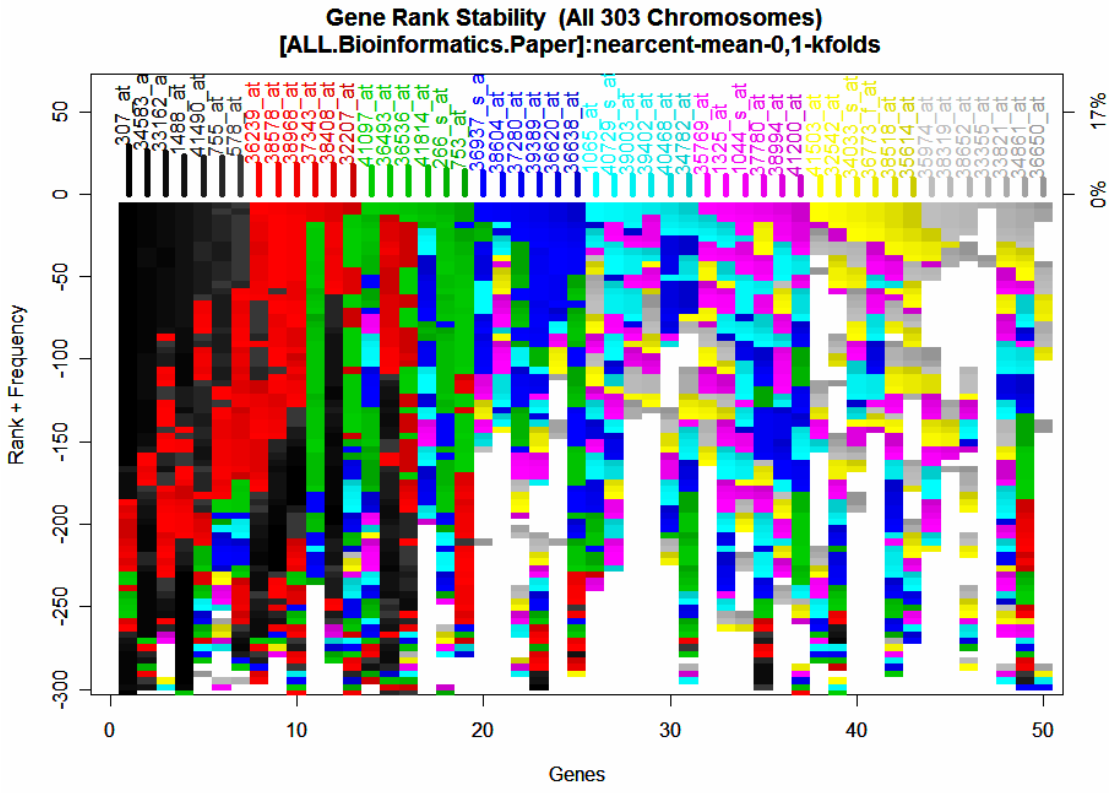

Figure 12 Gene Ranks across past evolutions.

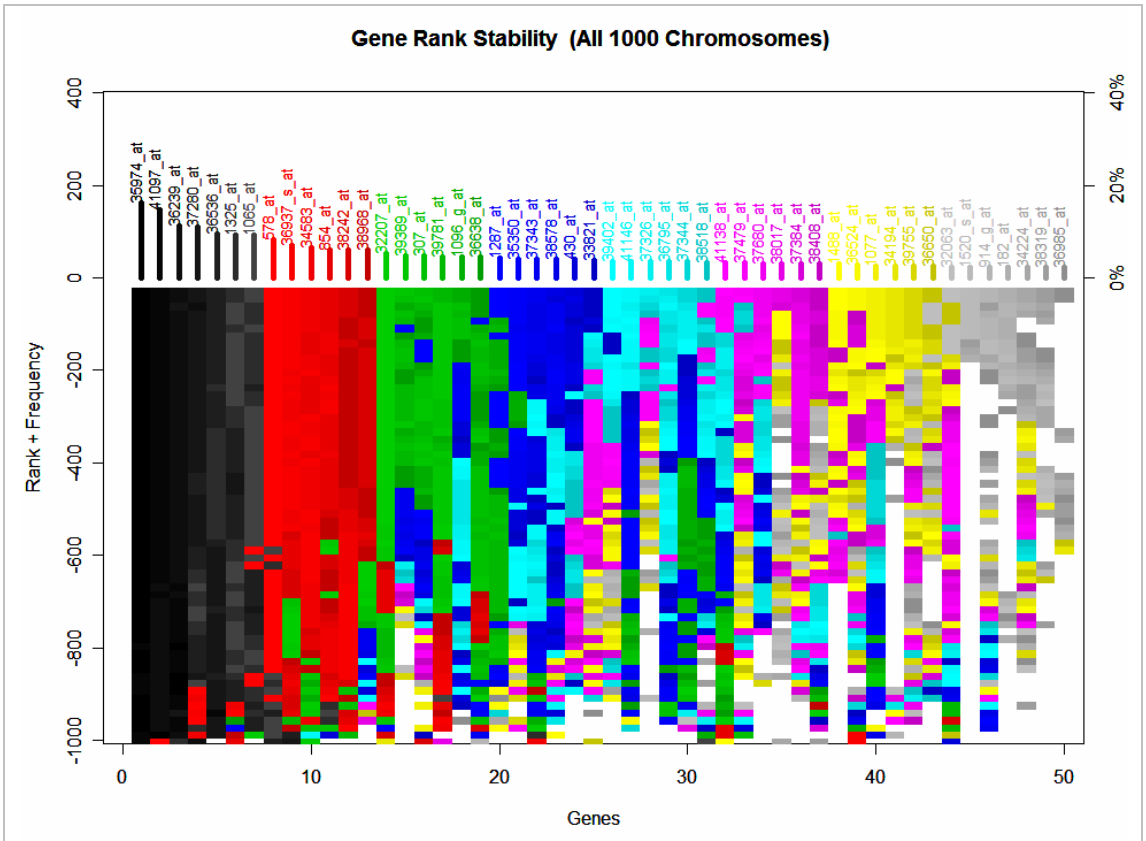

Figure 13 Rank Stability in 1000 chromosomes.

6.4.4 Are all genes included in a chromosome contributing to the model accuracy?

The chromosome size is fixed by an initial parameter in GALGO. This implies that some of the genes selected in the chromosome could not be contributing to the classification accuracy of the correspondent model. GALGO offers the possibility to identify these genes and remove them from the chromosomes. This can be done after the selection is completed or within the selection process itself. In order to perform this task we have implemented a backward selection procedure. The methodology works as follows. A given gene is removed from the chromosome. The classification accuracy of the resulting shorter chromosome is then computed. If this is not reduced, another elimination cycle is performed. If the Classification accuracy is reduced the gene is left in the chromosome and another elimination cycle is performed until all genes have been tested.

In order to perform this procedure type:

```
> rchr <- lapply(bb.nc$bestChromosomes[1:300],
robustGeneBackwardElimination, bb.nc, result="shortest")
```

The distribution of the size of the refined chromosome population can be plotted using the following function.

```
> barplot(table(unlist(lapply(rchr,length))),
main="Length of Shortened Chromosomes")
```

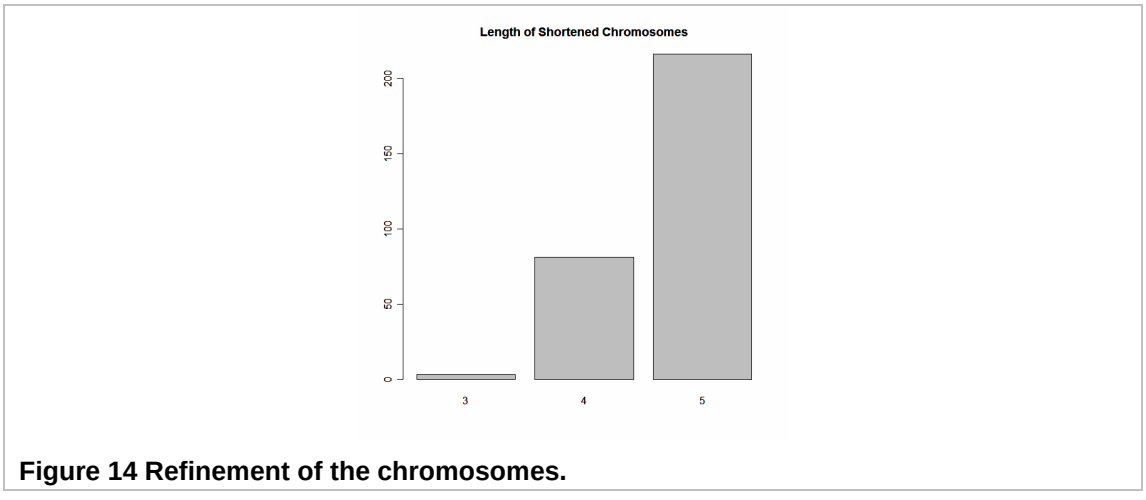

Figure 14 Refinement of the chromosomes.

