

# SignS: an open-source and freely available web-based tool for gene selection and molecular signatures for survival and censored data

Ramón Diaz-Uriarte\*<sup>1</sup>

<sup>1</sup>Statistical Computing Team, Structural Biology and Biocomputing Programme, Spanish National Cancer Center (CNIO), Melchor Fernández Almagro 3, Madrid, 28029, Spain

Email: Ramon Diaz-Uriarte\*- rdiaz02@gmail.com;

\*Corresponding author

## Abstract

---

SignS is an open-source, freely-available, web-based tool for gene selection and building molecular signatures with survival data. We use parallel computing via MPI. Predictive performance and stability of solutions is assessed via cross-validation. Genes and signatures in models can be sent to PaLS and IDConverter for examination of PubMed references, GO terms, KEGG and Reactome pathways of selected genes.

---

## Rationale

Many microarray studies involve human samples for which survival data are available. In the last two years there has been an increase in the number of new methods proposed for this kind of data [1–10]. Many of these papers have emphasized not only gene selection and survival prediction, but also “signature finding”: discovering sets of correlated genes that are relevant for survival prediction. For end-users (e.g., biomedical researchers with microarray data for a sample of patients for which survival is known), however, most of these methods are not easily accessible, which might explain why many papers in the primary biomedical literature implement from scratch varied ad-hoc approaches for gene selection in the context of survival prediction. Unfortunately, in many cases, survival data are reduced to arbitrarily determined classes, with

the consequent loss of information, simply because tools for class prediction are much more widely available. Thus, tools for end users are badly needed that, while retaining their user-friendliness, do not compromise statistical rigor.

Statistically, and in addition to appropriate analysis of censored data, such a tool should ensure that selection biases [11–14] are accounted for, to prevent overoptimistic assessments of the quality of the final model selected. Moreover, such a tool should also present the user with assessments of the stability of the results obtained: variable selection with microarray data (in general, in scenarios where the number of variables  $\gg$  than the number of samples) can lead to many solutions that have similar prediction errors, but that share few common genes [15–17]. Choosing one set of genes without awareness of the multiple solutions can create a false perception that the selected set is distinct from the rest of the genes. In addition to the statistical features, interpretation of results is enhanced if the tool provides additional information about “the interesting genes” such as PubMed references, Gene Ontology terms, and links to the UCSC and Ensembl databases and KEGG and Reactome pathways.

Such a tool should also try to incorporate the increasing availability of multicore processors and easily accessible clusters made with off-the-shelf components: the major opportunities for significant performance gains and the ability to analyze ever larger data sets do not lie in faster CPUs but rather in the increase in the available number of CPUs and CPU cores and, thus, in our ability to efficiently use parallel, distributed, and concurrent programming [18,19]. Parallelization, such as provided by MPI (Message Passing Interface) [20], allows to distribute the computations over a computing cluster, thus decreasing execution time. For an end user, parallelization can result in dramatic decreases in the time she/he needs to wait for the analysis to complete (see Benchmarking section).

Regarding the user interface, web-based applications have been gaining popularity in bioinformatics among other reasons because they allow the development of user-friendly applications that do not require software installation or upgrades from the user. In addition, web-based applications, if run in a computing cluster and implemented appropriately, make it possible to exploit parallelization.

Finally, source code availability under an open-source license allows researchers to improve upon the method, fix bugs, and verify claims by method developers, encourages reproducible research, and ensures that tool ownership resides in the international research community. These are all issues of particular concern in bioinformatics, which allow expedite progress that builds upon previous research [21,22].

We have developed a web-based tool, SignS, to fulfill the above needs. We know of no equivalent tool, and only BRB-ArrayTools, <http://linus.nci.nih.gov/BRB-ArrayTools.html>, by R. Simon and A. P. Lam,

provide somewhat similar functionality, but it is not web-based, does not ease accessing additional information, does not use parallel computing, and source code is not available. Thus, SignS is a unique tool, of immediate utility for biomedical researchers studying gene expression and its relation to survival (as is common, for example, in many cancer studies), and of broad appeal also to computational biologists, biostatisticians, and bioinformaticians because of the methods it implements and the combination of parallelization with web-based computing in an open-source application.

