maigesPack: A Computational Environment for Microarray Data Analysis

GUSTAVO H. ESTEVES,* ROBERTO HIRATA JR†

September 22, 2014

Abstract

Microarray technology is still an important way to assess gene expression in molecular biology. This mainly because one can measure the expression profiles for thousands of genes at the same time and that makes this technology a good option for some studies focused on systems biology. One of the main problems is that the experimental procedure is complex and the data has several sources of variance, that makes the statistical modeling more difficult. So far, there is no standard protocol for the generation and evaluation of microarray data. To mitigate the analysis process this paper presents an R package, named maigesPack. The software helps with the data organization and also with the analysis process. Besides that, it makes the analysis more robust, reliable and reproducible. The package aggregates several data analysis procedures reported in the literature, for instance: cluster analysis, differential expression, supervised classifiers, relevance networks and functional classification of gene groups or gene networks.

Keywords: microarray, R software, statistical methods, gene expression

I. Introduction

Microarray technology is still an important way to assess gene expression in molecular biology. A quick Google Scholar search (26/03/2014) shows about 36,300 results mentioning this technology during 2013. The reason for this number of papers is mainly because one can measure the expression profiles for thousands of genes at the same time and that turns the technology a good option for some studies focused on systems biology.

One of the main problems with this technology is that the experimental procedure is complex and the data has several sources of variance (?), that makes the statistical modeling more difficult. Besides that, the amount of numerical data and non-numerical data to deal with in a rich experiment is challenging. Beginning with the raw values given by the image analysis software, the analysis process unfolds in several steps, each one generating new and large data that are difficult to organize, keep track and later, reproduce.

These characteristics turn the data analysis a very complex and dynamic process, where the most adequate data analysis procedures must be identified (or proposed) and implemented. In this sense, a computational environment that facilitates the integration of the several data analysis methods already proposed would be of great help. Furthermore, this environment should be implemented in a way that facilitates the introduction of new algorithms, as they are proposed so frequently in the literature, and also that facilitates the documentation of all data methods used during the data analysis process.

To mitigate the analysis process, this paper presents an R package implemented in the R system, http://www.r-project.org [Ihaka and Gentleman 1996]. The package is entitled maigesPack, after the acronym MAIGES that stands for Mathematical Analysis of Interacting

*University of Paraíba State, email: gesteves@uepb.edu.br
†University of São Paulo, email: hirata@ime.usp.br
Gene Expression Systems. The main implementation integrates several data analysis packages already available in both CRAN repository and BioConductor project repository, http://www.bioconductor.org (Gentleman et al., 2004). Some computational intensive methods were implemented in C language and can be invoked as R functions. The input data and also the results of some analysis are organized to facilitate the analysis process. In this way, the analysis is more robust, reliable and reproducible.

The remainder of this paper is organized according to the following: Section II presents a Review of the available software to analyze microarray data. Section III presents a brief introduction about maigesPack organization. Section IV shows the implementation structure used into the package. Section V gives some data analysis examples with the corresponding R code. Finally, in Section VI main conclusion and some of the future plans for the package are presented.

II. REVIEW OF EXISTING SOFTWARE

A great problem in microarray data analysis is that different computational tools use specific data structures and/or implement different mathematical and statistical methods, what difficult the interrelation between these different data analysis methods. Besides R briefly mentioned above, there are some programs for this kind of analysis implemented into another computational languages, like Java, as MeV (Saeed et al., 2003) and Cytoscape (Shannon et al., 2003).

Inside R software, there are several packages focused on specific steps of microarray data analysis, like limma (Smyth, 2005) for the adjustment of linear models and the marray (Yang, 2009) or OLIN (Futschik and Crompton, 2004; Futschik, 2012) for exploratory analysis and normalization. Thus, we developed an environment that integrates many of these packages already done along with another data analysis methods aimed at the global microarray data analysis process. The implementation was done aiming to ensure reproducibility and process documentation capability.

Maybe the most famous project associated to microarray data analysis is the Bioconductor http://www.bioconductor.org, (Gentleman et al., 2004), already cited above. Recently, Shen et al. (2012), proposed a web application, known as ArrayOU, that integrates several Bioconductor packages and BioPerl modules specifically designed to process Agilent microarray data. However, it is only limited to differential gene expression analysis.

In the same line of Bioconductor project, Dai et al. (2012) presents a graphical interface, implemented in R, to integrate some packages from the project. However, instead the great contribution of the GUI interface, they also works only with differential gene expression, specially focused on Affymetrix and Illumina platforms.

Morris et al. (2008) proposed a suite of Perl modules, known as PerlMAT, to deal with microarray data, since image manipulation to some data analysis methods.

Infantino et al. (2008) present an approach to construct a simulated microarray image, mainly focusing into evaluation of image segmentation software. In the same line, Belean et al. (2011) shows a work to compare and implement microarray image processing techniques in an automated and less computational-time through parallel computation.

Given the amount of public microarray datasets, nowadays it also possible to compare various studies, taking care off eventual problems to merge datasets from different platforms, in a kind of meta-analysis of microarray data. In this way, Heider and Ali (2013) present a new R/Bioconductor package to do this kind of work.

Some other authors have worked in new insights to use microarray data into a non-traditional analysis methods, like the microarray microdissection with analysis of differences (Liebner et al.

1The acronym has been chosen also because maiges is also the name given to ordinary illiterated people who set themselves up as physicians.
Some efforts have also been done to compare and integrate traditional mRNA microarrays with new technologies recently arisen, like RNA-seq (Chavan et al., 2013; Hänzelmann et al., 2013), microarray expression profiling specific for microRNAs (Brock et al., 2013), or even for arrays specifically created to evaluate changes in DNA methylation, protein phosphorylation states, etc (Wrzodek et al., 2013). Some of these new experimental approaches are based on Next Generation Sequencing (NGS) and its use is growing inside academic community (Shendure and Ji, 2008).

As can be seen here, there are several programs and specially R packages to deal with microarray data. But, some of them just implement specific methods (like gene expression analysis) or, generally, they did not integrate with each other. So, a computational environment capable to do this in an efficient and secure way is important. This was the main focus of this work, and the package maigesPack showed here represents a first step in this direction.

III. maigesPack organization

R is a very rich environment for scientific data organization, analysis and reporting. The structure of the packaging system makes it easy for a scientist to organize data into classes and their methods to operate on them. A packaging system is also available to mitigate the creation of new packages. It creates the directory structure for a new package and also some default files.