The plot shows that a large proportion of the chromosomes require all five genes to accurately classify the samples.

6.5 Step 4 - Developing Representative Models

The GA procedure provides us with a large collection of chromosomes. Although these are all good solutions of the problem, it is not clear which one should be chosen for developing a classifier, for example, of clinical importance or for biological interpretation. For this reason there is a need to develop a single model that is, to

some extent, representative of the population. The simpler strategy to follow is to use the frequency of genes in the population of chromosomes as criteria for inclusion in a forward selection strategy. The model of choice will be the one with the highest classification accuracy and the lower number of genes. However GALGO also stores alternative models with similar accuracy and larger number of genes. This strategy ensures that the most represented genes in the population of chromosomes are included in a single summary model.

This procedure should be applied to the population of chromosomes generated by initial GA search. However, it can also be applied to the population of chromosomes that is the result of backward selection procedure explains in the previous paragraph.

The forward selection model can be generated by typing:

```
> fsm <- forwardSelectionModels(bb.nc)
> fsm$models
> ?forwardSelectionModels.BigBang # Help System
```

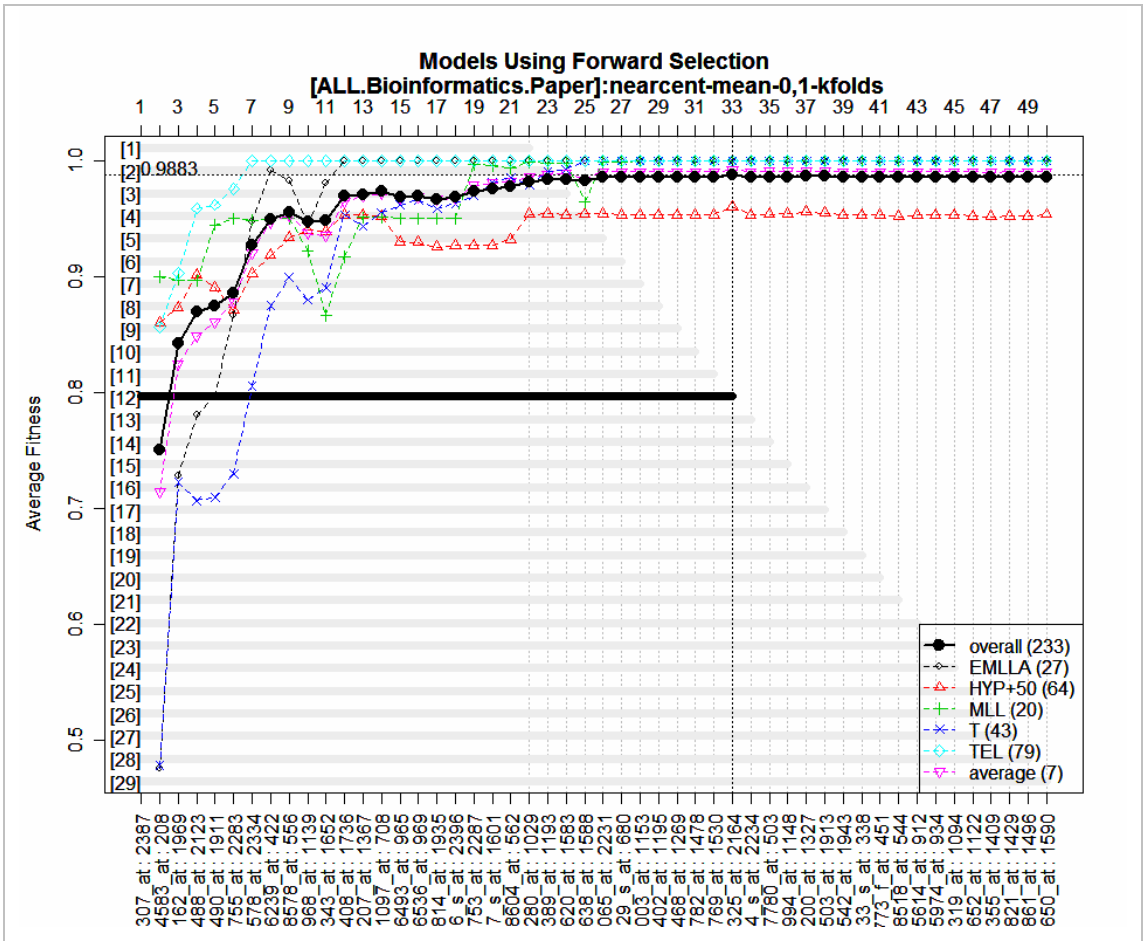

Figure 15 Forward selection using the most frequent genes. Horizontal axis represents the genes ordered by their rank. Vertical axis shows the classification accuracy. Solid line represents the overall accuracy (misclassified samples divided by the total number of samples). Coloured dashed lines represent the accuracy per class. 1 model resulted from the selection whose fitness value is maximum (black thick line), but 29 models were finally reported because they were very similar in absolute value.

Figure 15 shows the result of the ***best*** models from the most frequent genes using a forward selection strategy. Model labelled as 12, containing the most 33 frequent genes, was the best model in terms of accuracy. To visualize this model in a heatmap plot use the following code.

```
> heatmapModels(bb.nc, fsm, subset=12)
```

Details for visualization of models (or chromosomes) are given in section 6.6 and GALGO manual.

In order to generate the forward selection model of the population of refined chromosomes is slightly more complex. Firstly, we need to compute the gene frequency using the refined chromosomes, as follows.

```
> rchr <- lapply(bb.nc$bestChromosomes[1:300],
robustGeneBackwardElimination, bb.nc, result="shortest")
> rgf <- compCount(rchr, bb.nc$saveGeneBreaks)
> rfsm <- forwardSelectionModels(bb.nc,
geneIndexSet=order(rgf,decreasing=TRUE)[1:50])
```

The gene signatures associated to the resulting model(s) can be visualised using heat maps, PCA plots, or gene expression profiles (see section 6.6).

The classification accuracy of the summary models can be plotted using the code below:

```
> plot(bb.nc, type="confusion",
chromosomes=list(fsm$models[[1]]))
```

Or in a tabular format:

```
> cpm.1 <- classPredictionMatrix(bb.nc,
chromosomes=list(fsm$models[[1]]))
> cm.1 <- confusionMatrix(bb.nc, cpm.1)
```

Semsitivity and specificicy can also be computed using the following commands:

```
> mean(sensitivityClass(bb.nc, cm.1))
[1] 0.9863334
> mean(specificityClass(bb.nc, cm.1))
[1] 0.9965833
```

6.6 Visualizing Models and Chromosomes

Gene signatures associated within individual chromosomes or in a representative model (derived by forward selection) can be visualised in GALGO using a number of graphical functions. In this section, we will demonstrate the use of heat maps and PCA. For, the typical heat map format, use the following commands.

```
> heatmapModels(bb.nc, fsm, subset=1) # forward
> heatmapModels(bb.nc, bb.nc$bestChromosomes[1])
```

The results are shown in Figure 16*.