## Algorithms

There are few methods that explicitly attempt to perform gene selection with survival data while preserving the identity of the individual genes and allowing the recovery of highly correlated genes. Moreover, there are almost no comparisons among the available methods, except those from [2, 8, 23]. In this context, we chose to implement two very different approaches: those of Dave et al. [1] and Gui and Li [2]. The few available published comparisons suggest that both methods work reasonably well, with the method of Gui and Li [2] showing somewhat better predictive performance (at the cost of a large increase in computational cost).

Briefly, the steps of the method by Dave et al. [1] are: a) genes are filtered by p-value using a gene-by-gene Cox model; b) the retained genes are divided into those with a positive coefficient in the Cox model and those with a negative coefficient, and each group is clustered separately; c) a potential signature is a group of genes with a minimal (user-selected) correlation among themselves, and that have between a minimum and a maximum number of genes (again, parameters set by the user); d) the numeric value of a component, signature in the sense of [1] is the average expression level of all the genes in a given component; e) finally, we carry out stepwise variable selection using as starting point the best two-signature model. The main advantages of the approach of [1] are that it is easy to understand, that it explicitly attempts to return sets of correlated genes (signatures), and that the user is both forced to be explicit about, and allowed to choose, parameters with relatively straightforward mapping to the biology (such as the minimum correlation of genes within a signature, or the maximum and minimum number of genes in a signature). Thus, the method of [1] is ideal for exploratory analysis, which is further enhanced by our additions, in particular the assessment of stability of solutions and functional annotation via IDconverter and PaLS (see “Cross-validation and stability assessments” section). Finally, the approach of [1] is itself an influential biomedical paper.

In contrast, the approach of Gui and Li [2] has essentially no tunable parameters. The complete method,

including the dimensional reduction and the ability to capture correlated genes, follows from the penalization used (penalization is a general approach —with other examples being the Lasso and ridge regression— used to obtain estimates of the regression coefficients in high-dimensional spaces such as these, with few subjects and thousands of features). The solutions of this method depend on two parameters: a threshold parameter  $\tau$  and an infinitesimal increment  $\Delta\nu$ . The threshold parameter  $\tau$ , constrained between 0 and 1, is related to the amount of penalization, and larger values lead to a larger numbers of coefficients being set to 0 (i.e., to the selection of a smaller number of genes). The infinitesimal increment  $\Delta\nu$  affects the rate of change of the coefficients at each iteration of the algorithm. Note that, operationally, we can instead choose a sufficiently small  $\Delta\nu$ , and modify the number of iterations. [2] use cross-validation to choose the optimal parameters: a set of  $\tau$  is decided in advance, and cross-validation is used to select the  $\Delta\nu$  (or, alternatively, the number of iterations), that minimizes the cross-validated partial likelihood (CVPL). The CVPL is a function of the difference in the log Cox’s partial likelihoods for the models with and without the  $i$ th group of subjects. Once the optimal parameters are decided, it is immediate to obtain the fitted coefficients for the genes and the scores for subjects not in the sample. Thus, if we choose a sufficiently small  $\Delta\nu$ , as we do in SignS, no parameters need to be chosen by the user, as these are selected via cross-validation.

## Implementation

The method of Dave et al. [1] is available in both the web-based application and the underlying R code. The method of Gui and Li [2], however, is only available in the R code because it is too slow for routine use and we found that few users were selecting it, as it was of harder interpretation for users and did not provide clear prediction improvements. The core statistical functionality of both methods is written in R [24]. Where possible, computations have been parallelized using MPI (its LAM[Local Area Multicomputer]/MPI implementation) via the R-packages Rmpi (<http://www.stats.uwo.ca/faculty/yu/Rmpi>), by H. Yu, Snow (<http://cran.r-project.org/src/contrib/Descriptions/snow.html>), by L. Tierney, A. J. Rossini, Na Li and H. Sevcikova, and papply (<http://ace.acadiau.ca/math/ACMMaC/software/papply/>) by D. Currie. For the web-based application, the CGI (Common Gateway Interface), initial data validation, and the setting-up and closing of the parallel infrastructure (booting and halting the LAM/MPI universes) are written in Python.

