A Cloud Computing System to Quickly Implement New Microarray Data Pre-processing Methods

Dajie Luo[1,#], Prithish Banerjee[1,#], E. James Harner[1], James A. Mobley[2] and Dongquan Chen[3,4,*]

[1]*Department of Statistics, West Virginia University, Morgantown, WV 26505, USA*

[2] *Department of Surgery, University of Alabama at Birmingham (UAB), USA*

[3]*Biostatistics and Bioinformatics Shared Facility, Comprehensive Cancer Center and* [4]*Division of Preventive Medicine, UAB. Birmingham, AL 35294, USA*

Abstract: *Background:* Pre-processing, including normalization of raw microarray data is crucial to microarray-related data analysis. It takes time and effort to build newly-developed algorithms into commercial software or locally developed systems. While most new algorithms emerge in the form of sharable R packages, it can be difficult for many biologists to apply them as soon as they are available. Currently, we rely on statisticians and experienced programmers to develop and implement code to access those R packages. Therefore, we need a robust procedure to quickly implement pre-processing methods as they appear. The newly emerging cloud computing concept has directed us toward a new way for providing an easily accessible service to the biologists without requiring them to have any programming knowledge in R.

Results: Based on our earlier Java-based software tool JavaStat, we developed an internet based application prototype to upload data and carry out pre-processing applications that include normalization, statistical analyses and plots. More importantly, R packages, e. g., for newly-developed normalization methods, and GC-robust multichip algorithm (RMA) for exon arrays, can be easily incorporated into the system with limited inputs from a biologist or a programmer. The data are stored in the cloud and the R code runs on server.

Conclusion: The newly emerged cloud computing concept provides us a new way to provide an easily accessible and up-to-date service to biologists, as evidenced by our JavaStat system to incorporate new pre-processing package as they appear. Users can access the application with a newly incorporated module through the Web. We expect this and other similar systems greatly decrease turn-around time, improve accessibility of newly developed R model for pre-processing algorithms.

Keywords: Microarray, normalization, software, Java-based.

INTRODUCTION

JavaStat

JavaStat, a previously reported system written in Java [1], has a highly interactive statistical and modeling environment. The front-end is an interactive graphical user interface (GUI) for data analysis and dynamic visualization with data management capabilities. The back-end server uses R/Bio-conductor as a powerful computing engine to run complex statistical models and carry out various types of microarray analyses. The concept revolves around the use of RMI (Remote Method Invocation) to communicate front-end commands with a back-end Java server program (JRIServer), which in turn communicates with R using JRI (Java/R Interface). A workflow has been developed in JavaStat for this purpose. The computation and data storage are completely done on the server, so it is not necessary to install R on client machines and thus no additional knowledge of R is required for clients.

*Address correspondence to this author at the Division of Preventive Medicine University of Alabama at Birmingham, Birmingham, AL 35294, USA; Tel: (205) 975-7131; Fax: (205) 934-4262; E-mail: dongquan@uab.edu
[#]The authors contribute equally to the work.

Although a user account is required to enable the feature of workflows, (e.g. genomic analysis and modeling), the system could potentially support a large number of users, as the server side program is designed to be scalable. While R is a single thread application, our system enables multiple R instances to run simultaneously on the server to load balance requests from large number of clients. The benefits of cloud computing is to reduce investment in infrastructure such as computing, storage, software, and upkeep. The analysis of genomics datasets in R usually requires high end computing due to the large size of high throughput data sets. Within our architecture, the computing is performed on the server side, so the client computer doesn't need to be very powerful or costly.

Current Microarray Data Analysis Tools and Normalization Methods

We have been using commercial products such as GeneSpring (Agilent, CA), Partek (Partek Inc, MO), Genome Studio (Illumina, Inc, CA), and open source tools such as Bioconductor [2], which are capable of analyzing most types of single and dual color arrays. Affymetrix also pro-

1875-0362/12

vides a simple tool called Expression Console for simple data preprocessing and displays. It is common that newly developed pre-processing methods are not made readily available to investigators simply because they need to be incorporated into either commercial products or locally-developed tools. Although it is usually quicker for new tools to be available in the open-source community, such as in the R Foundation, biologists and statisticians often still require extra help from programmers to implement and use these newly developed packages.