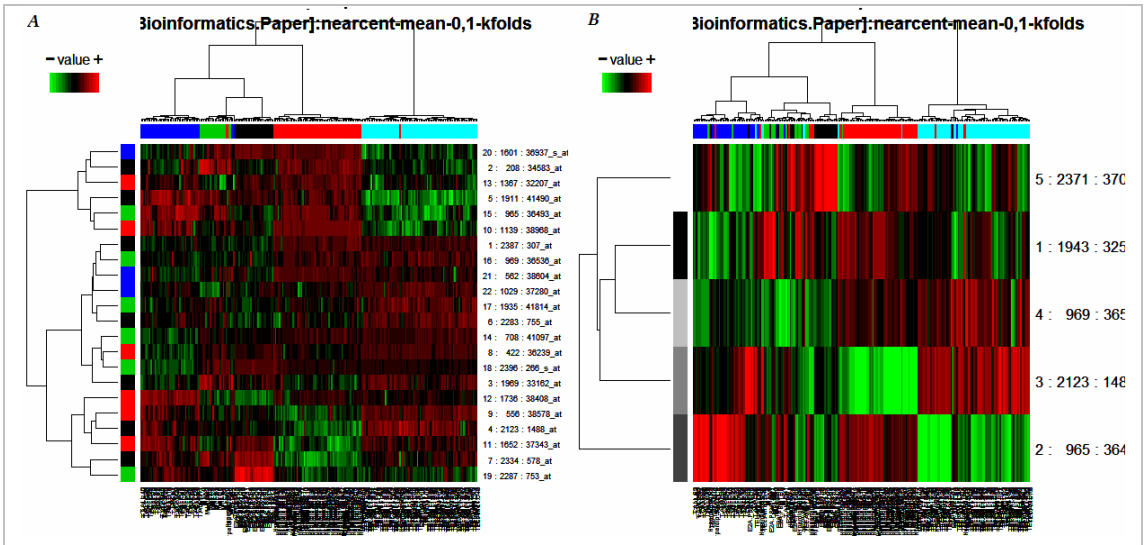

Figure 16 Heatmaps. From a model resulted from forward selection (A) and an original evolved chromosome (B).

In order to visualise the relation of samples using the genes selected in a chromosome or in a representative model we can also use principal component analysis representation. In order to do this, type the following command (Figure 17).

```
> pcaModels(bb.nc, fsm, subset=1)
> pcaModels(bb.nc, bb.nc$bestChromosomes[1])
```

By default, only the first four components are shown, which can be changed specifying the *npc* parameter.

* Remember that the hierarchical clustering of samples given in the heat map is the product of an unsupervised algorithm, which may differ from the classification method of our choice. Therefore, the relative sample order in the heat map, the original class, and the predicted class by the model may all be different. Nevertheless, many of the times, the hierarchical clustering gives a good overview.

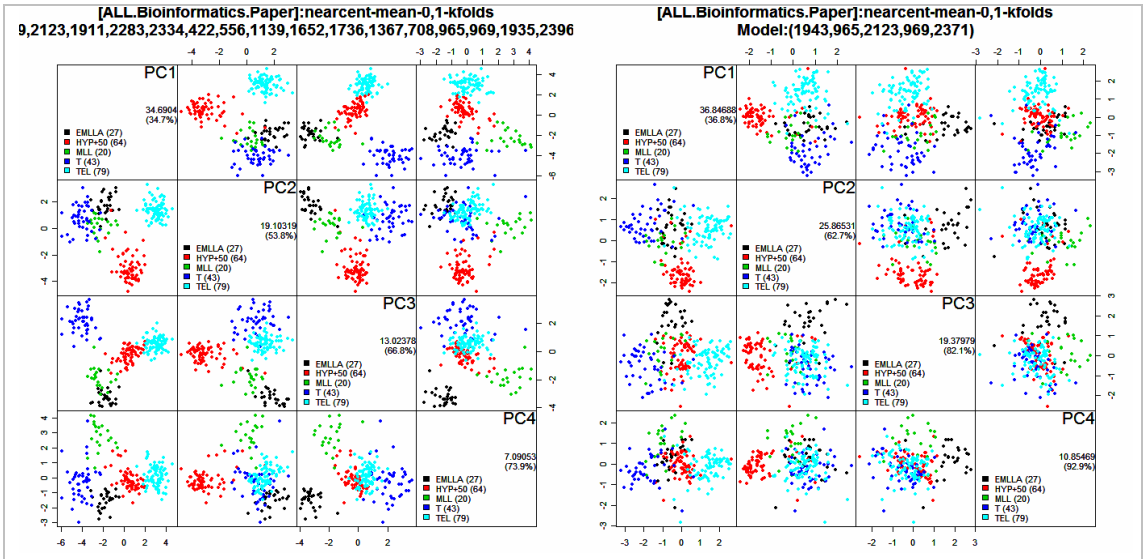

Figure 17 Depiction of a model (left) and a chromosome (right) in PCA space.

6.7 Predicting Class Membership of Unknown Samples

An important feature of models developed using GALGO is their ability to make predictions. Models developed in GALGO can be used to predict class membership from an independent set of samples. The following code exemplify how make predictions in a new or independent dataset for all chromosomes collected in the BigBang object. This example uses a dummy dataset collected from the same data for illustrative purposes.

```
> data(ALL)
# dummy data: the first 15 samples from original ALL data
# which must be EMLLA(E2A-PBX) class
> dummy <- ALL[,1:15]
> ?predict.BigBang
> cpm <- predict(bb.nc, newdata=dummy,
func=classPredictionMatrix, splits=1:10)
> cpm
> plot(bb.nc, cpm, type="confusion")
```

In the above code, *dummy* was temporally appended to the original data. Then *classPredictionMatrix* was run for all chromosomes. *splits* is a parameter used in *classPredictionMatrix* (which was used to illustrate the use of user parameters for any function specified in *func*). The result of the plot is shown in Figure 18 where the new data was labelled as “UNKNOWN”. The black bars in these samples indicate that they were predicted as EMLLA^{i} (as expected).

To predict new data using an individual model, we may use the *classPredictionMatrix* method using the *chromosomes* parameter (see ?classPredictionMatrix.BigBang), such as in the following code.

```
> cpm <- predict(bb.nc, newdata=dummy,
func=classPredictionMatrix, chromosomes=fsm$models[1])
> cpm
> plot(bb.nc, cpm, type="confusion")
```

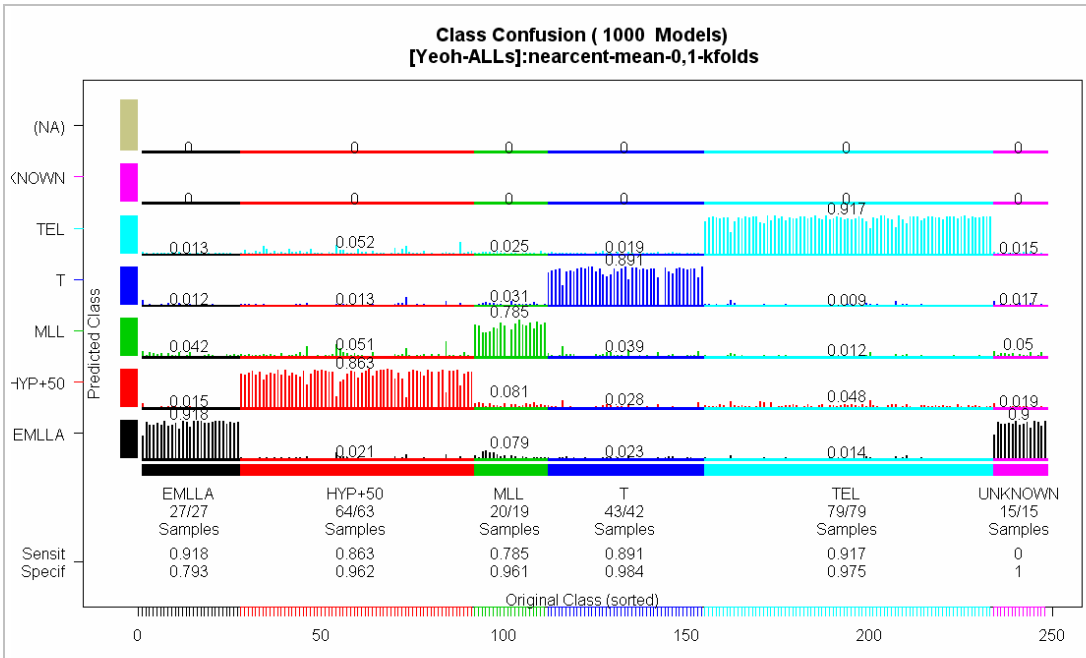

Figure 18 Class prediction for unknown samples (the last 15 samples in the right).

