OXFORD UNIVERSITY PRESS | Bioinformatics

runibic: a Bioconductor package for parallel row-based biclustering of gene expression data

Journal:	*Bioinformatics*
Manuscript ID	BIOINF-2018-0073.R1
Category:	Applications Note
Date Submitted by the Author:	27-May-2018
Complete List of Authors:	Orzechowski, Patryk; University of Pennsylvania Perelman School of Medicine, Biostatistics, Epidemiology, and Informatics Panszczyk, Artur; AGH University of Science and Technology, Automatics and Biomedical Engineering Huang, Xiuzhen; Arkansas State University, Computer Science Moore, Jason; University of Pennsylvania, Genetics
Keywords:	Machine learning, Cluster analysis, Gene expression, Software, Bioconductor

"main-revision1" — 2018/5/25 — 22:13 — page 1 — #1

Gene expression

runibic: a Bioconductor package for parallel row-based biclustering of gene expression data

Patryk Orzechowski [1,2]*, Artur Pańszczyk [2], Xiuzhen Huang [3], and Jason H. Moore [1,*]

[1]Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA 19104, USA,
[2]Department of Automatics and Biomedical Engineering, AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Krakow, Poland, and
[3]Department of Computer Science, Arkansas State University, Jonesboro, AR 72467, USA

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Biclustering is an unsupervised technique of simultaneous clustering of rows and columns of input matrix. With multiple biclustering algorithms proposed, UniBic remains one of the most accurate methods developed so far.
Results: In this paper we introduce a Bioconductor package called *runibic* with parallel implementation of UniBic. For the convenience the algorithm was reimplemented, parallelized, and wrapped within an R package called *runibic*. The package includes: (1) a couple of times faster parallel version of the original sequential algorithm, (2) much more efficient memory management, (3) modularity which allows to build new methods on top of the provided one, and (4) integration with the modern Bioconductor packages such as *SummarizedExperiment*, *ExpressionSet* and *biclust*.
Availability: The package is implemented in R (3.4) and is available from Bioconductor (3.6) at the following URL *http://bioconductor.org/packages/runibic* with installation instructions and tutorial.
Contact: patryk.orzechowski@gmail.com, jhmoore@upenn.edu
Supplementary information: Supplementary informations are available in vignette of the package.

1 Introduction

The recent advantages in transcriptomic analysis, including development of high-throughput and high-resolution platforms including RNA-seq, Single-cell RNA-seq (scRNA-seq) or high-throughput PCR have allowed to design experiments that provide datasets with even hundreds of thousands columns and thousands rows. This have set new requirements for data analytics. Modern methods need to yield accurate results for large datasets and are expected to finish computations in reasonable time.

With growing amount of genomic data there is an urgent need for efficient and precise methods that are able to capture the underlying patterns in gene expression datasets. One of the techniques that proved to be very insightful in gene expression analysis is biclustering, which allows to detect subsets of genes and samples in complex and noisy data. Biclustering is considered NP-hard as it investigates relations between multiple rows that occur in different subsets of columns. The running time of the algorithms is usually highly dependent on the size of the input data.

The vast majority of existing biclustering methods are sequential. There are a couple of common reasons for this. Some methods are specifically designed to yield only one bicluster at a time. Each run of the algorithm depends on the previous findings. Other methods use graph-based structures, which are difficult to parallelize, or perform hardly scalable statistical analyses. For some group of the methods parallelization may even not be beneficial, as they extensively use binary operations. Bioconductor in version 3.5 provides the following biclustering methods and packages for gene expression analysis: *eisa* and *isa2* (Csardi *et al.*, 2010), *biclust* (Kaiser *et al.*, 2015), *fabia* (Hochreiter *et al.*, 2010), *hapfabia* (Hochreiter, 2013), *QUBIC* (Zhang *et al.*, 2017), *rqubic* (Zhang, 2015), MCbiclust (Bentham, 2017), *s4vd* (Sill and Kaiser, 2015), and *iBBiG* (Gusenleitner *et al.*, 2012). The vast majority of the aforementioned packages are implemented in R, which is slower than C. Some of the packages, e.g. *QUBIC*, benefit from calls to high-performance C++ linear algebra libraries, such as *Rcpp* (Eddelbuettel and François, 2011) and *RcppArmadilo* (Eddelbuettel and Sanderson, 2014). The comparison of R packages functionality is presented in Table 1. The detailed information on algorithms available within the packages could be found in Supplementary Material.

One of the recent breakthroughs in gene expression analysis was development of UniBic (Wang *et al.*, 2016). The algorithm originally implemented in C managed to capture biologically meaningful trend-preserving patterns and proved to outperform multiple other methods. The

Table 1. Comparison of functionalities of different R packages. (*) - Only Bimax algorithm uses wrapped C function call.

Description	runibic	QUBIC	biclust*	s4vd	fabia	isa2
Support for numeric and integer datasets	yes	yes	yes	yes	yes	yes
Parallel implementation of methods	yes	no	no	no	no	no
Integration with Biclust	yes	yes	yes	yes	no	yes
Integration with SummarizedExperiment	yes	no	no	no	no	no
Uses C/C++ routines	yes	yes	(*)	yes	yes	yes

method also showed great potential for parallelization. Unfortunately the implementation of the method wasn't efficient enough and the code had some memory leaks.