The implementation of the approach in Dave et al. [1] follows closely the description of their method in the

supplementary material to their paper. Our main departures from their implementation are: a) we do not split the data into two halves, but instead use cross-validation to assess model selection; b) it is unclear how the initial two-signature model was selected by the authors, and we choose the one with the largest likelihood, which in this case would be identical to using AIC, Akaike Information Criterion, as a criterion, since all two-signature models have the same number of parameters; c) it seems, from the supplementary material, that the authors used p-values in their forward variable selection, whereas we used AIC, generally a preferred criterion for model selection [25], with stepwise variable addition (i.e., at each step we re-evaluate if any variables removed in previous steps should be incorporated, or any variables previously introduced into the model should be eliminated; if stepwise model selection leads to numerical problems we automatically switch to forward variable selection).

For the method of Gui and Li [2] our code is based on the original code by the authors, with many modifications for speed improvement and parallelization. First, several common operations in the code were implemented using faster (sequential) code (e.g., using “crossprod” instead of the naive  $\mathbf{X}'y$ , vectorizing loops, rewriting expressions to use fewer steps, etc). Next, the code was parallelized. Taking into account the number of nodes we had available and the number of nodes that can often be used in off-the-shelf computing clusters, we parallelized the computations that search for the best  $\tau$ . Following [2], we explore the six possible  $\tau$  values 0, 0.2, 0.4, 0.6, 0.8, 1.0 and select the one that minimizes the CVPL, using 10-fold crossvalidation. Thus, we can parallelize the finding of the best  $\tau$  into 60 independent computations (10 searches at each of the candidate  $\tau$ ). Notice that parallelizing at this level yields increased speed even if no global cross-validation is performed. The speed-ups achieved with the code changes are discussed below. SignS can run from a laptop to a cluster of workstations. Our installation runs on a cluster of 30 computing nodes, each with two dual-core AMD Opteron CPUs and 6 GB of RAM. Additional nodes provide load-balancing, high-availability, and shared storage. The input for the web-based application are either plain text files, or files that come from other tools of the Asterias suite (<http://asterias.bioinfo.cnio.es>). Further documentation and examples for the web-based application are available from its on-line help (<http://signs2.bioinfo.cnio.es/help/index.html>). SignS has been running in production use for over a year, and bug-tracking is available from <http://bioinformatics.org/asterias>. SignS also includes a test suite that uses FunkLoad (<http://funkload.nuxeo.org>); the tests allow to verify that the user interface and numerical output are working, thus ensuring appropriate quality control and regression testing.

### **Cross-validation and stability assessments**

To provide estimates of the performance of the final model and analysis of stability we use 10-fold CV (cross-validation). To assess predictive performance we use a simple and common [1, 2, 6–8, 10, 23, 26–28] strategy: splitting the test samples into several (2, 3, or 4) groups based on their predicted scores (or predicted survival), and comparing their survival. It must be emphasized that the predicted scores are obtained from a full cross-validation, so the predicted scores for a sample correspond to the CV-fold for which that sample never participated in any of the steps above. For stability analysis, we report the number of signatures and the genes in each signature for the run with the original sample and the 10 CV runs. The list of these signatures and genes can be sent to our application PaLS (<http://idclight.bioinfo.cnio.es>) to examine PubMed references, Gene Ontology terms, KEGG pathways or Reactome pathways that are common to a user-selected percentage of genes and/or signatures. Tables with output from each run include clickable links to our application IDClight (<http://idclight.bioinfo.cnio.es>) [29] which provides additional information, including mapping between gene and protein identifiers, PubMed references, Gene Ontology terms, and KEGG and Reactome pathways.

### **Benchmarking**

The speedups achieved in the method by Gui and Li [2] with our code changes and parallelization are shown in Figure 1. We achieve speedups of a factor of 100x when we run 2 slave Rmpi processes per node (each node has 2 dual-core CPUs). Even when running 60 slave processes per computing node, we achieve speedups of a factor of 10x. Note that this is a situation where we are forcing the operating system scheduler to perform extra work (and these results are from machines running a stock Debian GNU/Linux 2.6.8 kernel, without any changes to the scheduler).

