Qualitative Biclustering with Bioconductor Package rqubic

Jitao David Zhang, Laura Badi and Martin Ebeling

May 3, 2016

Abstract

Biclustering has been suggested and found very useful to discover gene regulation patterns from gene expression microarrays. Several quantitative algorithms, among others *CC* and *BIMAX*, have been implemented in R, mainly by the *biclust* package. To our best knowledge, there have been so far no qualitative biclustering methods implemented.

Therefore we introduce *rqubic*, a Bioconductor package implementing the qualitative biclustering (QUBIC) algorithm. Compared to quantitative alternatives, this algorithm is less sensitive to outliers and heavy tails of microarray data. In addition, it is straightforward to plug other discretized data types in the algorithm, for example differentially expressed transcripts in NGS experiments, for which several R packages (e.g. *edgeR* and *DESeq*) have been available.

The vignette introduces the functionalities provided by the *rqubic* package. And we demonstrate the usage of the package by implementing a biclustering software pipeline.

1 Introduction

R and Bioconductor package *rqubic* implements a qualitative biclustering algorithm, *QUBIC*, first introduced by [**?**]. The input data is typically a $N \times M$ matrix representing expression levels of N genes in M samples. The algorithm first applies quantile discretization. In its basic mode, the discretization classifies any gene into three categories: expression up-regulated,down-regulated or unchanged. A more refined discretization is possible by increasing ranks, which are defined as the number of levels in one changing direction (therefore the simple example above has a rank of one). A rank of two, for example, allows to discretize the matrix into five levels (-2, -1, 0, +1 and +2, or very weak, weak, normal, strong, very strong).

Once the discretized matrix is available, a heuristic algorithm is applied to identify biclusters. This procedure starts with setting seeds, which are pairs of gene sharing expression patterns in a number of samples. Biclusters are identified from these seeds, by searching for other genes sharing the expression patterns, or for those which have an even

similar match. This step is repeated for all generated seeds, and a number of maximal biclusters (defined as the number of genes times the number of conditions in that bicluster) are reported.

2 Functionality

The *rqubic* package implements data structures and functionalities to perform biclustering with the QUBIC algorithm in R. In addition it provides a set of tools to parse QUBIC program outputs into R data structures, so as to enable further analysis of existing QUBIC biclustering results.

The QUBIC algorithm is to large extent in ANSI-C implemented, so as to save memory usage and to speed the biclustering. It shares codes with the QUBIC implementation in C, with modifications in the program structure as well as implementation details. The basic R implementation of the QUBIC algorithm provides consistent results on a given dataset with the C implementation. The R implementation allows flexible further development of the algorithm.

The parsers for QUBIC C program outputs are straight-forward implemented in R functions. Their uses and examples can be found by executing following codes in R.

```
> library(rqubic)
> library(Biobase)
> library(biclust)
> help("parseQubicRules")
> example(parseQubicRules)
```

3 Case study: a software pipeline to identify biclusters

Here we demonstrate the use of the *rqubic* by identifying biclusters from a microarray dataset.

First we build an object of the `ExpressionSet` class. If the given dataset is already an `ExpressionSet`, this step can be skipped. Although *rqubic* can also accepts `matrix` as input, here we use the *ExpressionSet* to demonstrate a general approach.

```
> library(rqubic)
> library(Biobase)
> library(biclust)
> data(BicatYeast)
> demo.exprs <- new("ExpressionSet", exprs=BicatYeast)
> ## processing the condition information
> demo.cond.split <- strsplit(sub("\\.CEL", "", colnames(BicatYeast)), "_")
> demo.group <- sapply(demo.cond.split, function(x) paste(x[-length(x)], collapse="_"))
```

```
> demo.time <- sapply(demo.cond.split, function(x) x[length(x)])
> pData(demo.exprs) <- data.frame(group=demo.group,
+                                 time=demo.time)
> sampleNames(demo.exprs) <- paste(demo.group, demo.time)
```

Once the `ExpressionSet` object is ready, we could discretize the dataset with `quantile.discretization` implemented in the *biclust* package. Discretization methods other than the default quantile approach might also be used.

By default, the rank is set to 1. Expression levels of each feature is classified into down, unchanged and up in samples.

```
> demo.disc <- quantileDiscretize(demo.exprs)
```

The discretized matrix is used to generate seeds. This step can be slow if there are many features (rows) in the matrix, for example, > 10000 in a human microarray experiment.

```
> demo.seed <- generateSeeds(demo.disc)
```

Finally, by the heuristic algorithm described before, the microarray expression matrix are bi-clustered.

```
> demo.bic <- quBicluster(demo.seed, demo.disc)
> demo.bic

An object of class QUBICBiclusterSet

Used features: 419
Used conditions: 70
Parameters: k=3, f=1, c=0.95, o=100, q=0.06, r=1
Call:
        blocksByIndex(seeds = seeds, eset = eset, index = rcIndex, report.no = report.no,
            tolerance = tolerance, filter.proportion = filter.proportion)

Number of Biclusters found:  87

First  5  Cluster sizes:
                   BC 1 BC 2 BC 3 BC 4 BC 5
Number of Rows:      72   53   37   80   56
Number of Columns:    4    5    7    3    4
```

As the code above shows, a summary of the *QUBICBiclusterSet* object, `demo.bic`, is given when the object name is given as the only command (namely calling the `print` function implicitly). The summary, extending from the method for the *biclust* objects, contains

important information about the biclusters found in the expression dataset. They include used features and conditions (note that there is no guarantee that all features/conditions will be within one or more biclusters), parameters used to identify biclusters, the function call, and the sizes of first five clusters.