The package presented in this section, maigesPack, is an example of a package created using this system. Figure 1 presents the main object classes and methods that were implemented in this package. The figure shows an oriented and acyclic graph where the gray rectangles represent object classes, gray octagons represent graphical or textual outputs (like figures, colormaps, HTML or text files, and so on) and the edges represent computational methods that act over the initial objects giving rise to new objects that they point to. The rectangular blocks in dashed lines (with different colors) present different modules grouping several specific types of data analysis methods, like differential gene expression search, classification methods, gene set expression analysis and relevance networks.

Practically, the graph presents the statistical and mathematical methods that are implemented so far and that can be used to analyze microarray data. The class maiges is the main class of the environment, that is, all data analysis methods act over objects from this class. Each edge represents one or more computational methods used for a specific data analysis process. It is important to notice that the use of these computational methods involves the specification of several parameters, what may not be a trivial task. This design of computational classes and methods confer modularity to the data analysis process, making the parameters specification a simpler task, since each procedure is represented by a well defined sequence of methods in a single way. Furthermore, this feature provides an easy way to extending the computational environment, since new methods can be implemented in new analysis modules.

The graph helps to mitigate the design of the entire data analysis process that can be seen as a three-step procedure. The first one is to draw a graph of an analysis procedure, where the methods to be used must be identified accordingly to the statistical modeling. In the second step, the computational methods already implemented should be identified for each analysis. In this step, new methods that are not yet implemented in the environment are also identified. Finally, in the third step, the parameters associated with the computational methods to be used should be adjusted. This last step defines an instantiation graph (see Figure 2), that is, a graph for the statistical analysis procedure. The computational methods and their parameters are recorded by this instantiation graph. The interpretation of this figure is similar to Figure 1 but, in this case, edges represent methods along with the used parameters. The object stats1 into
Figure 1: Classes and methods implemented in the package. Gray rectangles represent object classes, gray octagons represent graphical or textual outputs. Edges represent mappings between objects of different types. Rectangles in dashed lines represent different data analysis modules.
Figure 2, for instance, represents the result of an analysis of differentially expressed genes by the Wilcoxon test. A key difference between the environment and instantiation graphs is the fact that in the latter each edge of the graph represents a single analysis method that was applied to the source object, generating the target object. Finally, the graph can be later transcribed to a script that runs automatically in the environment.

Figure 2 presents two data analyzes aiming to find differentially expressed (DE) genes. One of them using a Wilcoxon test (DEgenes2by2.wilcox method) and the other one using a resampling test (DEgenes2by2.bootT method). In addition, we also used a method to do hierarchical clustering from the objects generated by the DE analysis. Notice that the entire analysis process starts from an object of class `maigesRaw` and so, following a path in the graph you can follow the methods employed until completion of an specific data analysis process, such as the construction of hierarchical clustering and the generation of HTML tables.

These characteristics of our computational environment give reproducibility to the data analysis process because, once the instantiation graph was defined and executed, it can be stored for an entire data analysis documentation. Once the graph is stored, all data analysis can be re-executed later, without any difficult. Furthermore, if any parameter needs to be altered in any method, or if some data information needs to be modified (like, slide or patient annotation), a new instantiation graph will be created in a simple and efficient way, and the entire analysis can be easily re-executed, which confers robustness to the data analysis process.

IV. maigesPack Implementation

As mentioned above, several computational programs for microarray data analysis have been proposed. Most of them are implemented into R statistical program – [http://www.r-project.org](http://www.r-project.org) (Ihaka and Gentleman, 1996; R Development Core Team, 2011) – that is specially interesting for this type of analysis, since it has several statistical methods and graphical tools implemented,
and naturally it was used to implement maigesPack.

In this environment, we use a procedure to load the gene expression data that ensures data integrity regardless the format of numeric tables generated by the image processing program used. Additionally, several different packages from CRAN and BioConductor specifically designed for processing, normalization and basic data analyzes have been included. Tables 1 and 2 describe the packages from CRAN and from BioConductor that were used, respectively.

**Table 1:** CRAN packages used into the computational environment presented here.

| CRAN Package | Description                              |
|--------------|------------------------------------------|
| amap         | distance functions and $k$-means clustering |
| class        | $k$-neighborhood classification          |
| e1071        | SVM classification                        |
| gplots       | graphical tools for colormaps             |
| MASS         | Fisher’s linear discriminant              |
| R2HTML       | write HTML pages from R objects          |
| rgl          | tri-dimensional graphical tools           |
| som          | SOM clustering                            |

I. Object classes

In this computational environment we defined some object classes to manage since the entire dataset, both raw and normalized, up to specific results from data analysis methods. Initially, we defined an object class that receives all numerical data and cases information. This class is called maigesPreRaw and is formed by lists that stores numeric raw values obtained from data tables generated by the image analysis software, text vectors specifying groups of genes to be evaluated in the work, graphs representing gene regulatory networks of interest, numerical values specifying the chip configuration and labels for the genes and observations. It was also added into this class a logical vector to specify bad spots and another string slot that can be used to store any additional or relevant information about the study.

**Table 2:** Bioconductor packages used into the computational environment presented here.

| BioC. Packages | Description                                      |
|---------------|--------------------------------------------------|
| annotate      | write HTML pages with genomic annotation         |
| convert       | convert between different object classes         |
| graph         | graph manipulation                               |
| limma         | normalization and DE genes analysis by ANOVA     |
| marray        | normalization and exploratory data analysis      |
| multtest      | p-value multiple tests adjustment                |
| OLIN          | normalization by OLIN and OSLIN methods          |

This object class acts as an entrance hall at R which defines an intermediate step, previous to the data analysis. In this intermediate phase it is possible to do exploratory analyzes seeking to identify eventual problems in the dataset, this can result in an update in the bad spots slot. An example of a problem that can be detected in this step is the “fingerprint” into a chip, this one came from a public dataset (Shyamsundar et al., 2005), as illustrated at Figure 3. A maigesPreRaw object is created using the function `loadData`, there are also specific functions to load information about gene groups and networks.

From maigesPreRaw class presented before, we can create objects of class maigesRaw using a method to be presented latter. The main idea here is to store the raw data after the exploratory analysis, where the main problems were already identified and the data are ready for the
normalization step. This class defines four numerical matrices to store the foreground and background intensities both for channel one and two, besides two logical matrices, one of them to index the spots to be used into the normalization step and another to index spots into gene groups. Finally, three lists store gene networks and labels for samples and genes.

Another object class implemented here is named `maiges` that is generated from `maigesRaw` class using specific data normalization methods, discussed in more details latter. The matrices that stores the intensity values at `maigesRaw` class are replaced by other two matrices here containing the log-ratio intensity values (entitled here as \( W \)) and the mean intensity values \( (A) \), also in logarithmic scale. Besides these two matrices, this class also defines another three ones that can store standard deviation and confidence intervals for the normalized values. These values are calculated through an iterative process that repeats the normalization method selecting random selected spot groups. The remaining fields of the `maigesRaw` are kept into `maiges` class. Special attention is given to the class `maigesANOVA`, which is an extension of the class `maiges` containing two additional slots, one of them defines a matrix for the coefficients/contrasts used to fit an analysis of variance model (ANOVA) and the other one stores results from parameters estimation.