Figure 2 shows the average time a user will wait for the analysis to complete (user wall time) as a function of the number of simultaneous users using the application in that very moment. (When using a slow internet connection these numbers will increase and, e.g., uploading a data set of 8.5 MB, such as DLBCL, to the application can take over 5 minutes). As can be seen from the figure, SignS can handle a large number of simultaneous users. This is the result of both the parallelization of the computations and the load balancing, which balances the non-parallelized code. Note that situations with 10 or more simultaneous users are completely unrealistic, since the average number of daily users of SignS is less than 4. The above benchmarks, though, show that SignS can handle even those high numbers of users, which makes it suitable for classroom use.

Scripts for all benchmarks are available from the repositories.

## Availability

Web-based application: <http://signs2.bioinfo.cnio.es>. All source code, including the web-based application, R code, and functional tests, available from Bioinformatics.org (<http://bioinformatics.org/asterias>) or The Launchpad (<http://launchpad.net/asterias>).

## List of abbreviations used

**AIC** Akaike Information Criterion

**CGI** Common Gateway Interface

**CV** Cross-validation

**CVPL** Cross-validated partial likelihood

**GO** Gene Ontology

**KEGG** Kyoto Encyclopeida of Genes and Genomes *zz*

**LAM** Local Area Multicomputer

**MPI** Message Passing Interface

## Acknowledgements

A. Alibés and A. Cañada for IDClight and PaLS. C. Lázaro-Perea and an anonymous reader for comments on the ms. Bioinformatics.org and The Launchpad for repository hosting. Funding provided by Fundación de Investigación Médica Mutua Madrileña and Project TIC2003-09331-C02-02 of the Spanish MEC.

R.D.-U. partially supported by the Ramón y Cajal programme of the Spanish MEC.

## References

1. Dave SS, Wright G, Tan B, Rosenwald A, Gascoyne RD, Chan WC, Fisher RI, Braziel RM, Rimsza LM, Grogan TM, Miller TP, LeBlanc M, Greiner TC, Weisenburger DD, Lynch JC, Vose J, Armitage JO, Smeland EB, Kvaloy S, Holte H, Delabie J, Connors JM, Lansdorp PM, Ouyang Q, Lister TA, Davies AJ, Norton AJ, Muller-Hermelink HK, Ott G, Campo E, Montserrat E, Wilson WH, Jaffe ES, Simon R, Yang L, Powell J, Zhao H, Goldschmidt N, Chiorazzi M, Staudt LM: **Prediction of survival in follicular lymphoma based on molecular features of tumor-infiltrating immune cells.** *N Engl J Med* 2004, **351**(21):2159–2169.
2. Gui J, Li H: **Threshold gradient descent method for censored data regression with applications in pharmacogenomics.** *Pac Symp Biocomput* 2005, :272–283.