To further explore the features and conditions that are within biclusters, one could use functions include `features`, `conditions`; to examine each bicluster, the corresponding functions will be `BCfeatures` and `BCconditions`, where *BC* stands for bicluster. Counting function, for example `featureCount` and `BCfeatureCount`. For more details on these functions, one could call on-line help pages by executing:

```
> help("features", package="rqubic")
```

Note that the third bicluster includes all samples of the group `cell\_cycle\_aph`. We show the parallel plot of this bicluster in figure **??** by the function implemented in the *biclust* package.

4 Prospects

Since the biclustering algorithm usually produces a large number of biclusters, one open question is how to determine which biclusters are the best or most informative. We are now working on a set of measures, and they will be implemented in the *rqubic* package once they are available.

5 Session Info

The vignette was produced within the following session:

- R version 3.3.0 RC (2016-04-26 r70550), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, utils
- Other packages: Biobase 2.32.0, BiocGenerics 0.18.0, MASS 7.3-45, biclust 1.2.0, colorspace 1.2-6, lattice 0.20-33, rqubic 1.18.0
- Loaded via a namespace (and not attached): flexclust 1.3-4, modeltools 0.2-21, stats4 3.3.0, tools 3.3.0

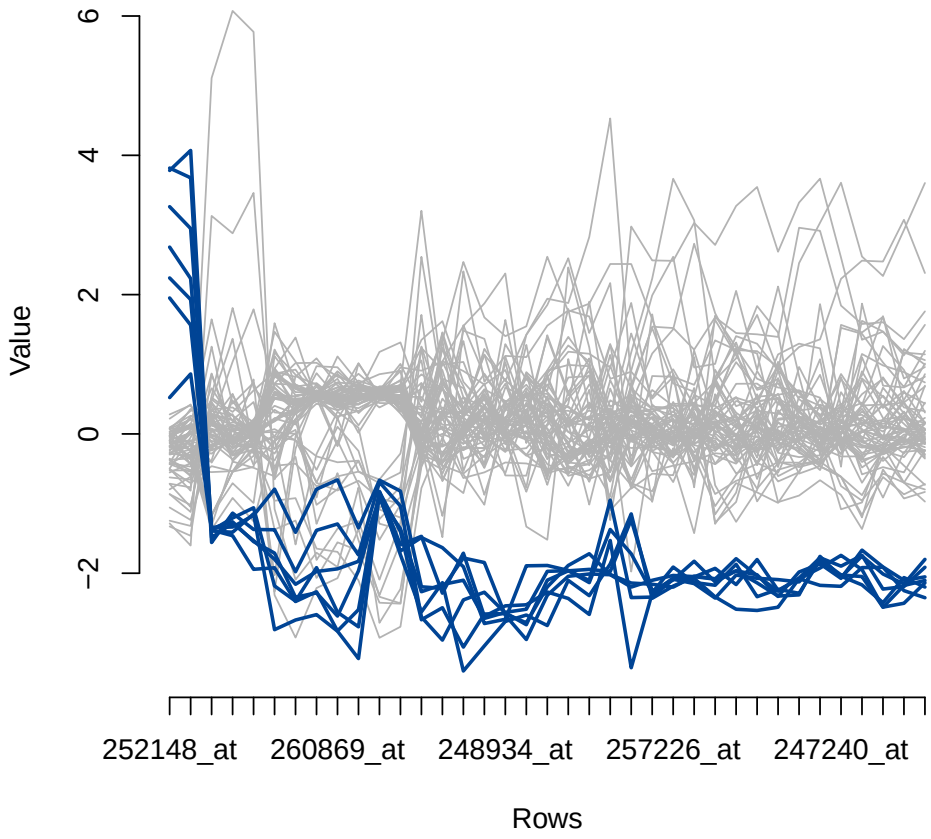

Figure 1: Parallel plot of one bicluster identified by rqubic. Each line represents the expression profile of one sample: blue lines indicate samples in the third bicluster, while the gray lines are rest of the samples. Each point of the line indicate the expression value, normalized by centering and scaling, of one feature in that bicluster. It is clear that for most of the genes identified in this bicluster, the expression levels in the cell cycle aph subset samples are lower. One of the next steps could be to perform gene set enrichment analysis to elucidate functions of these genes.