6.8 Biological Interpretation of Genes Selected in Models

Statistical models can be used for classification purposes (e.g. to identify diagnostic markers) but are also a useful source of biological information. For this reason the biological interpretation of these models can provide an insight in the molecular mechanisms behind the biological system. In order to identify associations between the collections of models with functional pathways we have mapped the 50 most represented genes in the model population on the Ingenuity database using the web-based Ingenuity pathway analysis tool (Ingenuity® Systems, www.ingenuity.com). This database store maps of canonical functional pathways and functional relationships supported by published literature and by protein-protein interaction data.We found that genes selected in our models are significantly enriched in four interaction networks (table 1). Table 1 also show functions that are significantly enriched. Table 1 show that all networks are characterised by cell death, cellular growth, and cellular division. A particularly interesting network (network 1 in table 1 and Figure 19) connects IL-1β to five other genes that are selected with high frequency in the models. IL-1β is an essential regulator of proliferation in leukaemia and has been demonstrated that the ability of cells to respond to this stimuli is predictive of disease outcome (Ezaki *et al.*, 1995; Hulkkonen *et al.*, 2000). These results exemplify the interpretation of classification models can be used to formulate biologically interesting hypothesis.

#	*Genes*	*Score*	*Selected Genes*	*Top Functions*
1	BAK1, BCL2, BNIP3, CABP1, CD2, **CD24, CD44, CTGF**, defb4 (human), ENPP1, **FLT3**, FLT3LG, GCH1, H2-D1, IL15, **IL1B**, IL1RAP, **INSR, IRF4, ITPR3**, **MX1**, NFATC2, OGN, PDCD4, PHLDA1, **PRKCQ**, PTP4A1, **PTPRK**, RPS3A, SEPP1, SF1, SLC20A1, **TCFL5**, **TM4SF2**, ZFP36L1	21	13	Cell Death, Cellular Development, Cellular Growth and Proliferation
2	**ALOX5**, ALOX5AP, **CBFA2T3**, CCNE1, **CD3E**, CDKN2B, CTNNA1, **E2F5**, **ERG**, F13A1, FLT3LG, FOXG1B, FUT7, HDAC1, HOXD3, **IGFBP7**, IGL@, IL4, **ITPR1**, MEFV, NEDD9, **PBX1**, PHB, PKNOX1, **POU2AF1**, POU2F2, **PRKCH**, **RAG1, RAG2**, SCN2A1, SMAD5, SMAD6, SPI1, TGFB1, TGIF	19	12	Cellular Development, Cell Death, Hematological System Development and Function
3	BIK, COL4A1, ENPP2, **FNBP1**, H2-D1, HAMP2, HAS1, HPSE, IL6, IL13, IL1F6, LIR9, **LTB**, **LY86**, MYOD1, NEDD9, P53AIP1, PARP2, PSMB4, **PTP4A3**, **PTPRE**, **SCHIP1**, SIVA, **SLA**, Slco1a4, **SMAD1**, SRC, **TERF2**, TERF2IP, TINF2, TNF, **TNFRSF7**, TNFSF7, TP53, XRCC5	15	10	Cell Cycle, Cell Death, Hematological Disease
4	**ACTN1**, APS, BLK, **BLNK**, BTK, CAMK2A, **CD9**, **CD19**, CD22, CD38, **CD72**, CD81, CD79A, CD79B, CDK5R1, CKLFSF3, CR2, GRN, HRAS, ITGA1, ITGB4, **LRMP**, MAPK1, MCP, NR0B2, **PDLIM1**, PLCG2, **PPM1F**, PTGFRN, RHOB, SCNN1A, **SH3BP5**, SPARC, VAV2, VIM	13	9	Hematological System Development and Function, Immune and Lymphatic System Development and Function, Tissue Morphology

Table 1. Significant Networks.

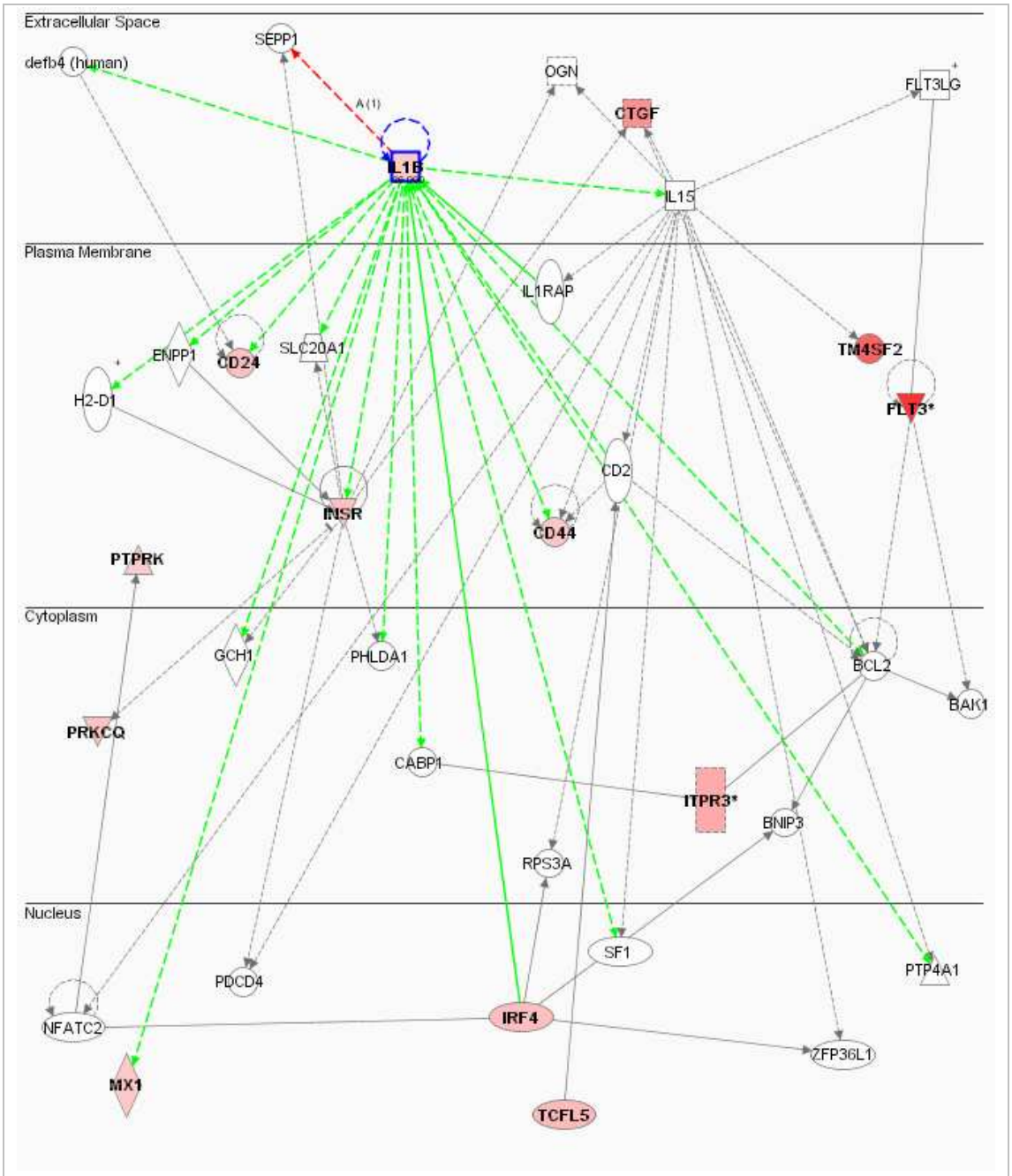

Figure 19 Interaction Network. IL-1β is bolded in blue whereas its interactions are highlighted in green. Genes shaded in red are included in the top 50 representative genes. Interaction are interpreted both as protein-protein interactions and transcription interaction (such as transcription factors) supported by literature. Network 1 from table 1.