3. Hothorn T, Bühlmann P, Dudoit S, Molinaro A, van der Laan MJ: **Survival Ensembles**. *Biostatistics* 2006, **7**(3):355–373.
4. Bair E, Tibshirani R: **Semi-supervised methods to predict patient survival from gene expression data**. *PLoS Biol* 2004, **2**(4).
5. Bair R, Hastie T, Paul D, Tibshirani R: *Journal American Statistical Association* 2006, **101**:119–137.
6. Kaderali L, Zander T, Faigle U, Wolf J, Schultze JL, Schrader R: **CASPAR: a hierarchical bayesian approach to predict survival times in cancer from gene expression data**. *Bioinformatics* 2006, **22**(12):1495–1502.
7. Park PJ, Tian L, Kohane IS: **Linking gene expression data with patient survival times using partial least squares**. *Bioinformatics* 2002, **18**:S120–127.
8. Ma S, Huang J: **Clustering Threshold Gradient Descent Regularization: with applications to microarray studies**. *Bioinformatics* 2006, :btl632.
9. Sha N, Tadesse MG, Vannucci M: **Bayesian variable selection for the analysis of microarray data with censored outcomes**. *Bioinformatics* 2006, **22**(18):2262–2268.
10. Ma S, Kosorok MR, , Fine JP: **Additive Risk Models for Survival Data with High-Dimensional Covariates**. *Biometrics* 2006, **62**:202–210.
11. Ambroise C, McLachlan GJ: **Selection bias in gene extraction on the basis of microarray gene-expression data**. *Proc Natl Acad Sci USA* 2002, **99**(10):6562–6566.
12. Simon R, Radmacher MD, Dobbin K, McShane LM: **Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification**. *Journal of the National Cancer Institute* 2003, **95**:14–18.
13. Varma S, Simon R: **Bias in error estimation when using cross-validation for model selection**. *BMC Bioinformatics* 2006, **7**.
14. Dudoit S, Fridlyand J: **Classification in microarray experiments**. In *Statistical analysis of gene expression microarray data*. Edited by Speed T, New York: Chapman & Hall 2003:93–158.
15. Somorjai RL, Dolenko B, Baumgartner R: **Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions**. *Bioinformatics* 2003, **19**:1484–1491.
16. Pan KH, Lih CJ, Cohen SN: **Effects of threshold choice on biological conclusions reached during analysis of gene expression by DNA microarrays**. *Proc Natl Acad Sci U S A* 2005, **102**:8961–8965.
17. Díaz-Uriarte R, Alvarez de Andrés S: **Gene selection and classification of microarray data using random forest**. *BMC Bioinformatics* 2006, **7**.
18. Sutter H: **The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software**. *Dr. Dobbs's Journal* 2005, **30**(3):202–210.
19. Kontoghiorghes EJ: *Handbook of Parallel Computing and Statistics*. Boca Raton, FL: Chapman & Hall, CRC 2006.
20. Pacheco P: *Parallel programming with MPI*. San Francisco: Morgan kaufman 1997.
21. Dudoit S, Gentleman RC, Quackenbush J: **Open source software for the analysis of microarray data**. *Biotechniques* 2003, **Suppl**:45–51.
22. Díaz-Uriarte R: **Supervised methods with genomic data: a review and cautionary view**. In *Data analysis and visualization in genomics and proteomics*. Edited by Azuaje F, Dopazo J, New York: Wiley 2005:193–214.
23. Gui J, Li H: **Penalized Cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data**. *Bioinformatics* 2005, **21**(13):3001–3008.
24. R Development Core Team: *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria 2004. [ISBN 3-900051-00-3].
25. Harrell F: *Regression Modeling Strategies*. New York: Springer 2006.
26. Li H, Luan Y: **Boosting proportional hazards models using smoothing splines, with applications to high-dimensional microarray data**. *Bioinformatics* 2005, **21**(10):2403–2409.

27. Li H, Gui J: **Partial Cox regression analysis for high-dimensional microarray gene expression data.** *Bioinformatics* 2004, **20**:i208–215.
28. Pawitan Y, Bjöhle J, Wedren S, Humphreys K, Skoog L, Huang F, Amler L, Shaw P, Hall P, Bergh J: **Gene expression profiling for prognosis using Cox regression.** *Stat Med* 2004, **23**(11):1767–1780.
29. Alibés A, Yankilevich P, Cañada A, Diaz-Uriarte R: **IDconverter and IDClight: conversion and annotation of gene and protein IDs.** *BMC Bioinformatics* 2007, **8**.