All object classes defined in this environment were implemented in a way that some slots aren’t mandatory. For instance, if the user will not do any analysis that needs gene network information, the slot associated with this may stay empty into the object. In case the user try to do an analysis that require this slot, the method will prompt a warning or error message, while all other methods that do not need this information remains working without any problem. As far as we know, this functionality isn’t present in others microarray data analysis R packages. Another object classes were also implemented to store specific data analysis results, as described below:

- `maigesDE` - class to store results from differential gene expression analysis;
- `maigesDEcluster` - an extension to the previous class, which adds a matrix of log-ratio \( W \) for a posterior cluster analysis;
- `maigesClass` - class to store results of discriminant (or classification) analysis;
• maigesActMod - class to store results from functional classification of gene sets;
• maigesRelNetB - class to store results from relevance network analysis [Butte et al., 2000];
• maigesRelNetM - class to store the results from an adapted method for relevance network analysis;
• maigesActNet - class to store results from functional classification of gene networks.

It is important to mention that these classes makes an efficient and robust structure for the computational environment created to microarray data analysis. Also we defined some methods for the conversion between the main classes maigesRaw and maiges in other classes defined at marray (Yang, 2009) and limma (Smyth, 2005) packages. This turned possible to use all available methods from both packages without major problems in our environment. We also created the reciprocal methods, that is, methods that convert objects from marray and limma packages to classes defined here. Below, we present briefly, the data analysis methods that were done in this work. All the methods acts over objects from classes presented here giving as output other objects, graphics and text or HTML files. Some examples will be given at Section V.

II. Data load procedure

The procedure proposed here aims to guarantee data integrity. It is important to note that this task needs some care outside R environment, where all data tables must be checked for presence of any kind of problem, related with differences in line numbers, character coding, data fields with less (or more) elements than others, and so on. Our experience with microarray data analysis had showed several problems like these.

Once these evaluation were done, one may use the loadData method to load all data information. This method generates an object of class maigesPreRaw, and receives just one parameter specifying a configuration file. That specifies all necessary information for the data load process.

Figure 4 shows an example of a configuration file from a real dataset worked by our group. As a result from loadData method we have an object of maigesPreRaw class. After the generation of this object, it is possible to use the methods addGeneGrps and/or addPaths to load information from gene groups (like GO categories) and gene regulation networks (like KEGG pathways), respectively. Currently we are working to improve these functions to use methods from GO.db, KEGG.db and KEGGgraph, as well as looking to use other annotation packages like org.Hs.eg.db.

```r
dataDir = "data"
ext = NULL
sampleFile = "sample_file_new_260107.txt"
datasetId = "Test Dataset"
geneMap = "map48k_07_12_04.txt"
headers = c('Ch1 Mean', 'Ch1 B Mean', 'Ch2 Mean', 'Ch2 B Mean', 'Flags')
skip = 62
sep = "," 
gridR = 12
dridC = 4
printTipR = 10
printTipC = 10
```

Figure 4: An instance of a configuration file used as a parameter for the `loadData` method. Each line represents an information according with the items above.

Once generated the initial maigesPreRaw class object, it must be converted into a maigesRaw class object using the function createMaigesRaw. This method receives an instance of maigesPreRaw and parameters specifying the numerical fields that stores background and foreground
intensity values for both channels that will be used for normalization and data analysis. Furthermore, this method may also use two string parameters giving the gene labels that will be used to map gene groups and gene networks to the respective gene labels, case gene sets and networks will be used in the study.

III. Methods for exploratory data analysis

Previous to any kind of normalization or data analysis, a careful exploratory analysis is extremely important. This work must be done aiming to look for potential problems or pitfalls into the data, that may result in systematic effects. So this initial work is very important to guide the correct identification of data normalization methods. As mentioned above, we did not define any specific method for exploratory data analysis into an instance of class maigesPreRaw. However, all standard R graphical tools, like scatter plots, boxplots, etc, may be executed directly.

For instances of class maigesRaw, it may be interesting to do some descriptive graphics, like MA plots, boxplots and chip image representations. These can be done using some graphical tools implemented inside both marray and limma packages. In our work, we also created methods for the conversion of objects from classes maigesRaw and maiges to classes marrayRaw and marrayNorm, respectively (for package marray), and also to convert same classes to RGList and MAList (for package limma). This turned possible to use, inside our environment, all methods already implemented into both packages. For the descriptive graphical tools, the methods from marray package are used as standard, once they are more generic and present more graphical representation options. It is also possible to convert objects between the classes defined by limma and marray packages, using the package convert (Smyth et al., 2005).

It is important to mention that in this work the common M value, that is the log-ratio from cy5 (R) over cy3 (G) intensity values given by $M = \log_2(R) - \log_2(G)$, was redefined for the log-ratio between the interest sample, denoted by I, over reference sample, denoted by C, intensities (independent of who was marked with one or another dye), this ratio was renamed here for W, ie,

$$W = \log_2(I) - \log_2(C).$$

This was done to prevent common misleading associated with experiments using dye-swap as experimental design.

IV. Data normalization methods

For normalization procedures, there are several available methods, such as lowess non-linear regression (Cleveland, 1979) (global or by blocks), OLIN-OSLIN (Futschik and Crompton, 2004), scale adjustments (global or by blocks) and VSN (variance stabilization normalization) (Huber et al., 2002). All these normalization methods were already implemented in some bioconductor packages, specifically limma, marray and OLIN (Futschik, 2012).

So, we developed the normLoc method based in some functions from limma package for location bias correction. We also developed two methods for scale adjustment (normScaleLimma and normScaleMarray) that use functions implemented into limma and marray packages, respectively. Were implemented another method too, called normOLIN to do normalization based on Optimized Local Intensity-dependent Normalization (OLIN) proposed by Futschik and Crompton (2004) and already implemented into a package with same name and available into BioConductor project. More details about these data normalization methods can be found into the respective references or in Yang and Thorne (2003).

Besides these normalization methods we also constructed an algorithm to repeat the lowess adjustment several times using a pre-defined proportion of dataset points, that are randomly selected. This process turns possible to estimate standard deviation and a confidence interval for the spots after the normalization step. It is important to note too, that several methods for
background subtraction already implemented into limma package may be directly used in the calls for our functions.