Appendix A
A Comparison between GALGO and Univariate Variable Selection Methods

A.1 Background

The development of statistical models linking the molecular state of a cell to its physiology is one of the most important tasks in the analysis of Functional Genomics data. Because of the large number of variables measured a comprehensive evaluation of variable subsets cannot be performed with available computational resources. It follows that an efficient variable selection strategy is required. Variable selection strategies can be subdivided into univariate and multivariate methods. Univariate approaches test one feature at a time for their ability to discriminate a dependent variable. The top most significant features are then used to develop a statistical model (see schematic representation in Figure 1A). Multivariate approaches takes into consideration that variables are influencing a biological outcome in the context of networks of interacting genes rather than in isolation and can take into considerations synergy between genes, proteins, or metabolites. Although these approaches have been very successful there are still issues in the development of multivariate models from large datasets. These issues are related to the extremely large number of possible models that would need to be evaluated to identify the most predictive. In order to address these issues, a number of variable selection strategies have been developed and tested on functional genomics datasets. Among these Genetic Algorithms appear to be very efficient in large scale datasets. For this reason we have developed GALGO, a software environment to develop and evaluate multivariate statistical models that uses a genetic algorithm variable selection strategy (Figure 1B).
In this Appendix we report the results of a comparison between two common univariate variable selection strategies (F-statistic and d-statistic) with GALGO in association with a number of classification methods. The models we have developed have been analysed in respect to classification accuracy, number of genes required to achieve the highest classification accuracy and the identity of the genes selected in the models. In order to make sure that our comparison is of general validity we have used three different datasets.
Our results support the use of multivariate variable selection in developing statistical models and in particular support the use of GALGO as a general software environment for model selection.

A.2 Methods

Variable selection: In this comparison we have used three variable selection strategies. These are: The F test, d statistics and Genetic Algorithms.
Classification methods: We have compared F statistics and Genetic Algorithms with the following methods: 1) Diagonal Linear Discriminant Analysis (DLDA)*, 2) Support Vector Machines (SVM), 3) Random Forest (RF) and 4) K-Nearest-Neighbours (KNN). We have also compared the established tool PAM (d-statistics in combination with nearest centroid) with Genetic Algorithm in combination with nearest centroid (NC).

Construction of a representative model: To generate a representative model we used a Forward Selection (FS) strategy in both multivariate and univariate model selection. Briefly, an initial model is created using the first two genes from an ordered list of genes (using p-values or d-statistic for univariate variable selection or gene frequency using GALGO). Then, the model is assessed using a classification method to estimate the classification error. Subsequently, the model is lengthened with the next gene in the ordered list and the resulted model is re-assesed. This cycle continues until all genes in a list have been included. The model whose classification error is the lowest is chosen. In the case of draw, the smallest model is selected.
Method specific gene signatures: To determine the degree of overlap between the different models at a gene identity level we have build a pool gene set containing the genes from the models generated with all five methods. For each gene in a given model, we counted the number of times it appears in the pool set. Genes appearing only once were defined as model-specific.
Implementation: To develop classifiers with a univariate variable selction strategy (F- or d-statistics) we have used the Web based tool TNASAS (Vaquerizas et. al. 2005) (http://tnasas.bioinfo.cipf.es), which is part of the Gene Expression Pattern Analysis Suite (GEPAS, http://gepas.bioinfo.cipf.es). All classification methods tested in combination with genetic algorithms have been used in the GALGO implementation with default settings with the addition of a backward selection step for model enhancement (see paragraph 5.5 of the supplementary material for a description of the procedure). The representative models were developed from 1,000 chromosomes.

*** NOTE: The MLHD method we implemented in GALGO is equivalent LDA.**

A.3 Datasets

ALL-Subclasses Dataset (ALLS): This dataset, developed by Yeoh *et al.* (Yeoh *et al.* 2002), describes the expression profile of 327 acute lymphoblastic leukaemia (ALL) patients representing 7 different disease sub-classes. The authors have used Affymetrix GeneChips. In this comparison we have selected the five largest classes (EMLLA[i], Hyp+50, MLL, T, and TEL including respectively 27, 64, 20, 43, and 79 samples). The original dataset downloaded from http://www.stjuderesearch.org/data/ALL1/ comprising 12,600 genes have been filtered to eliminate the most invariant genes (original E2A-PBX class was renamed to EMLLA). The standard deviation and difference between maximum and minimum expression value were calculated for each gene. The genes were ranked by these values, and if they were within the top 15% for either, were selected for further analysis. The dataset after filtering contained the expression values for 2,435 genes.
ALL-AML Dataset (ALL/AML): This dataset, developed by Golub *et. al.* (Golub *et. al.*, 1999) describes the transcriptional state of 47 acute lymphoblastic leukaemia (ALL) and 25 acute myeloid leukaemia (AML) patients. Data were processed as in the original publication. Briefly, intensity values were re-scaled such that overall intensities for each chip are equivalent. This was done by fitting a linear regression model using the intensities of all genes with "P" (present) calls in both the first sample (baseline) and each of the other samples. The inverse of the "slope" of the linear regression line becomes the (multiplicative) re-scaling factor for the current sample.

This was done for every chip (sample) in the dataset except the baseline which gets a re-scaling factor of one.
A further processing step was performed to eliminate genes that were not detected in the majority of the samples. For this reason we filtered out every gene that were not expressed (Flagged as M or A) in more then 80% of the samples.
Breast Cancer Dataset (BC): This dataset was developed by van 't Veer *et al.* (van 't Veer *et al.* , 2002) and represents 78 patients subdivided in two groups with different clinical outcome (44 patients with no metastases developed within the first five years versus 34 patients positive for metastases within the first five years). Data were normalized as described in the original publication. Genes with a p-value (Confidence level that a gene's mean ratio is significantly different from 1) larger then 0.001 in all samples were filtered out.

A.4 Results

Table 1, 2 and 3 show the result of the analysis performed using GALGO with five different classification methods (NC, KNN, SVM, MLHD, RF) on three datasets (BC, ALL/AML, ALLS). The tables report the classification accuracy and model size for the best representative models developed using forward selection and for the top five individual chromosomes selected by the GA search. The table show that in all cases GALGO can identify accurate models of a relatively small size.

Table 4, 5, 6 and figures 1, 2, and 3 summarize the result of the comparison between GALGO and univariate variable selection strategies. In the ***Breast Cancer dataset*** (table 4 and figure 1) GALGO produced models with higher classification accuracy regardless of the classification method used. The size of the models developed with GALGO (table 4 and figure 1) was generally smaller than the models developed with the univariate variable selection strategy. The largest difference being the models developed with the KNN method (these require 2920 genes with univariate model selection and 31 genes with GALGO).

In the ***ALL-AML dataset*** (table 5 and Figure 2) the classification accuracy of models developed with univariate and multivariate models was comparable (Galgo gave models in the range between 3% and 10% of error whereas the univariate methods gave models with error in the range between 3% and 7 %). However, the models developed using GALGO were markedly smaller in size (a range of 4 to 49 genes respect to 79 to 1697 in the univariate variable selection) (table 5 and figure 2).