## Figures

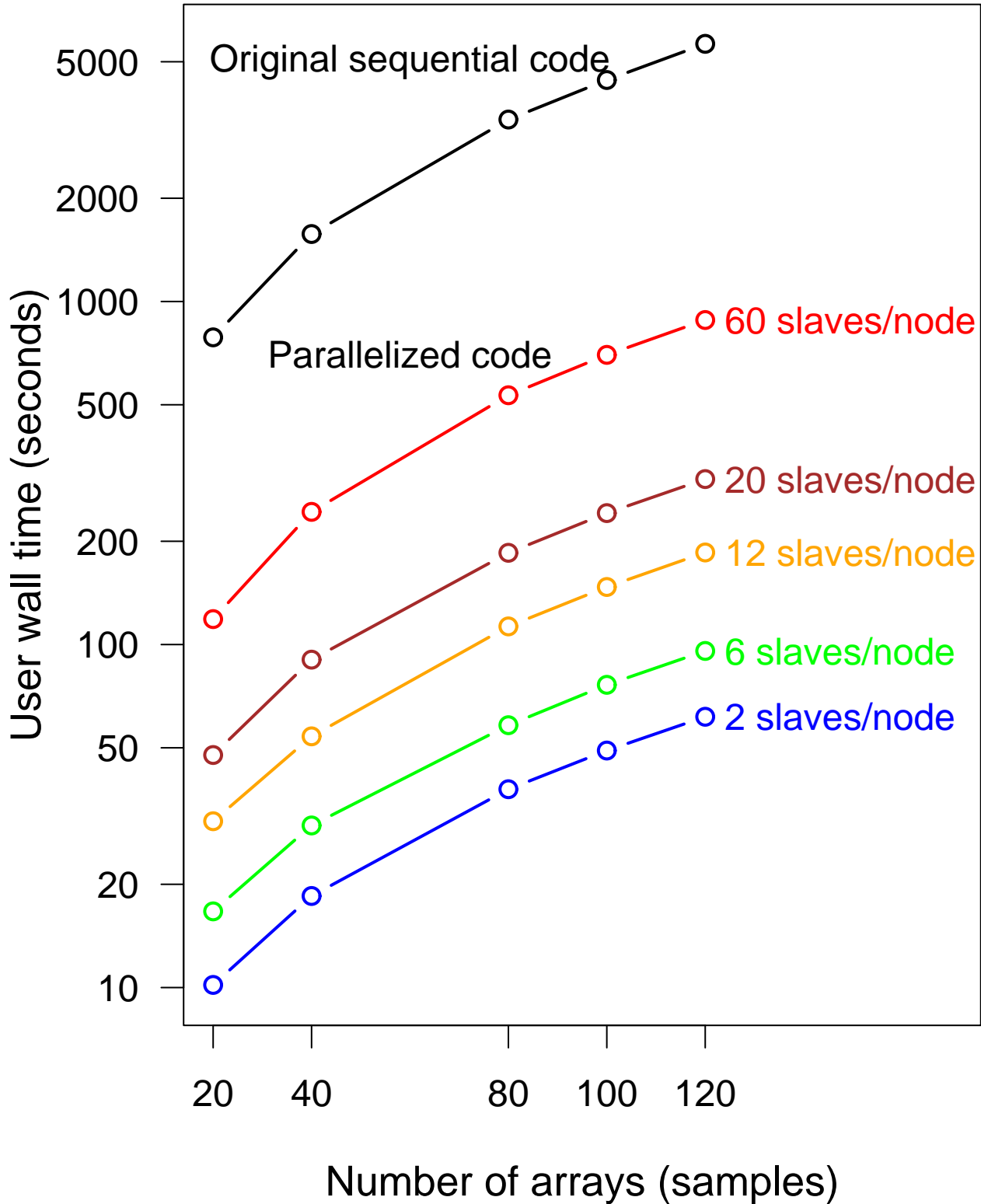
### **Figure 1 - Effects of parallelization and other code changes in the speed of the threshold gradient descent method**

Shown are user wall times for a data set (with varying number of either subjects or genes) when using the function “gdcvpl” (in the original code) and its equivalent “tauBestP” (SignS) which employ cross-validation to find the best parameters. Values represented are the mean of five replicate runs, performed in an otherwise idle cluster, with 30 nodes, each with two dual-core AMD Opteron 2.2 GHz CPUs and 6 GB RAM, running Debian GNU/Linux and a stock 2.6.8 kernel, with version 7.1.2 of LAM/MPI and version 2.1.4 (patched) of R. The data were obtained by sampling genes and subjects randomly from the DLBCL data set used in Fig. 2.