The pre-processing procedure is a crucial step before statistics, and is meant to minimize the inherent non-linearity of microarray analysis stemming from scanners, reagents (e.g. dyes), and sequence contents, while also adjusting for changes due to experiments carried out at different times and/or by different people. At present, there is no consensus on pre-processing methods, which have been emerging more frequently than ever before. Notably, the most commonly used pre-processing methods include MAS5 (Microarray Suite 5, Affymetrix) for single color arrays [3], Dchip for multi-color arrays [4], and others such as the robust multi-array analysis (RMA) [5], quantile [6], GC-RMA [7], and PLIER [3]. More recently PQN and DQN [8] attempted to compare various normalization methods. However, not all these methods are available in any of the software packages mentioned above. The objective of this study is to develop a procedure to streamline the implementation and to make newly-developed pre-processing methods available as they appear.

MATERIAL AND METHODS

Implementation

Clients access the server as shown in the Fig. (**1**). A group of duplicated applications called JRIServers run continuously on the cloud-based server. The JRIServer listens to requests from the client programs and accesses R installed on server. The R objects that were created throughout the active computation period are retained in each user's workspace. Thus the subsequent requests from the same user can access and manipulate the user's data.

Operational Procedure

System access and analysis follow the following procedure: 1) JavaStat sends the request that contains the command, client data, and the client identification to a server through the Internet. 2) The JRIServer puts the request that contains the identification into a queue. Control is instantly returned to the client, which is free to send another remote request. 3) JRIServer has a thread checking the queue continuously. It reads the next request out of the queue if the queue is not empty. 4) The thread generates R code and runs it on the server. 5) R reads (or writes) the user-owned workspace file as required. 6) The thread converts the R results into a Java object when the analysis is completed and puts it into a repository with finished tasks. 7) The client program JavaStat polls the server periodically. 8) The server reads results from the repository when polled and sends them back to the client according to cached identifications.

This architecture supports multiple JRIServers and R instances running on the same server to improve the efficiency. If multiple users are on the client side, each of them may access a different JRIServer and their requests are run in parallel.

RESULTS AND DISCUSSION

Functionality of the Modified System

We built our system based on an earlier system as described [1]. Users can monitor the workflow, preview the number of genes of significance, and the running status. JavaStat utilizes the Affymetrix-related R packages to implement the pre-processing workflow for the 3'-UTR-based array, gene-based arrays, and exon-based arrays. The process includes background adjustment, normalization, and summarization. Users can apply a combination of options in each step. The finished project is then saved as an easy-to-follow tree structure as shown in the Fig. (**2**).

JavaStat uses Java Web Start technology that can be deployed over a local user network with a single click. It ensures that the most current version of the application is deployed to the client desktop. Thus it offers maintenance free benefits for clients. Although Web a application could be