In the ***ALL-Subclasses dataset*** (table 6 and Figure 3) GALGO generated either model with comparable accuracy (the maximum difference in classification accuracy was 2%) or higher accuracy respect to univariate models (1% against 17% using Random Forest and 1% against 13% with NC). As in the other datasets the size of the models developed using GALGO was markedly smaller then models developed with Univariate methods (the range of model size in this dataset was between 4 and 49 whereas the range of model size in the univariate selected models was between 75 and 1697).
In two of the datasets the model size is dramatically different making obvious that multivariate models are very effective in identifying different gene subsets. In the Breast Cancer dataset gene sets are of a more comparable size. Table 7 summarize the overlap in gene composition of the models developed with the different methods in

Breast Cancer dataset. These results suggest that multivariate model selection tend to give very different gene subsets respect to the univariate variable selection strategy. In interpreting these results however we should take into account that the classification error of models developed from univariate variable selection strategies was very low.

A.5 Discussion

The models we have developed have been analysed in respect to classification accuracy, number of genes required to achieve the highest classification accuracy and the identity of the genes selected in the models. All these factors are important in determining the usefulness of a methodology. High classification accuracy it is obviously a very desirable property but in order for the models to be biologically interpretable and of practical use it is also important that the gene set is a manageable size. The identity of the genes is also a very important factor. One of the reasons why multivariate methods may be a good option is that they allow the identification of genes that contribute to a biological effect in association. These could not be discovered by univariate variable selection methods where every gene is tested in isolation. If univariate and multivariate approaches provide models with comparable classification accuracy but with different genes then the two approaches have to be considered complementary as they are likely to represent different underlying biological processes.

Our results shows that the methodology we have implemented in the R package GALGO tends to produce models with comparable or better classification accuracies respect to univariate variable selection strategies. The multivariate selected models generally use a smaller number of genes than univariate models in all datasets and with all the methods we have tested. This results support the use of a multivariate model selection strategy in the analysis of functional genomics data and in particularly support GALGO as a general tool.

BREAST CANCER (2 Classes)										
Method	*KNN*		*SVM*		*NC*		*MLHD*		*RF*	
Model	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*
GA+FS 1st	32	0.16	12	0.17	35	0.15	4	0.18	47	0.17
GA+FS 2nd	33	0.16	9	0.18	11	0.15	-	-	37	0.18
GA 1st	5	0.20	5	0.17	5	0.18	5	0.17	5	0.18
GA 2nd	5	0.21	5	0.18	5	0.19	5	0.18	5	0.24
GA 3rd	5	0.22	5	0.19	5	0.19	5	0.18	5	0.24
GA 4th	5	0.22	5	0.19	5	0.19	5	0.19	5	0.25
GA 5th	5	0.22	5	0.20	5	0.19	5	0.19	5	0.25
GA+BE+FS 1st	31	0.15	12	0.17	23	0.14	4	0.18	40	0.18
GA+BE+FS 2nd	32	0.15	-	-	9	0.15	-	-	14	0.19
GA+BE 1st	5	0.21	5	0.17	3	0.17	4	0.17	4	0.18
GA+BE 2nd	3	0.21	4	0.17	4	0.17	3	0.17	3	0.23
GA+BE 3rd	4	0.21	2	0.18	3	0.18	4	0.17	2	0.24
GA+BE 4th	4	0.21	2	0.18	5	0.18	4	0.18	4	0.24
GA+BE 5th	3	0.22	2	0.18	4	0.18	3	0.19	3	0.24

Table 1. (Appendix)

Abbreviations for all tables:
GA – Genetic Algorithms, FS – Forward Selection, BE – Backward Elimination. DLDA – Diagonal Linear Discriminant Analysis, PAM – Shrunken Centroids, PAMR – Shrunken Centroids R package, KNN – K-Nearest-Neighbours, SVM – Support Vector Machines, NC – Nearest Centroid, MLHD – Maximum Likelihood Discriminant Functions, RF – Random Forest.

ALL-AML Dataset (2 Classes)										
Method	*KNN*		*SVM*		*NC*		*MLHD*		*RF*	
Model	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*
GA+FS 1st	42	0.06	50	0.07	37	0.05	9	0.14	47	0.08
GA+FS 2nd	37	0.06	23	0.07	24	0.06	17	0.14	45	0.08
GA 1st	5	0.11	5	0.07	5	0.13	5	0.10	5	0.12
GA 2nd	5	0.12	5	0.11	5	0.15	5	0.10	5	0.15
GA 3rd	5	0.13	5	0.11	5	0.15	5	0.11	5	0.15
GA 4th	5	0.13	5	0.12	5	0.15	5	0.12	5	0.16
GA 5th	5	0.13	5	0.13	5	0.15	5	0.12	5	0.16
GA+BE+FS 1st	45	0.04	25	0.06	29	0.03	13	0.12	49	0.07
GA+BE+FS 2nd	40	0.05	24	0.06	27	0.03	34	0.13	32	0.08
GA+BE 1st	3	0.08	4	0.07	2	0.12	4	0.10	2	0.12
GA+BE 2nd	3	0.09	3	0.11	2	0.12	5	0.10	4	0.14
GA+BE 3rd	3	0.11	5	0.11	5	0.13	4	0.11	4	0.15
GA+BE 4th	3	0.11	3	0.12	4	0.14	4	0.11	5	0.15
GA+BE 5th	4	0.12	3	0.12	3	0.15	2	0.11	4	0.16

Table 2 (Appendix)

ALL-Subclasses Dataset (5 Classes)										
Method	*KNN*		*SVM*		*NC*		*MLHD*		*RF*	
Model	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*	*Size*	*Error*
GA+FS 1st	47	0.00	10	0.02	50	0.01	23	0.01	14	0.01
GA+FS 2nd	13	0.01	9	0.03	16	0.02	15	0.02	10	0.02
GA 1st	5	0.06	5	0.05	5	0.06	5	0.06	5	0.08
GA 2nd	5	0.06	5	0.05	5	0.07	5	0.06	5	0.08
GA 3rd	5	0.06	5	0.05	5	0.07	5	0.06	5	0.08
GA 4th	5	0.06	5	0.06	5	0.07	5	0.06	5	0.08
GA 5th	5	0.06	5	0.06	5	0.07	5	0.06	5	0.09
GA+BE+FS 1st	47	0.00	10	0.02	50	0.01	20	0.01	19	0.01
GA+BE+FS 2nd	13	0.01	9	0.03	16	0.02	15	0.02	10	0.02
GA+BE 1st	4	0.06	5	0.05	5	0.06	5	0.06	4	0.08
GA+BE 2nd	5	0.06	5	0.05	4	0.07	5	0.06	4	0.08
GA+BE 3rd	4	0.06	5	0.06	5	0.07	5	0.06	4	0.08
GA+BE 4th	5	0.06	5	0.06	5	0.07	4	0.06	5	0.09
GA+BE 5th	4	0.06	4	0.06	5	0.07	5	0.06	5	0.09

Table 3 (Appendix)

BREAST CANCER (2 Classes)			
	Model Selection	*Optimal Model*	
Gene Selection	*+Classifier*	*Size*	*Error*
F-statistic (univariate)	FS+DLDA	2	0.32
F-statistic (univariate)	FS+SVM	5	0.36
F-statistic (univariate)	FS+RF	10	0.35
F-statistic (univariate)	FS+KNN	2920	0.44
d-statistic (univariate)	FS+PAM	51	0.36
GALGO+MLHD (multivariate)	BE+MLHD	4	0.17
GALGO+SVM (multivariate)	SVM	5	0.17
GALGO+RF (multivariate)	FS+RF	47	0.17
GALGO+KNN (multivariate)	BE+FS+KNN	31	0.15
GALGO+NC (multivariate)	BE+FS+NC	23	0.14

Table 4 (Appendix)

ALL-AML (2 Classes)			
	Model Selection	*Optimal Model*	
Gene Selection	*+Classifier*	*Size*	*Error*
F-statistic (univariate)	DLDA	500	0.06
F-statistic (univariate)	SVM	120	0.04
F-statistic (univariate)	RF	500	0.03
F-statistic (univariate)	KNN	75	0.07
d-statistic (univariate)	PAM	1697	0.06
GALGO+MLHD (multivariate)	BE+MLHD	4	0.10
GALGO+SVM (multivariate)	BE+FS+SVM	25	0.06
GALGO+RF (multivariate)	BE+FS+RF	49	0.07
GALGO+KNN (multivariate)	BE+FS+KNN	45	0.04
GALGO+NC (multivariate)	BE+FS+NC	29	0.03