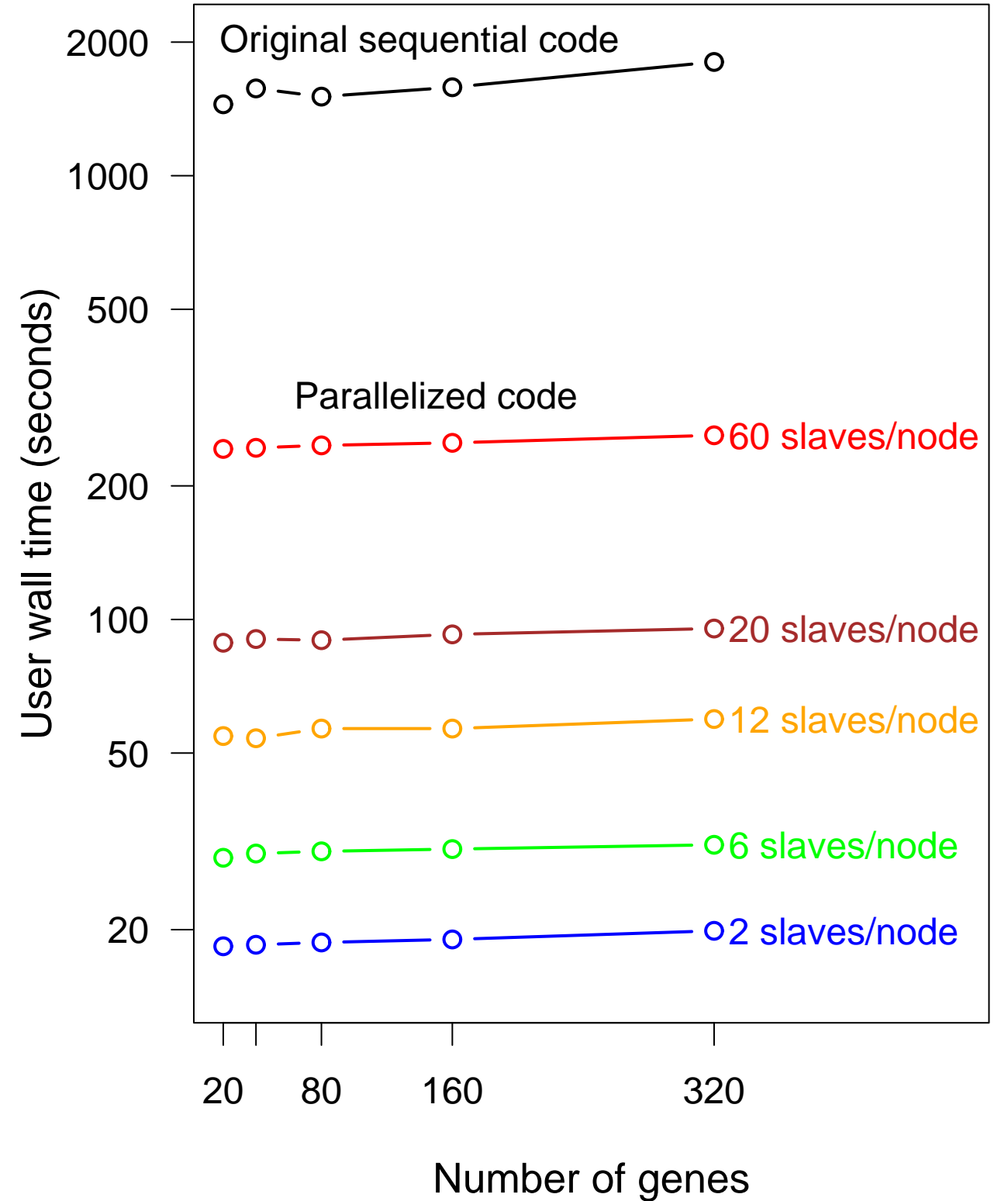
### **Figure 2 - User wall time of the web-based application as a function of number of simultaneous users for two different (and real) data sets, obtained from [4]**

To increase the realism of simultaneous accesses, there is delay of 5 seconds between simultaneous accesses, as might be expected, for example, from a classroom demonstration (i.e., when simulating 20 simultaneous users, the cluster is actually receiving new connections over a  $20 * 5$  second period, with one new connection every 5 seconds). Values shown are the mean of several runs: 10 for 1 user, 10 for 5 users, 20 for 10 users, 15 for 15 users, 40 for 20 users, and 50 for 50 users. Hardware and software the same as in Figure 1.