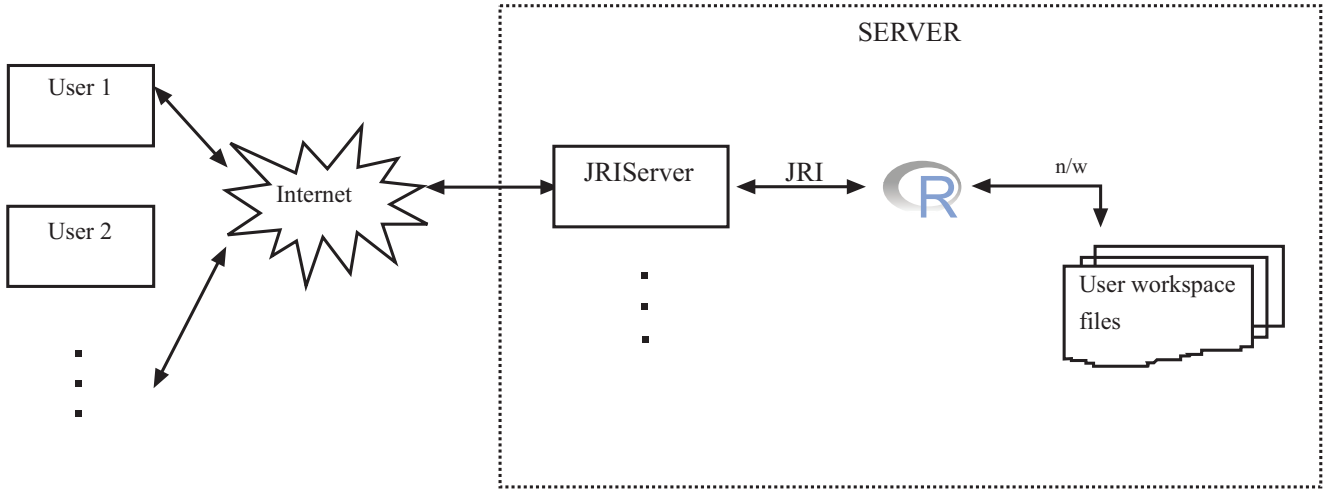

Fig. (1). Software architecture and user data management.

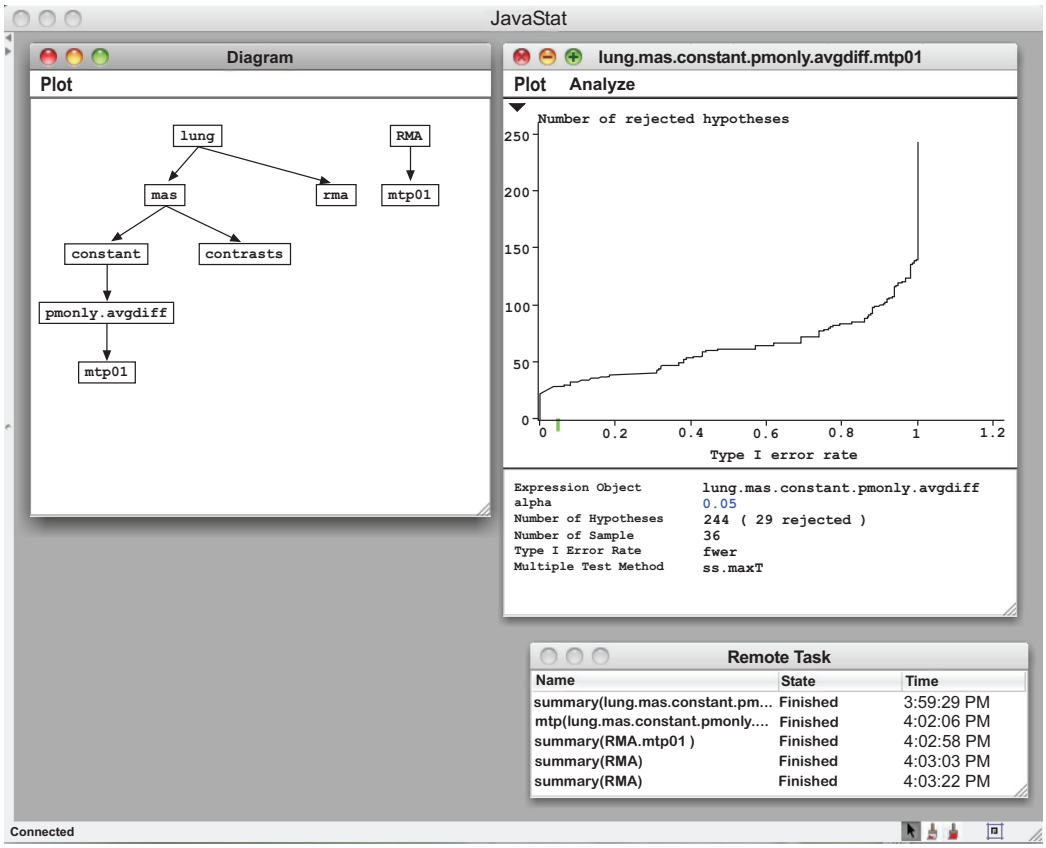

Fig. (2). JavaStat: a Java-based platform for array statistics**.**

used as a frontend for cloud computing, it lacks dynamic graphics and other features of the Java frontend unless certain plug-ins are used. JavaStat supports data management in spreadsheet like sorting, merging, saving, etc. These features are difficult to implement using HTML technology alone. Most of the Bioconductor commands take time to finish, JavaStat uses threading to poll results from the server to make it faster. Usually a Web system has to send the user emails when the results are ready.