Table 5 (Appendix)

ALL-Subclasses (5 Classes)			
Gene Selection	*Model Selection +Classifier*	*Optimal Model Size*	*Error*
F-statistic (univariate)	DLDA	200	0.02
F-statistic (univariate)	SVM	75	0.00
F-statistic (univariate)	RF	1000	0.17
F-statistic (univariate)	KNN	120	0.01
d-statistic (univariate)	PAM	439	0.13
GALGO+MLHD (multivariate)	BE+FS+MLHD	23	0.01
GALGO+SVM (multivariate)	FS+SVM	10	0.02
GALGO+RF (multivariate)	FS+RF	14	0.01
GALGO+KNN (multivariate)	FS+KNN	47	0.01
GALGO+NC (multivariate)	FS+NC	50	0.01

Table 6 (Appendix)

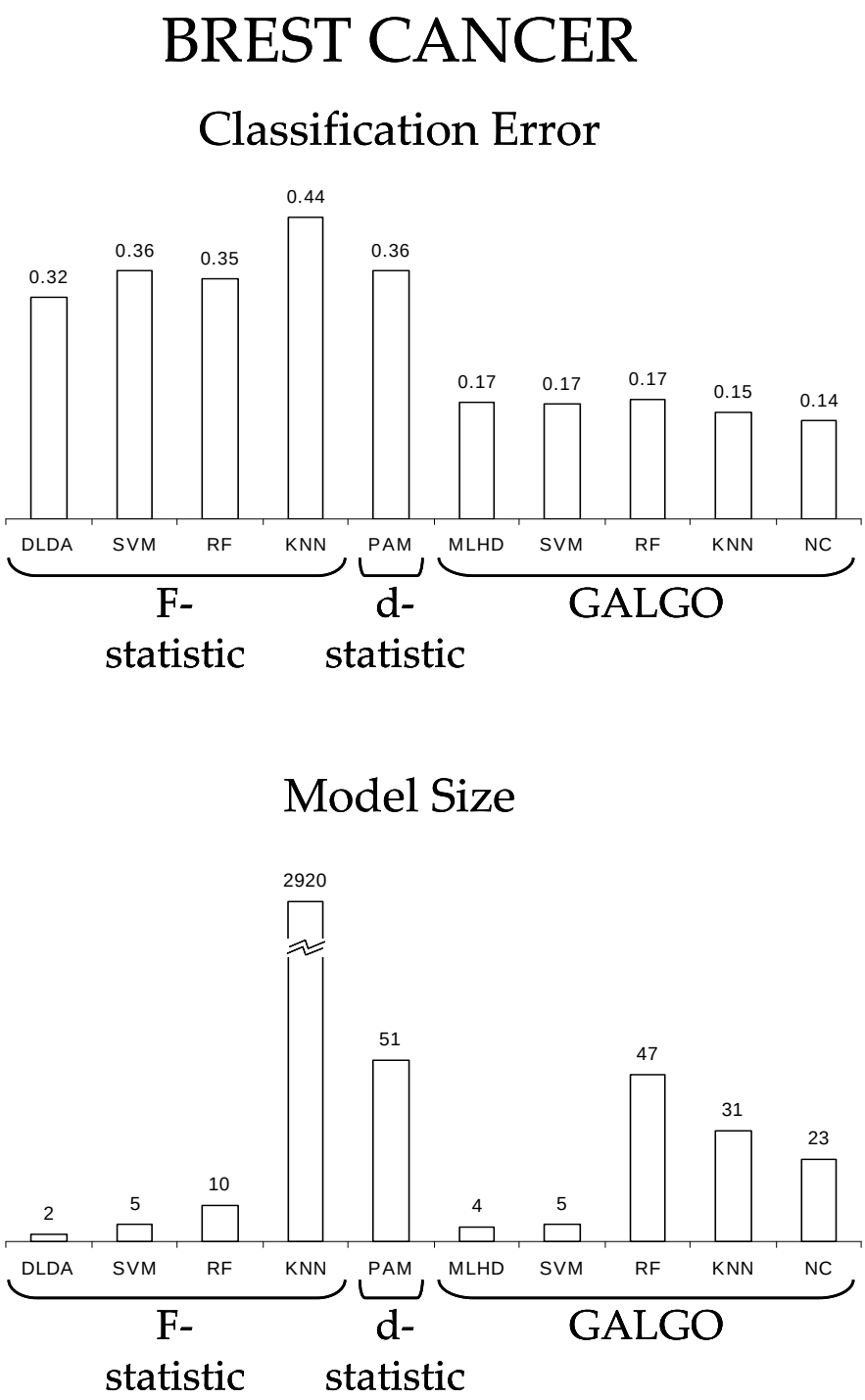

Figure 1 (Appendix)

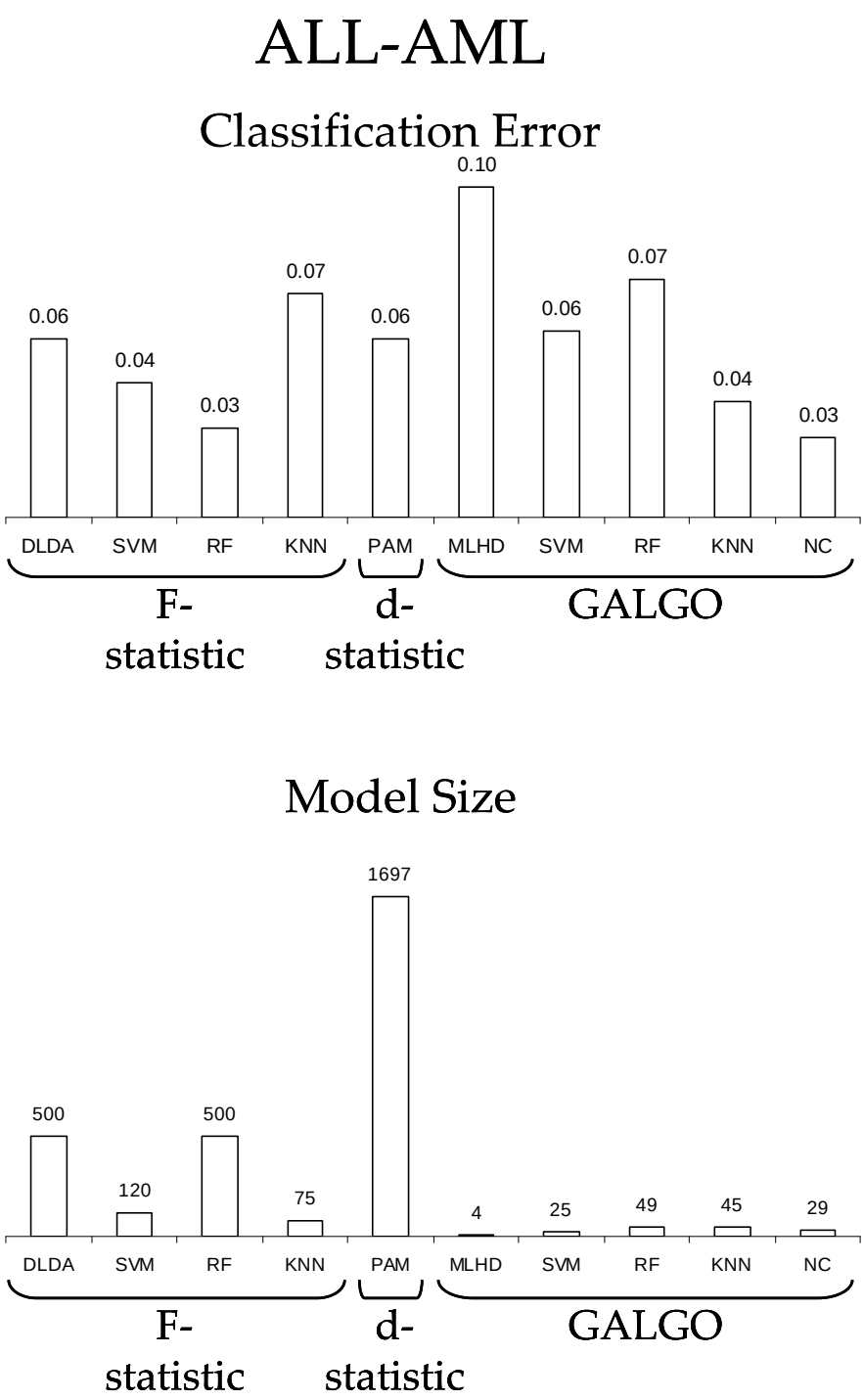

Figure 2 (Appendix)