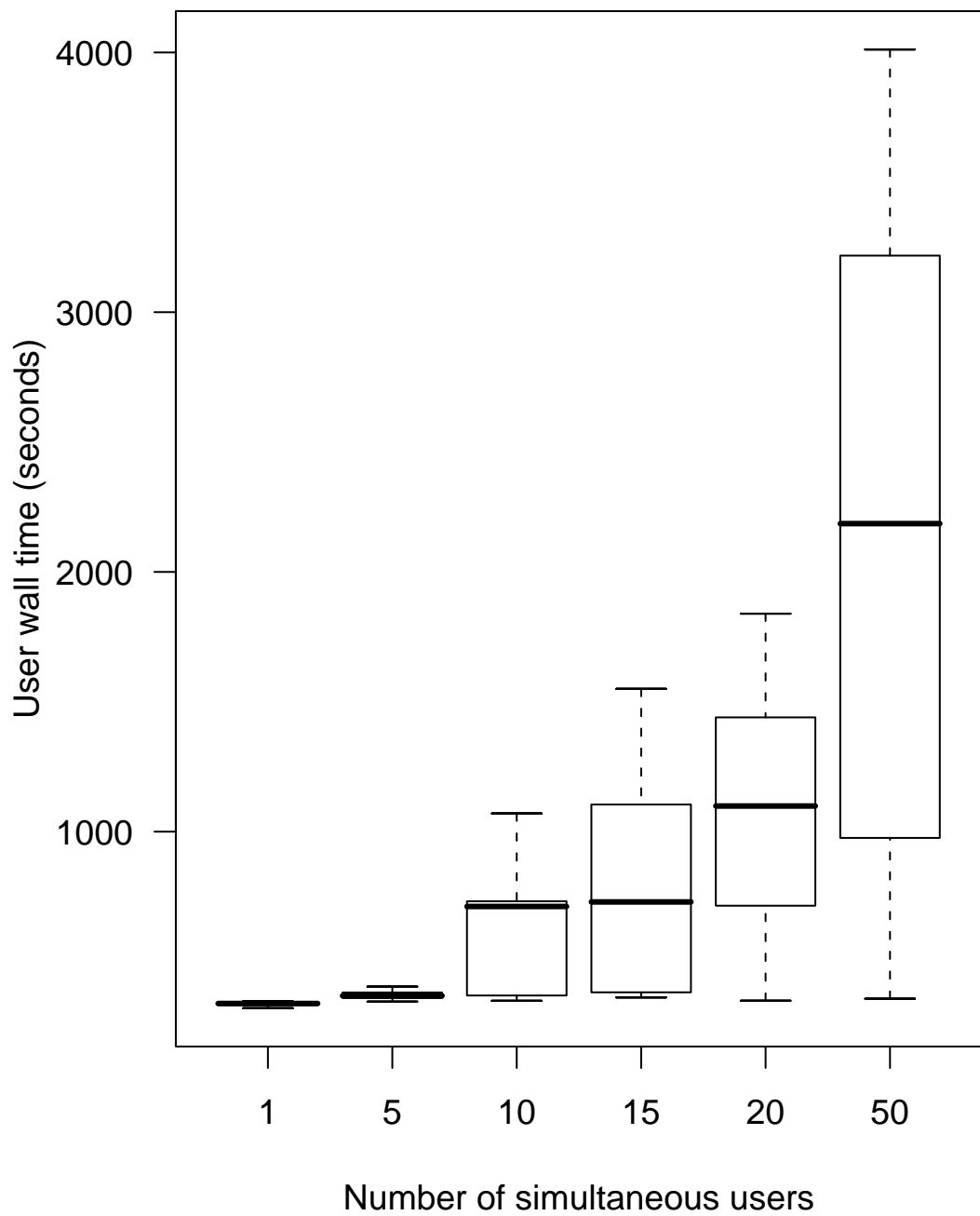
Effect of number of arrays (number of genes = 40)



Effect of number of genes (number of arrays = 40)



**Breast data set (78 arrays x 4751 genes)**



**DLBCL data set (160 arrays x 7399 genes)**