JavaStat can perform multiple hypothesis testing on tab-delimited text files. Thus the user can preprocess raw data in the Affymetrix Expression Console and upload it to the cloud for multiple hypothesis testing (using the 'multtest' R package) to find significant genes between treatment and control groups, as shown in the Fig. (**3**).

Additional normalization methods such as GC-RMA for Exon arrays were incorporated for Affymetrix-based arrays in JavaStat as shown in the Fig. (**4**). The modified system offers options to adjust expression levels based on GC contents for 3' arrays, gene arrays, and exon arrays. These features were not available in GeneSpring at the time of development.

Various adjustment options for multiple t-tests can also be applied. The results including raw data, p value, adjusted p values, gene IDs and names, etc, can be downloaded for downstream analysis such as pathways analysis. Other annotation files such as Gene Ontology (GO) annotations can also be merged to the result as shown in the Fig. (**5**).

Another important option available to the user is to import and export .cel files in Affymetrix-required format, but with normalization applied in JavaStat. Users can then utilize the modified .cel file within the software of their choice for further analysis. Normalization applied directly to .cel files are carried out by an R package named aroma.affymetrix in CRAN repository. Users can also choose to continue their analysis within JavaStat.

In summary, for additional pre-processing or normalizations, we developed the following procedure to implement them as they appear. This includes: 1) Administrator installs new Bioconductor packages in R. 2) The software developer modifies the Java program to call function of new normalization methods in R. Java code may be modified on both the server side and client side. 3) Server programs are deployed to the cloud. Client programs are deployed to Web server. 4) At the next launch of JavaStat, the client will automatically detect and install the updates. Then the new function will be available to the users.

Implementation Issues and Solutions

With the use of R/Bioconductor as a backend computing engine, the effort of programming is significantly reduced. But embedding R in an application is a two-edged sword. The debugging process may be difficult because R packages are usually hard to implement by users although the developers may provide source codes. For example, at the time when we were developing this software, the "aroma" package initially could not handle missing values in the exon array; the "multtest" package for multiple t-tests prompts an

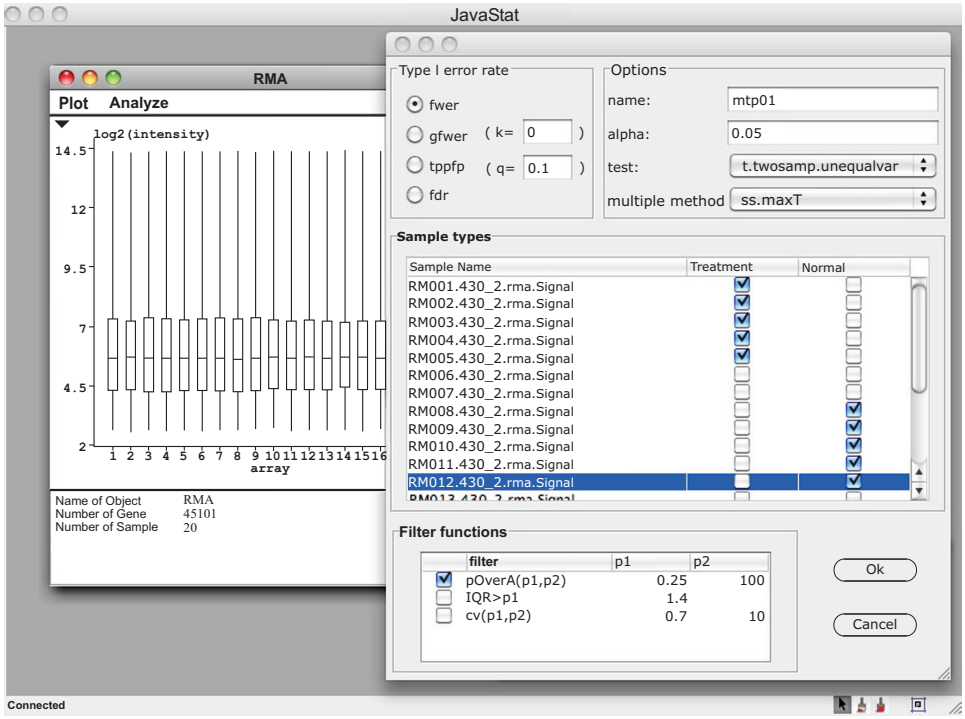

Fig. (3). QC, grouping and statistics in JavaStat.