V. Methods for differential gene expression analysis, clustering and classification

Maybe one of the most trivial microarray data analysis, but having great importance for biologists in terms of interpretation, is the search for differentially expressed (DE) genes, that consists at identification of individual genes showing distinct mean expression between two or more tissue types between the observed sample. More information about DE analysis can found in Dudoit et al. (2002b) or Dudoit and Yang (2002).

To do differential gene expression analysis, we used two statistical tests already implemented into R (R Development Core Team, 2011) by default (into \texttt{stats} package), that is the Student’s t test for comparison of two sample means and its non parametric equivalent Wilcoxon test (that is similar to Mann-Whitney test). They were used into \texttt{deGenes2by2Ttest} and \texttt{deGenes2by2Wilcox}, to look for DE spots (thought to be representative for some genes) in two different biological conditions. Note that \texttt{stats} package belongs to main \texttt{R} installation, and so, it is not cited at Table 1. Additionally to these classical tests, we also implemented another one based on resampling (or bootstrap) strategy, what gives a more robust option of non parametric test for DE analysis, the method was named \texttt{deGenes2by2BootT}. All methods mentioned here apply into \texttt{maiges} class objects.

For datasets with more complex experimental designs, like that with more than two factors or factorial designs, we used ANOVA models implemented into limma package in one method called \texttt{deGenesANOVA}. This method acts over \texttt{maigesANOVA} class objects that is obtained from \texttt{maiges} class objects using the \texttt{designANOVA} function, that constructs the design and contrasts matrices for the ANOVA model (Smyth, 2005).

We also did a method, \texttt{tablesDE}, to save an HTML (or CSV) file containing the results of DE analysis, and another three methods, that is \texttt{hierMde}, \texttt{kmeansMde} and \texttt{somMde}, to do cluster analysis based into the most DE genes selected by p-values of the tests. All these methods use algorithms for multiple tests p-value adjustment available into \texttt{multtest} package (Pollard et al., 2005), also from BioConductor project. Respectively, the methods uses the hierarchical, \texttt{k}-means and \texttt{Self Organizing Maps} algorithms for clustering implemented in some \texttt{R} packages.

Also, we constructed equivalent methods to do the same cluster analysis into the dataset, without differentially gene expression selection, they are \texttt{hierM}, \texttt{kmeansM} and \texttt{somM}, respectively. These three methods may be applied into objects from classes \texttt{maigesRaw} or \texttt{maiges}, and may be applied for one specific gene group or gene network.

Similarly to the cluster analysis, we also implemented some functions to the main classification techniques. In this case, we developed methods for Fisher linear discriminant analysis (Johnson and Wichern, 2002; Mardia et al., 1979), \textit{Support Vector Machines} (Burges, 1998; Mukherjee and Rifkin, 1999; Vapnik, 1982, 1998) and \textit{k}-neighbors (Dudoit et al. 2002a; Hastie et al., 2001; Ripley, 1996). These methods may be used through \texttt{classifyLDA}, \texttt{classifySVM} and \texttt{classifyKNN}. In this type of analysis it is made an exhaustive search for groups of few genes (like pairs, trios or quartets) capable of distinguishing between different biological conditions, based only on their expression values. The implementation of these functions used several methods from class, \texttt{MASS} (Venables and Ripley, 2002) and \texttt{e1071} (Dimitriadou et al., 2006) \texttt{R} packages. One major problem associated with the exhaustive search is the elevated computational cost, specially for huge datasets. To contour these limitation we implemented similar methods (\texttt{classifyLDAsc}, \texttt{classifySVMsc} and \texttt{classifyKNNsc}) using an heuristic algorithm known as search and choose, that first look for few single genes (like DE genes) and than makes the exhaustive search in these few genes (Cristo, 2003).
VI. Methods for relevance networks and functional classification of gene sets

A kind of generalization of DE genes pointed in the previous section is the classification of gene sets as DE that, differently of the previous one, now classify groups of genes instead of individual ones as DE. This is biologically important to study genes associated with biological functions like Gene Ontology (GO) or gene networks. For these groups of genes it is also possible to construct relevance networks that look for pairs of genes with alterations in association of their expression values.

In our environment, we also implemented models for functional classification of gene networks, as proposed by Segal et al. (2004) and for construction of relevance networks, proposed by Butte et al. (2000). This was done into activeMod and relNetworkR functions, respectively, using some packages from base R installation (specially stats). Beyond these methods, we also did some tools for results visualization, including heatmaps to show activation/inactivation profiles of gene groups and graphical representation for the relevance networks.

Furthermore, we adapted Butte’s original method for relevance networks, at relNetworkM function, where we look for pairs of genes that show significant coefficient correlation alterations between two distinct biological conditions.

As can be seen above, relevance networks construction requires the use of some association measure and the main coefficient used for this is Pearson’s correlation, as proposed originally by Butte et al. (2000). But this coefficient can’t detect non-linear associations so, to circumvent this problem, Butte and Kohane (2000) proposed the use of mutual information as association measure to construct relevance networks. This association measure was also implemented into our environment, as the MI method, using an algorithm proposed by Kraskov et al. (2004).

Another frequent problem associated with Pearson’s correlation is the high susceptibility to outliers presence into dataset, what can make the resulting coefficient measure highly biased. In this way, to minimize the problem we created a more robust method, implemented as robustCor, that remove one extreme value using an idea similar to leave-one-out from classification techniques.

Other important contribution of our work was the proposition of a statistical model to do functional classification of gene regulation networks, were we use information from the p-value of a zero correlation test to construct a test statistic with known probability distribution. The manuscript of this method is under production. This statistical model were implemented into activeNet R method.

VII. C implementation of some functions

The R environment is an extremely powerful tool, as it is designed as statistical computation language for data analysis and manipulation. Also, its graphical tools offers a multitude of possibilities of data visualization and presentation. However, as R is an interpreted language, it pays off a high computational cost for large amounts of data (what is very common in gene expression datasets), specially for methods that needs many loops and nested conditional expressions. This happens into our codes mainly for resampling of t test and calculation of robust correlation coefficient, and this makes a pure R implementation of these codes inefficient.

However, it is possible to use pre-compiled code into another computational language (like C, C++ or Fortran) inside R functions. As these compiled code are much faster than interpreted one, we gain a lot of time using this strategy. Much of the main R methods are implemented into another language (mainly C, C++ or Fortran) and the compiled library are loaded automatically at R initialization. So, to optimize our functions, we implemented the most computational costly piece of code for deGenes2by2BootT, robustCor and MI into C. This gave a significant gain in computational time for these functions.
V. SOME EXAMPLES

In this section we demonstrate the functionality of maigesPack using a dataset with some observations of gastro-esophageal data. This dataset was analyzed by our group and resulted in a publication of a research article in Cancer Research \cite{Gomes2005}. Following we will show some data analysis examples using a piece of this dataset, that is included as part of the package, as gastro dataset. The original data have 4800 spots (approximately 4400 unique genes) and 172 chips with dye swap, what resulted in a dataset of 86 samples. The sub-dataset included in this package has 500 spots (representing 486 unique genes) and 40 chips (from 20 observations).