2 Methods

In this paper we introduce a Bioconductor package called *runibic* with parallel implementation of one of the most accurate biclustering methods: UniBic. The algorithm, originally released as sequential, has proven to outperform multiple popular biclustering state-of-the-art biclustering methods (Wang *et al.*, 2016). After code refactoring *UniBic* was reimplemented into more modern C++11 programming language. By parallelizing chunks of the code using OpenMP standard Dagum and Menon (1998), we obtained up to a couple of times speedup in terms of execution time for popular genomic datasets. With fixing some of the memory management bugs of the algorithm our package provides more stable and reliable implementation of UniBic algorithm. Starting from Bioconductor 3.7, a consistency with the original implementation is maintained by using *useLegacy=TRUE* flag in *runibic* function calls (no flag needs to be used for the improved version of UniBic).

The *runibic* package takes advantage of *Rcpp* library that allows seamless integration of C++ code with R environment. The *runibic* package is also integrated with *biclust* package methods for biclustering process. Results returned from *runibic* are wrapped into a *Biclust* object, which can be used for further examination, including visualization and analysis provided by *biclust* package.

```
library(runibic)
test <- matrix(rnorm(1000), 100, 100)
res <- runibic(test)
```

Similarly, the *biclust* method could be applied to any matrix extracted from *ExpressionSet* using $exprs()$ function. Multiple other examples explaining the usage of the package are presented in supplementary material as well as in the package manual available at Bioconductor.

Apart from allowing analysis of genomic data from historical *ExpressionSet*, *runibic* package is compatible with *SummarizedExperiment* class (Morgan *et al.*, 2017). This class offers more flexibility in terms of experiment design and supports both Single-cell RNA-seq and ChIP-seq. This makes *runibic* a very easy tool for performing modern biclustering analysis on different types of data. An example on using *runibic* with *SummarizedExperiment* class is provided in Supplementary Material.

3 Results

To investigate running times of the method, we have applied it to several popular datasets. The running times of the revised and the original UniBic algorithm as well as the revised parallel version are presented in Table 2.

The refactored and optimized *runibic* run up to over 8 times faster than the original implementation of the *Unibic* algorithm. The comparison of UniBic with other methods could be found in the original paper (Wang *et al.*, 2016).

Table 2. Running times of the original version of UniBic Wang et al. (2016) and parallel UniBic in R from Bioconductor package.

Dataset	Rows	Columns	UniBic run time(s)	runibic run time(s)	Improved
Escherichia coli	4297	466	1478.3	290.9	5.1x
GSE66913	16436	167	546.7	67.6	8.1x
GSE42408	25662	208	3305.3	821.6	4.0x
airway	64102	8	903.9	487.0	1.8x

4 Conclusions

In this paper we introduce *runibic* package with revised and parallelized version of UniBic biclustering algorithm. The package is available with the latest release of Bioconductor (3.6). Modular structure of the package improves interpretability of the method and adds more flexibility. The package provides *runibic* method that could be applied to any matrix in R, expression set extracted from *ExpressionSet* or *SummarizedExperiment* class. Integration with popular R and Bioconductor packages (e.g. *biclust*, *QUBIC*), as well as extensive documentation on one of the most accurate biclustering algorithms make *runibic* package very convenient to use.

Funding

This research was supported in part by PL-Grid Infrastructure and by grants LM012601, TR001263, ES013508 from the National Institutes of Health (USA).

References

Bentham, R. (2017). *MCbiclust: Massive correlating biclusters for gene expression data and associated methods*. R package version 1.0.1.

Csardi, G., Kutalik, Z., and Bergmann, S. (2010). Modular analysis of gene expression data with r. *Bioinformatics*, **26**, 1376–7.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, **5**(1), 46–55.

Eddelbuettel, D. and François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, **40**(8), 1–18.

Eddelbuettel, D. and Sanderson, C. (2014). Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics & Data Analysis*, **71**, 1054–1063.

Gusenleitner, D., Howe, E. A., Bentink, S., Quackenbush, J., and Culhane, A. C. (2012). ibbig: iterative binary bi-clustering of gene sets. *Bioinformatics*, **28**(19), 2484–2492.

Hochreiter, S. (2013). Hapfabia: identification of very short segments of identity by descent characterized by rare variants in large sequencing data. *Nucleic acids research*, **41**(22), e202–e202.

Hochreiter, S., Bodenhofer, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Van Sanden, S., Lin, D., Talloen, W., Bijnens, L., G"ohlmann, H. W. H., Shkedy, Z., and Clevert, D.-A. (2010). FABIA: Factor analysis for bicluster acquisition. *Bioinformatics*, **26**(12), 1520–1527. doi:10.1093/bioinformatics/btq227.

Kaiser, S., Santamaria, R., Khamiakova, T., Sill, M., Theron, R., Quintales, L., Leisch, F., and De Troyer., E. (2015). *biclust: BiCluster Algorithms*. R package version 1.2.0.

Morgan, M., Obenchain, V., Hester, J., and PagÃ¨s, H. (2017). *SummarizedExperiment: SummarizedExperiment container*. R package version 1.6.5.

Sill, M. and Kaiser, S. (2015). *s4vd: Biclustering via Sparse Singular Value Decomposition Incorporating Stability Selection*. R package version 1.1-1.

Wang, Z., Li, G., Robinson, R. W., and Huang, X. (2016). Unibic: Sequential row-based biclustering algorithm for analysis of gene expression data. *Scientific reports*, **6**.

Zhang, J. D. (2015). *rqubic: Qualitative biclustering algorithm for expression data analysis in R*. R package version 1.22.0.

Zhang, Y., Xie, J., Yang, J., Fennell, A., Zhang, C., and Ma, Q. (2017). QUBIC: a bioconductor package for qualitative biclustering analysis of gene co-expression data. *Bioinformatics*, **33**(3), 450–452.