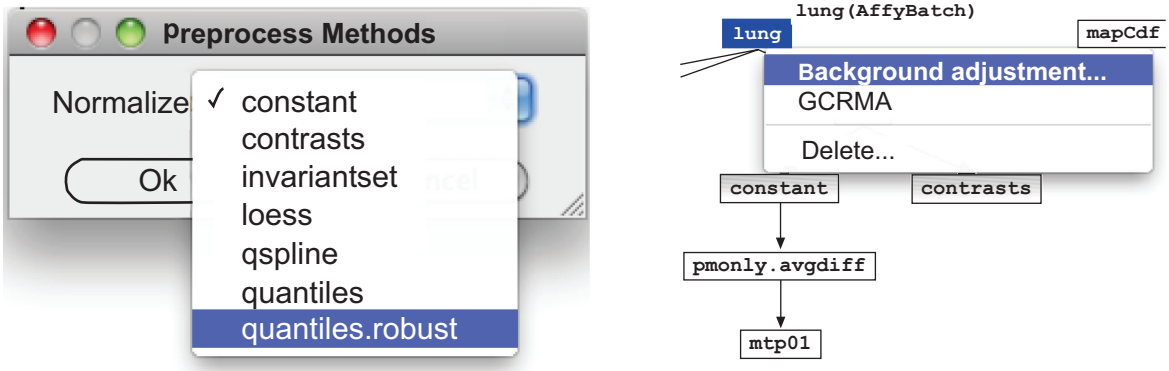

Fig. (4). Data preprocessing and normalization.

error if duplicated values exist in one group. Since we rely on the authors of the packages to fix those problems, it is unpredictable and time-consuming since package owners are not obliged to provide customer support. Another problem is that R packages are not necessarily backward compatible. It sometimes erases all installed packages when installed or upgraded. R packages need a longer life span or continued supports from owner are needed. Backward compatibility is crucial for broad applications. Our approach of implementing packages on a server lessens the requirement since all installations are centrally monitored by server administrator and client software is not affected.

CONCLUSION

The newly emerging cloud computing concept provides us a new way to provide an easily accessible service to biologists, as evidenced by our JavaStat system to incorporate new pre-processing packages as they appear. Users can access the Web-based application to upload data, run normalizations and analyses, and download results without much input from a biologist or a programmer. Similar system will greatly increase the accessibility of new algorithms in R.

AVAIABILITY AND REQUIRMENT

- Project name: JavaStat.
- Project home page: http://javastat.stat.wvu.edu.
- Operating system(s): Platform independent.
- Programming language: Java.
- Other requirements: Java 1.3 or higher.
- License: Free access for academic users.
- Any restrictions to use by non-academics: yes.

LIST OF ABBREVIATION

GC-RMA = Guanine Cytosine Robust Multi-Array Analysis.

Rejected Genes

Table Rows Columns Analyze Graph

Obs	C	M	gene	adjp	rawp	log2fc	CL200...	CL200
1	●		1596_g_at	0.0	0.0	-2.0761	83.0286	71.7
2	●		1929_at	0.0	0.0	-1.8467	56.6642	37.8
3	●		32052_at	0.0	0.0	-2.3758	4820.2891	969
4	●		32542_at	0.0	0.0	-2.4202	251.3645	108.0
5	●		32905_s_at	0.0	0.0	-2.0934	530.8479	274.0
6	●		34194_at	0.0	0.0	-1.9573	15.6506	25.0
7	●		34708_at	0.0	0.0	-2.6848	190.4287	94.0
8	●		35730_at	0.0	0.0	-2.4859	167.7163	26.4
9	●		35868_at	0.0	0.0	-3.0397	156.7576	173.4
10	●		36119_at	0.0	0.0	-1.8239	125.3238	135.1

Fig. (5). Results can be displayed together with annotations.