The implementation we did use some variables (known into maigesPack as labels) for samples and genes. For gastro dataset we have three important labels for samples and 5 important labels for genes. To samples we have Sample that shows an unique identification to each observation, Tissue that shows the tissue classification for each observation (Aeso is Adenocarcinoma from esophagus, Aest is adeno from stomach, Neso is normal esophagus and Nest is normal stomach) and Type that shows the general type of the observations (Col is columnar tissue and Sq is squamous tissue). For genes the important labels are Name, Genid, GeneName, ClusterId and Annot.

To load this dataset start an R in any working directory, load maigesPack and the gastro dataset using the commands given below:

\begin{verbatim}
> library(maigesPack)
> data(gastro)
\end{verbatim}

The command, data(gastro) above, loads five objects into R session. Table 3 shows names and classes from these objects.

| Object Name  | Object Class     |
|--------------|------------------|
| gastro       | maigesPreRaw     |
| gastro.raw   | maigesRaw        |
| gastro.raw2  | maigesRaw        |
| gastro.norm  | maiges           |
| gastro.summ  | maiges           |

So, to see all sample labels available into gastro, type names(gastro@Slabels), analogously, type names(gastro@Glabels) to see all gene labels. These commands also works for the other four objects available into gastro dataset.

The object gastro.raw, that is of class maigesRaw, is already available into dataset. The comando used to generate this object was the one given below.

\begin{verbatim}
> gastro.raw = createMaigesRaw(gastro, greenDataField="Ch1.Mean",
+    greenBackDataField="Ch1.B.Mean", redDataField="Ch2.Mean",
+    redBackDataField="Ch2.B.Mean", flagDataField="Flags",
+    gLabelGrp="GeneName", gLabelPath="GeneName")
\end{verbatim}

I. EXPLORATORY ANALYSIS

From the object gastro.raw it is possible to do several exploratory analysis. The generic function plot can be used to show WA plots. The command below do this plot for the first chip and the result is represented at Figure 5.

\begin{verbatim}
> plot(gastro.raw[,1], bkgSub="none")
\end{verbatim}
Figure 5: WA plot for the first chip of the \texttt{gastro.raw} object (a) and for \texttt{gastro.norm} object (b).

In above command, changing the parameter \texttt{bkgSub}, the user change the method of background subtraction, for example, if the user specify \texttt{bkgSub="subtract"} the conventional background subtraction is done. You may also do representations for the numerical values into the chip, using the method \texttt{image}. See command below, that represents the \(W\) values (the \(W\) values are represented by default, that is, without any additional parameter, if additional parameters are added, the conventional \(M\) values are used) for the first chip that generated the Figure 5. note that only some points are represented, because we represent only 500 spots randomly selected.

\begin{Sinput}
> image(gastro.raw[,1])
\end{Sinput}

To change the \(W\) for another value, like \(A\) values, use \texttt{image(gastro.raw[,1], "maA")}. Take a look at \texttt{maImage} method from package \texttt{marray} to see additional parameters for this function. Another type of exploratory graphics is the boxplot, that may be made using the following command, figure not shown.

\begin{Sinput}
> boxplot(gastro.raw[,2])
> boxplot(gastro.raw)
\end{Sinput}

The user may also do an hierarchical cluster to visualize the quality of the data. Specially when replicates of the observations are done (like the dye swap). Use the command below to construct the hierarchical dendrogram represented at Figure 7(a).

\begin{Sinput}
> hierm(gastro.raw, rmGenes=c("BLANK","DAP","LYS","PHE","Q\_GENE","THR","TRP"),sLabelID="Sample", gLabelID="Name", doHeat=FALSE)
\end{Sinput}

II. Normalizing the dataset

The first step to normalize the data is to select the spots to be used for estimating the normalization factor. In this package this is done by the function \texttt{selSpots}. In our example use the command below.
Figure 6: Image for the first chip of the gastro data.

> gastro.raw2 = selSpots(gastro.raw, sigNoise=1, rmFlag=NULL, 
gLabelsID="Name", remove=list(c("BLANK","DAP","LYS","PHE","Q_GENE","THR","TRP")))

This function automatically sets the spots for each chip (column) independently that will be used according with the criteria specified. For example, if the user want to remove spots with signal to noise ratio less than 1.5, specify this value as sigNoise=1.5, to remove spots marked with flags 1 and 4, specify rmFlag=c(1,4), and so on.

Once this is done, the main function to do the normalization is normLoc that uses another function from limma package. The most usual normalization method may be applied by the function.

> gastro.norm = normLoc(gastro.raw2, span=0.4, method="loess")

There are another possibilities to do the normalization, like the method OLIN implemented in a package with the same name, and the method that repeats the lowess fitting calculating the standard variation and confidence interval. These may be done by using the commands below, but pay attention with them because they are very time consuming, so it is interesting to look their arguments.

> gastro.norm = normOLIN(gastro.raw2)
> gastro.norm = normRepLoess(gastro.raw2)

After the locale normalization the user can use the functions normScaleMarray and normScaleLimma to do scale adjustment. The first method use a function from marray package and the second use a function from limma package. To do scale adjustment between print tips use the command below.

> gastro.norm = normScaleMarray(gastro.norm, norm="printTipMAD")

To adjust the scale between chips estimating the adjustment factor by MAD use the following command.
Figure 7: Hierarchical cluster for all observations from raw (a) an normalized (b) gastro dataset.

> gastro.norm = normScaleMarray(gastro.norm, norm="globalMAD")

In the other hand, limma package offers another possibilities. These were also incorporated into maigesPack package. To do the scale adjustment for chips using an estimator slightly different from the MAD, use:

> gastro.norm = normScaleLimma(gastro.norm, method="scale")

Another possibility is to do the stabilization of the variance along A values for all chips. But this method must be applied directly over the raw object. This may be done using the command below. Again, pay attention with this method, because it is also very time consuming.

> gastro.norm = normScaleLimma(gastro.raw2, method="vsn")

All normalization functions generate objects of class maiges. The exploratory functions exemplified for raw objects may also be used with this class, in same way they are applied into that objects Figure 5(b) presents the normalized version of WA plot for the first chip. Also the hierarchical clustering must be done after the normalization step. So into this dataset (with dye swaps), we can observe the replicate pairing for most patients in dye swap. This can be seen at Figure 7(b), compare the swap pairings into (a) and (b) figures.