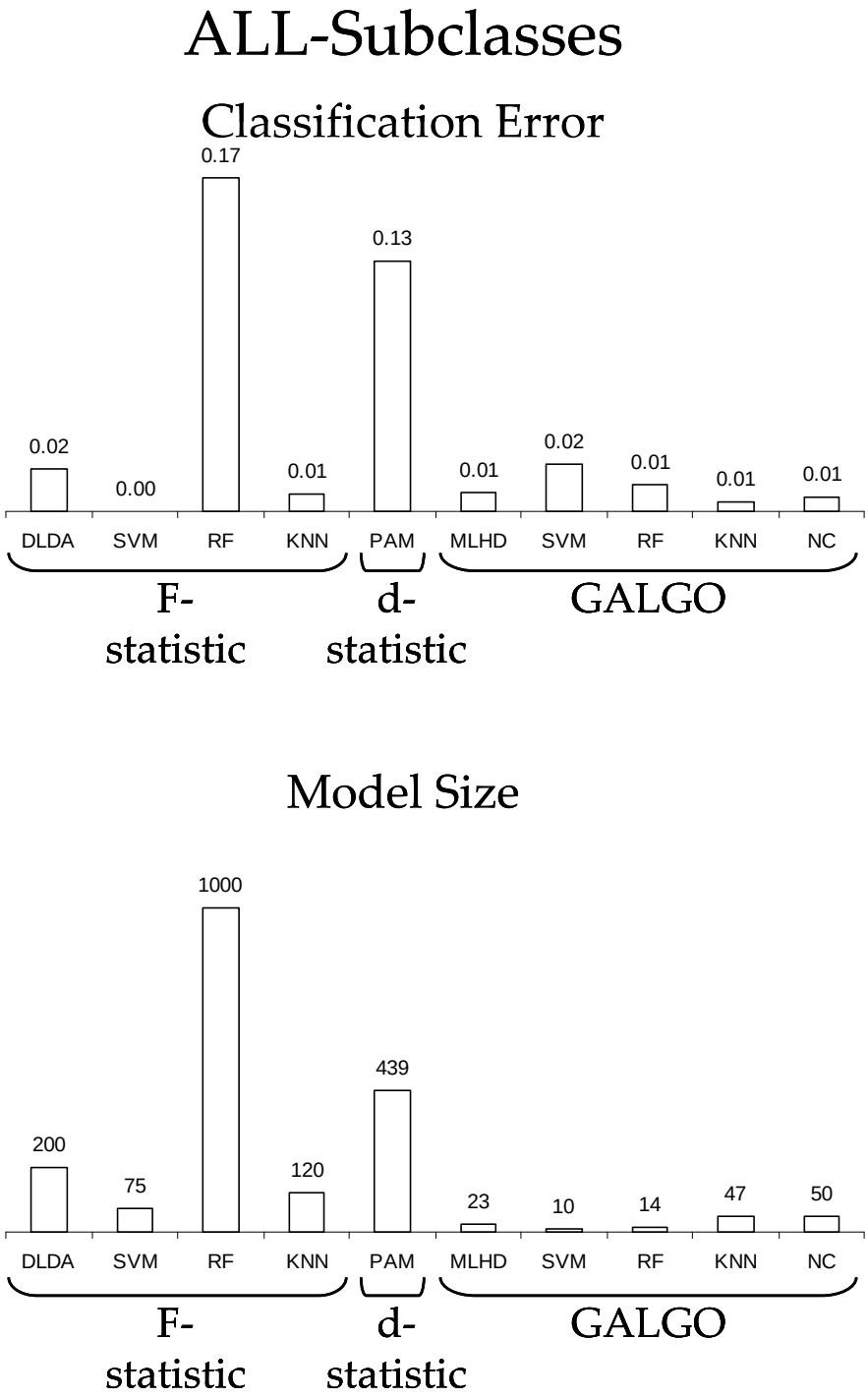

Figure 3 (Appendix)

Method	*Model Size*	*Method-Specific Genes*	*Percent of Method-Specific Genes*
F+DLDA	2	0	0%
F+SVM	5	0	0%
F+RF	10	0	0%
F+KNN	2920	-	-
d+PAM	51	38	75%
GA+BE+MLHD	4	2	50%
GA+SVM	5	3	60%
GA+FS+RF	47	32	68%
GA+BE+FS+KNN	31	11	35%
GA+BE+FS+NC	23	11	48%

Table 7 (Appendix)

REFERENCES

Goldberg, David E. (1989), *Genetic algorithms in search, optimization, and machine learning* (Reading, Mass.: Addison-Wesley Pub. Co. xiii, 412 p.

Li, L. P., et al. (2001), 'Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method', *Bioinformatics*, 17 (12), 1131-42.

Ooi, C. H. and Tan, P. (2003), 'Genetic algorithms applied to multi-class prediction for the analysis of gene expression data', *Bioinformatics*, 19 (1), 37-44.

Yeoh, E. J., et al. (2002), 'Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling', *Cancer Cell*, 1 (2), 133-43.

Vaquerizas J.M., Conde L., Yankilevich P., Cabezon A., Minguez P., Diaz-Uriarte R., Al-Shahrour F., Herrero J & Dopazo J. (2005), Gepas an experiment-oriented pipeline for the analysis of microarray gene expression data. *Nucleic Acids Research* 33 (Web Server issue):W616-W620.

Tibshirani R, Hastie T, Narasimhan B, Chu G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci U S A.* 2002 May 14;99(10):6567-72.

Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*. 1999 Oct 15;286(5439):531-7.

van ‘t Veer LJ, Dai H, van de Vijver MJ, He YD, Hart AAM., Mao M, Peterse HL, van der Kooy K, Marton MJ, Witteveen AT, Schreiber GJ, Kerkhoven RM, Roberts C, Linsley PS, Bernards R, and Friend SH. Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 415, 530 - 536 (2002).

Efron, B., and Tibshirani, R. J. (1993). An introduction to the bootstrap. New York: Chapman & Hall.

Sha, N. J., Vannucci, M., Tadesse, M. G., Brown, P. J., Dragoni, I., Davies, N., Roberts, T. R. C., Contestabile, A., Salmon, M., Buckley, C. and Falciani, F. (2004). "Bayesian variable selection in multinomial probit models to identify molecular signatures of disease stage." *Biometrics* 60(3): 812-819.

Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Ann Arbor, University of Michigan Press.

Ezaki K, Tsuzuki M, Katsuta I, Maruyama F, Kojima H, Okamoto M, Nomura T, Wakita M, Miyazaki H, Sobue R, Matsui T, Ino T, and Hirano M. (1995). Interleukin-1β (IL-1β) and acute leukemia: *In vitro* proliferative response to IL-1β, IL-1β content of leukemic cells and treatment outcome. *Leukemia Research.* 1995 Jan;19(1):35-41.

Hulkkonen J, Vilpo J, Vilpo L, Koski T, Hurme M. (2000) Interleukin-1 beta, interleukin-1 receptor antagonist and interleukin-6 plasma levels and cytokine gene polymorphisms in chronic lymphocytic leukemia: correlation with prognostic parameters. *Haematologica.* 2000 Jun;85(6):600-

Ingenuity® Systems, Mountain View, California 94043, USA. http://www.ingenuity.com

[i] Original E2A-PBX class was renamed to EMLLA.