HTML = Hypertext Markup Language.

GO = Gene Ontology.

AUTHORS' CONTRIBUTION

DC is the PI, DL is a system developer and administrator and PB is an R and Java programmer. Together with system architect JH who is the mentor of DL and PB, and JM who provided advices from application perspectives, all contributed to system design, development and implementation.

ACKOWLEDGEMENT

The authors would like to thank colleagues in WVU Center of Neurosciences and Clinical Translational Sciences Institute (CTSI) for helpful discussions, and Claire S. Chen for proof reading. The project is partially support by a COBRE grant from NCRR, NIH to D. Luo, P. Banerjee and Institutional funding from WVU CTSI.

CONFLICT OF INTERESTS

The authors do not declare any conflict of interest in this study.

APPENDIX: Procedure *Readme* for accessing JavaStat

1. Contact the JavaStat administrator for a user name and password.
2. Install Java on your computers.
3. Download the application from http://javastat.stat.wvu.edu. Since JavaStat uses Java Web Start, no other installation process is needed.
4. Users have two options to perform the analysis. One is to upload CEL files for analysis. This kind of analysis will include array preprocessing and then differential gene expression analysis. The other option is to perform your normalization process in Affymetrix Expression Console, then to output gene expression data to a tab delimited text file.
5. Click "R Login" to login to the server with your username and password.
6. Select menu "Workflow" -> "Upload". You can either upload a group of CEL files for 3' Affymetrix array, Exon array, or one tab delimited txt file that contains gene expression signal values.
7. Once the data is uploaded, select menu "Workflow" -> "Diagram Objects". A tree structure of objects will appear. If you left-click one object, the name and type of this object will be displayed. You can right-click each object to perform appropriate tasks such as normalization and multiple hypothesis testing. After selection, a dialog will prompt the user with choices for a particular task. For example, the dialog of "differential gene expression…" step will ask for the type I error rate, sample types, and sample filters, etc.
8. After a task is submitted to the server, the task panel will display the status of all the tasks running on the server. Depending on the size and complexity of your task, time to finish the task on the server may vary. You must leave JavaStat running to get the results back. Quitting the program at this time will cause the results to be lost.
9. You can double-click each object to load it from the server. After loading, a window that contains the graph presentation and report will be shown for this object. In the object window, you can select menu "Analysis" to perform an appropriate analysis. You can click the black triangle on the top-left corner of the graph to get the options menu. For example, in the object window, you can download adjusted p-values, log2 fold changes, and intensity values of the rejected genes from the options menu. Downloaded data in the table window can be saved as a .csv or tab-delimited text (.txt) file.

REFERENCES

[1] D. L. E. James Harner and J. Tan, "JavaStat: a Java/R-based statistical computing environment," *Comput. Stat.*, vol. 24, pp. 295-302, 2009.

[2] R. C. Gentleman, "Bioconductor: open software development for computational biology and bioinformatics," *Genome Biol.*, vol. 5, p. R80, 2004.

[3] E. Hubbell, "Robust estimators for expression analysis," *Bioinformatics*, vol. 18, pp. 1585-1592, Dec 2002.

[4] C. L. A. W. Wong, "The analysis of gene expression data: DNA-chip analyzer (dChip), " *Stat. Biol. Health*, pp. 120-141, 2003.

[5] R. A. Irizarry, "Summaries of affymetrix genechip probe level data," *Nucleic Acids Res.*, vol. 31, p. e15, Feb 2003.

[6] B. M. Bolstad, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics*, vol. 19, pp. 185-193, Jan 2003.

[7] R. A. I. Zhijin Wu, R. Gentleman, F. M. Murillo and F. Spencer, "A model-based background adjustment for oligonucleotide expression arrays," *J. Am. Stat. Assoc.*, vol. 99, pp. 909-917, 2004.

[8] W. M. Liu, "PQN and DQN: algorithms for expression microarrays," *J. Theor. Biol.*, vol. 243, pp. 273-278, Nov 21 2006.

Received: February 02, 2012 Revised: April 14, 2012 Accepted: April 16, 2012