Sometimes, the sequence (or genes) fixed onto spots may be replicated. The same thing may happen with the observations (the most common is dye swap). So depending on the statistical models used, the user must have to resume the data both for spots and biological observations. These may be done by using the function summarizeReplicates, as exemplified below. If the user needs to resume only spots or samples, simply specify funcS=NULL or funcG=NULL. If both of them are NULL no summary are done.

> gastro.summ = summarizeReplicates(gastro.norm, gLabelID="GeneName", + sLabelID="Sample", funcS="mean", funcG="median", + keepEmpty=FALSE, rmBad=FALSE)

III. Some data analysis methods

After pre-processing and subsequent data normalization steps, the user may use several statistical methods for data analysis. As mentioned above, maigesPack package integrated
some methods that were already available into R and/or BioConductor project. Between these methods we have cluster algorithms, differential gene expression and discrimination techniques, functional classification of gene groups or gene networks and construction of relevance networks.

III.1 Differentially expressed genes (DE genes)

As mentioned into Section IV we implemented three main methods to search by DE genes: deGenes2by2Ttest, deGenes2by2Wilcoxon and deGenes2by2BootT. The first one supposes that the data are normally distributed and uses the t statistic to do the calculations. The other two are non-parametric and do not assumes normality into data. The use of these functions is illustrated by the following command, the other two are used in the same way, and will not be presented here.

```r
> gastro.ttest = deGenes2by2Ttest(gastro.summ, sLabelID="Type")
> gastro.ttest

```

Object of class maigesDEcluster generated by Fri Sep 28 21:00:20 2012, using R version 2.15.1 (2012-06-22).

Classifying 487 genes as Differentially Expressed (DE)

The method used was T test.

Once the analysis was done. The user can see volcano plots and save HTML (or CSV) tables with the results using the commands below, respectively.

```r
> plot(gastro.ttest)
> tablesDE(gastro.ttest)

```

It is also possible to do cluster analysis selecting DE genes according with the p-values of the tests. The functions hierMde, somMde and kmeansMde do cluster analysis using hierarchical, SOM and k-means algorithms, respectively. To do the SOM cluster with 2 groups using 20 most differentially expressed genes (adjusting p-values by FDR) use the command below, the result is into Figure 8. Similarly, it is possible to do cluster analysis directly for objects of class maiges, using the functions hierM, somM and kmeansM, but without selecting differentially expressed genes.

```r
> somMde(gastro.ttest, sLabelID="Type", adjP="BH", nDEgenes=20, + xdim=2, ydim=1, topol="rect")

```

All the methods presented above are applicable to compare only two distinct biological sample types. When there are more than two types, it is possible to use ANOVA models. We also implemented a method to do this kind of analysis, using functions from limma. But first the user must run another method to construct design and contrasts matrices. To construct these matrices for an ANOVA model for Tissue factor (note that this factor has 4 sample types, type getLabels(gastro.summ, "Type") to see them), use the command below.

```r
> gastro.ANOVA = designANOVA(gastro.summ, factors="Tissue")

```

And them, to fit the model and to do the F test use the following command.

```r
> gastro.ANOVAfit = deGenesANOVA(gastro.ANOVA, retF=TRUE)
> gastro.ANOVAfit

```
Figure 8: SOM cluster with 2 groups using 20 most DE genes.

Object of class maigesDEcluster generated by Fri Sep 28 21:00:21 2012, using R version 2.15.1 (2012-06-22).

Classifying 487 genes as Differentially Expressed (DE)
   i 1 test(s).

The method used was ANOVA - F test.

If the user want to do the individual t tests use (this is the default) below.

> gastro.ANOVAfit = deGenesANOVA(gastro.ANOVA, retF=FALSE)

Another function that may be useful, specially for ANOVA analysis is boxplot. This type of plot may help in identify the tissues where some gene presented alteration. For example the gene KLK13, presents significantly alteration in the F test done above. Using the command below it is possible to see in what tissue this gene is altered. The result is illustrated in the Figure. The method boxplot also works with objects of class maiges.

> boxplot(gastro.ANOVAfit, name="KLK13", gLabelID="GeneName",
          + sLabelID="Tissue")

The object resulted from this fitting are also of class maigesDE (or maigesDEcluster). So, the commands for plot, to save tables and to construct clusters, presented above also works here. Pay attention that functions to do cluster analysis only work for objects of classes maigesDEcluster.
III.2 Discrimination analysis

Discrimination (or classification) analysis, that search by groups of few genes that can distinguish between different sample types was implemented in our package into classifyLDA, classifySVM and classifyKNN functions, as mentioned before. To evaluate the quality of the groups, we use the cross validation method.

It is possible to search exhaustively into all genes from the dataset, but this is very time consuming. So, we added the possibility to search for classifiers in gene groups or networks. To search by trios of genes that classify the two types (Col and Sq) using the Fisher’s method, for the sixth (cell cycle arrest, gene ontology code GO0007050) gene group (into GeneGrps slot from gastro.summ object) we use the following command.

```r
> gastro.class = classifyLDA(gastro.summ, sLabelID="Type", +   gNameID="GeneName", nGenes=3, geneGrp=6)
> gastro.class
```

Object of class maigesClass generated by Fri Sep 28 21:00:22 2012, using R version 2.15.1 (2012-06-22).

56 classifiers of 3 genes, ordered by CV value.

The method used was Fisher LDA by exhaustive search.

To visualize or save tables (HTML or CSV) for the classifiers it is possible to use the two commands below. But pay attention that plot only works for pairs or trios classifiers. For trios, it will do a 3-dimensional plot using rgl package.

```r
> plot(gastro.class, idx=1)
> tableClass(gastro.class)
```
As mentioned into before, exhaustive search is very time consuming for large gene sets and so we implemented the search and choose empiric method (classifyLDAsc, classifySVMsc and classifyKNNsc), that can be used in the same way as the exhaustive ones.

III.3 Functional classification of gene groups

Functional classification of gene groups (or modules), proposed by [Segal et al. (2004)](Segal2004). This method basically searches for groups of genes that present numbers of differentially expressed genes greater than the expected by chance. The user can do this kind of analysis for all gene groups loaded into the dataset (this information is stored in the slot GeneGrps from objects of classes maigesPreRaw, maigesRaw or maiges) for sample label Tissue using the command below.

```r
> gastro.mod = activeMod(gastro.summ, sLabelID="Tissue", cutExp=1, +     cutPhiper=0.05)
```

To plot the results, as an image (like a heatmap), for the individual observations the user can use the following command.

```r
> plot(gastro.mod, "S", margins=c(15,3))
```

For each tissue the user can use the command below. The result is represented in the Figure 10.

```r
> plot(gastro.mod, "C", margins=c(23,5))
```

![Figure 10: Result from functional classification of gene groups for Tissue label.](image)

The function also calculate an score (with significance levels, p-values) given the concordance of the gene and group classification. These values can be saved, as HTML tables, for each sample type using the command activeModScoreHTML.

III.4 Relevance networks

Finally, to conclude this section, relevance networks proposed by [Butte et al. (2000)](Butte2000) and also discussed earlier, estimate linear Pearson’s correlation values between all pairs of genes from a
given gene group and test the significance of this value under the hypothesis of null correlation. Then, we select the pairs with correlations values significantly different from zero using some p-value cutoff. To construct relevance networks for normal esophagus tissue (Neso) in the first gene group, use the command.

```r
> gastro.net = relNetworkB(gastro.summ, sLabelID="Tissue",
+                  samples="Neso", geneGrp=1)
```

To visualize the results it is possible to do an image of the correlation values for all pairs of genes or a graph representing that ones with p-values less than cutPval. This can be done using the below commands.

```r
> image(gastro.net)
> plot(gastro.net, cutPval=0.05)
```

However, the Butte’s original method do not makes possible to compare quantitatively the results obtained in two different biological conditions. So, we implemented another method that construct relevance networks with pairs of genes presenting altered correlation values between two sample types. This is done by a Fisher’s Z transformation. To find pairs of genes that present altered correlation values between esophagus’s normal and adenocarcinoma tissues, we use the following command.

```r
> gastro.net2 = relNetworkM(gastro.summ, sLabelID="Tissue",
+                  samples = list(Neso="Neso", Aeso="Aeso"), geneGrp=7)
```

Also, it is possible to use the commands `image` and `plot` to visualize the results, like the next two commands.

```r
> image(gastro.net2)
> plot(gastro.net2, cutPval=0.05)
```

Another graphical output from this kind of analysis is the comparison between the regression lines into the expression values for an altered gene pair. This can be done as showed by the next command, where we compare the regression for genes KLK13 and EVPL both into Neso and Aeso tissue types.

```r
> plotGenePair(gastro.net2, "KLK13", "EVPL")
```

In next section we present the main concluding remarks about the `maigesPack` as well as the availability of the package and the major next steps of our work into this R package.

### VI. Conclusions

This work was done inside a greater project entitled MAIGES (Mathematical Analysis of Interacting Gene Expression Systems), that is hosted at Statistics Department at São Paulo University.

As we show above, we created a computational environment based on the R software, integrating several mathematical and statistical tools already implemented into another R packages, both from CRAN and from BioConductor, as well as together another ones implemented in this work. We choose the R language to do the implementations because it is a statistical software with several statistical tools, and mainly the several probabilistic models already available. Also it has a great community all over the world developing tools in it, as the BioConductor package itself. This project focuses into developing statistical tools devoted mainly for genomic data analysis, as gene expression analysis, what was the focus of our work. Besides all these benefits, R is also free software, based into GNU public license.
The environment we presented here is intended as a modular computational framework, where new statistical and/or mathematical tools can be easily added. So, as can be observed into Figure 1, the dotted rectangles represents modules for DE genes analysis, clustering, classification, construction of relevance networks, and functional classification of gene sets and networks. So, as new methods are proposed in the literature they can be added to these modular data analysis scheme. Also, new modules for another type of analysis can be easily included into this structure.

Obviously, the computational environment presented here needs several adjustments in the implementations of several functions, since a data structure organization to several optimization in computational timings, that is our main working by this time. But, even so, it works well for a organized and fully reproducible gene expression (mainly based onto microarray technology) data analysis. Another future work that deserves attention is the adaptation of the package to deal with the technologies for large scale gene expression measurements, like RNA-seq or microRNA microarrays.

The maigesPack R package that resulted from this work was submitted an approved into BioConductor project and is available from the project repositories into the link http://www.bioconductor.org/packages/release/bioc/html/maigesPack.html. The actual version of the package is 1.26.0.

ACKNOWLEDGMENTS

This work was financed by CAPES. Also, we would like to thanks Raydonal Ospina Martinez and Diana Maia, for additional support to write the manuscript, and E. Jordão Neves for several help and suggestions in this work.

REFERENCES

B. Belean, M. Borda, B. LeGal, and R. Malutan. FPGA technology and parallel computing towards automatic microarray image processing. 2011 34th Int. Conf. Telecommun. Signal Process., pages 607–610, 2011. doi: 10.1109/TSP.2011.6043657.

G. N. Brock, P. Mukhopadhyay, V. Pihur, C. Webb, R. M. Greene, and M. M. Pisano. MmPalateMiRNA, an R package compendium illustrating analysis of miRNA microarray data. Source Code Biol. Med., 8(1):1, Jan. 2013. ISSN 1751-0473. doi: 10.1186/1751-0473-8-1. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3654997&tool=pmcentrez&rendertype=abstract.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov., 2:121–167, 1998.

A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. Pacific Symp. Biocomput., 5:415–426, 2000.

A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane. Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. PNAS, 97(22):12182–12186, 2000.

S. Chavan, M. Bauer, E. Peterson, C. Heuck, and D. Johann. Towards the integration, annotation and association of historical microarray experiments with RNA-seq. BMC Bioinformatics, 14:1–11, 2013. doi: 10.1186/1471-2105-14-s14-s4. URL citeulike-article-id:12729251\delimiter^026E30F$\text{http://dx.doi.org/10.1186/1471-2105-14-s14-s4}

W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. J. Am. Stat. Assoc., 74(368):829–836, 1979.
E. B. Cristo. Métodos Estatísticos na Análise de Experimentos de Microarray. Msc, Instituto de Matemática e Estatística - USP, 2003.

Y. Dai, L. Guo, M. Li, and Y.-B. Chen. Microarray R US: a user-friendly graphical interface to Bioconductor tools that enables accurate microarray data analysis and expedites comprehensive functional analysis of microarray results. *BMC Res. Notes*, 5:282, Jan. 2012. ISSN 1756-0500. doi: 10.1186/1756-0500-5-282. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3459790&tool=pmcentrez&rendertype=abstract.

E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and Andreas Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071)*, TU Wien, 2006.

S. Dudoit and Y. H. Yang. *The analysis of gene expression Data: methods and software*, chapter BioConduct. Springer, New York, 2002.

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *J. Am. Stat. Assoc.*, 97(457):77–87, 2002a.

S. Dudoit, Y. H. Yang, M. J. Callow, and T. P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Stat. Sin.*, 12(1):111–139, 2002b.

M. Futschik. *OLIN: Optimized local intensity-dependent normalisation of two-color microarrays*, 2012. URL http://itb.biologie.hu-berlin.de/~futschik/software/R/OLIN.

M. Futschik and T. Crompton. Model selection and efficiency testing for normalization of cDNA microarray data. *Genome Biol.*, 5(8):R60, 2004.

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, C. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, 5:R80, 2004.

L. I. Gomes, G. H. Esteves, A. F. Carvalho, E. B. Cristo, R. Hirata, W. K. Martins, S. M. Marques, L. P. Camargo, H. Brentani, A. Pelosof, C. Zitron, R. a. Sallum, A. Montagnini, F. a. Soares, E. J. a. Neves, and L. F. L. Reis. Expression profile of malignant and nonmalignant lesions of esophagus and stomach: differential activity of functional modules related to inflammation and lipid metabolism. *Cancer Res.*, 65(16):7127–36, Aug. 2005. ISSN 0008-5472. doi: 10.1158/0008-5472.CAN-05-1035. URL http://www.ncbi.nlm.nih.gov/pubmed/16103062.

S. Hänzelmann, R. Castelo, and J. Guinney. GSVA: gene set variation analysis for microarray and RNA-seq data. *BMC Bioinformatics*, 14:7, Jan. 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-7. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3618321&tool=pmcentrez&rendertype=abstract.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer Verlag, New York, 2001.

A. Heider and R. Alt. virtualArray: a R/bioconductor package to merge raw data from different microarray platforms. *BMC Bioinformatics*, 14:7, Jan. 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-7. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3599117&tool=pmcentrez&rendertype=abstract.

W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression.
Bioinformatics, 18(Supp. 1):S96—S104, 2002.

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. J. Comput. Graph. Stat., 5(3):299–314, 1996.

I. Infantino, C. Lodato, and S. Lopes. Testing and Evaluation of Microarray Image Analysis Software. 2008 Int. Conf. Complex, Intell. Softw. Intensive Syst., 2008. doi: 10.1109/CISIS.2008.19.

R. Johnson and D. Wichern. Applied Multivariate Statistical Analysis. Prentice Hall, New Jersey, 5th edition, 2002.

A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. Phys. Rev. E, 69(6):66138, June 2004. ISSN 1539-3755. doi: 10.1103/PhysRevE.69.066138. URL http://link.aps.org/doi/10.1103/PhysRevE.69.066138

D. A. Liebner, K. Huang, and J. D. Parvin. MMAD: microarray microdissection with analysis of differences is a computational tool for deconvoluting cell type-specific contributions from tissue samples. Bioinformatics, pages 1–8, Oct. 2013. ISSN 1367-4811. doi: 10.1093/bioinformatics/btt566. URL http://bioinformatics.oxfordjournals.org/content/early/2013/10/21/bioinformatics.btt566.abstract.html?papetochttp://www.ncbi.nlm.nih.gov/pubmed/24085566

K. Mardia, J. Kent, and J. Bibby. Multivariate Analysis. Academic Press, New York, 1979.

J. A. Morris, S. A. Gayther, I. J. Jacobs, and C. Jones. A suite of Perl modules for handling microarray data. Bioinformatics, 24(8):1102–3, Apr. 2008. ISSN 1367-4811. doi: 10.1093/bioinformatics/btn085. URL http://www.ncbi.nlm.nih.gov/pubmed/18353790

S. Mukherjee and R. Rifkin. Support Vector Machine Classification of Microarray Data. CBCL Pap., 8(4):59–60, 1999.

K. S. Pollard, Y. Ge, and S. Dudoit. multtest: Resampling-based multiple hypothesis testing, 2005.

R Development Core Team. R: A Language and Environment for Statistical Computing, 2011. URL http://www.r-project.org

B. D. Ripley. Pattern recognition and neural networks. Cambridge University Press, New York, 1996.

A. I. Saeed, V. Sharov, J. White, J. Li, W. Liang, N. Bhagabati, J. Braisted, M. Klapa, T. Currier, M. Thiagarajan, A. Sturn, M. Snuffin, A. Rezantsev, D. Popov, A. Ryltsov, E. Kostukovich, L. Borisovsky, Z. Liu, and J. Quackenbush. TM4: A free, open-source System for microarray data management and analysis. Biotechniques, 34(2):374–378, 2003.

E. Segal, N. Friedman, D. Koller, and A. Regev. A module map showing conditional activity of expression modules in cancer. Nat. Genet., 36(10):1090–1098, 2004.

P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res., 13:2498–2504, 2003.

K. Shen, S. E. Wyatt, and V. Nadella. ArrayOU: a web application for microarray data analysis and visualization. J. Biomol. Tech., 23:37–9, 2012. ISSN 1943-4731. doi: 10.7171/jbt.2012-2302-001. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3378286&tool=pmcentrez&rendertype=abstract

J. Shen-Gunther and J. Rebeles. Genotyping human papillomaviruses: development and
evaluation of a comprehensive DNA microarray. *Gynecol. Oncol.*, 128:433–41, 2013. ISSN 1095-6859. doi: 10.1016/j.ygyno.2012.11.028. URL http://www.ncbi.nlm.nih.gov/pubmed/23200917.

J. Shendure and H. Ji. Next-generation DNA sequencing. *Nat. Biotechnol.*, 26:1135–1145, 2008. ISSN 1087-0156. doi: 10.1038/nbt1486.

R. Shyamsundar, Y. H. Kim, J. P. Higgins, K. Montgomery, M. Jorden, A. Sethuraman, M. van de Rijn, D. Botstein, P. O. Brown, and J. R. Pollack. A DNA microarray survey of gene expression in normal human tissues. *Genome Biol.*, 6:R22, 2005.

G. Smyth, J. Wettenhall, and Y. H. Yang. *Convert: Convert Microarray Data Objects*, 2005. URL http://bioinf.wehi.edu.au/limma/convert.html.

G. K. Smyth. limma: Linear Models for Microarray Data User’s Guide. In *Bioinforma. Comput. Biol. Solut. using R Bioconductor*, pages 397–420. Springer, New York, 2005.

V. Vapnik. *Estimation of dependences based on empirical data*. Springer Verlag, New York, 1982.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4.

C. Wrzodek, J. Eichner, F. Büchel, and A. Zell. InCroMAP: integrated analysis of cross-platform microarray and pathway data. *Bioinformatics*, 29(4):506–8, Feb. 2013. ISSN 1367-4811. doi: 10.1093/bioinformatics/bts709. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3570209&tool=pmcentrez&rendertype=abstract.

Y. H. Yang. *marray: Exploratory analysis for two-color spotted microarray data*, 2009. URL http://www.maths.usyd.edu.au/u/jeany/.

Y. H. Yang and N. P. Thorne. Normalization for Two-color cDNA microarray data. *IMS Lect. Notes, Monogr. Ser.*, 40:403–418, 2003.