Bioinformatics, 31(11), 2015, 1851–1853
doi: 10.1093/bioinformatics/btv066
Advance Access Publication Date: 1 February 2015
Applications Note

OXFORD

Gene expression

ClassifyR: an R package for performance assessment of classification with applications to transcriptomics

Dario Strbenac[1,*], Graham J. Mann[2], John T. Ormerod[1] and Jean Y.H. Yang[1]

[1]School of Mathematics and Statistics, University of Sydney, New South Wales, Australia and [2]Melanoma Institute Australia, Sydney, New South Wales, Australia

*To whom correspondence should be addressed.

Associate Editor: Janet Kelso

Received on October 11, 2014; revised on January 7, 2015; accepted on January 27, 2015

Abstract

Although a large collection of classification software packages exist in R, a new generic framework for linking custom classification functions with classification performance measures is needed. A generic classification framework has been designed and implemented as an R package in an object oriented style. Its design places emphasis on parallel processing, reproducibility and extensibility. Finally, a comprehensive set of performance measures are available to ease post-processing. Taken together, these important characteristics enable rapid and reproducible benchmarking of alternative classifiers.

Availability and implementation: ClassifyR is implemented in R and can be obtained from the Bioconductor project: http://bioconductor.org/packages/release/bioc/html/ClassifyR.html

Contact: dario.strbenac@sydney.edu.au

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Classification is an important and frequently asked question in omics work. There are a large number of R packages providing a myriad of classification algorithms and tools. However, there are presently no packages which provide a generic framework for incorporating custom classification functions and cross validation (CV) estimation of performance measures. Most CV classification errors in existing packages are provided as specific functions associated with individual classification algorithms. These functions return the classification errors in differing formats. Hence, researchers need to write wrapper functions to process the results each time they use a different function created by a different author.

An effective classification package should exhibit the characteristics of fast computational time, reproducible results, flexibility to incorporate user-defined classification functions and post-processing tools for performance assessment. There are some R packages which provide a subset of these desirable features. For example, `SPRINT` (Mitchell *et al.*, 2014) provides a new framework for parallelizing classification algorithms in R, but requires new classification functions to be coded in C or Fortran. This is challenging for users who are only proficient in R. Many popular R packages such as `e1071` (Meyer *et al.*, 2014) and `PamR` (Hastie *et al.*, 2013) provide classifiers, but are not extensible. `caret` is a popular framework which provides parallelization and customisable classifiers, but almost no post-processing tools (Kuhn, 2008). Similar packages, such as `MLInterfaces`, `CMA` and `MCRestimate` are also available from Bioconductor, but are also not comprehensive classification solutions. An extensive list of classification packages is viewable on Comprehensive R Archive Network (CRAN) (Hothorn, 2014).

To address these issues, a new R package, `ClassifyR`, is introduced. It provides an implementation of a general framework for classification and provides simple ways to calculate multiple performance measures and explore the stability of feature selection, while being easily extensible. We will also illustrate an aspect of its

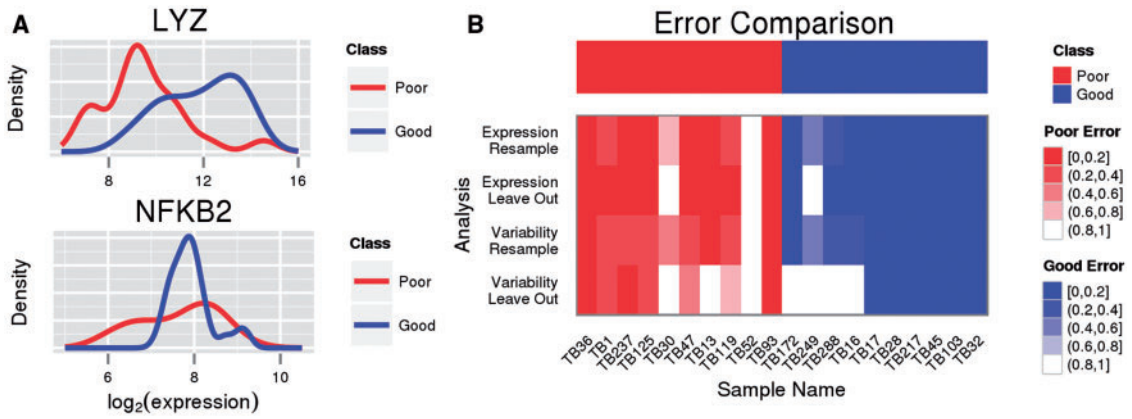

Fig. 1. Example output from post-processing functions. (**A**) Density plots produced by `plotFeatureClasses`. LYZ was most frequently selected for DE. NFKB2 was most often selected for DV. (**B**) Sample-wise error rates comparison for a subset of samples, produced by `errorMap` into the framework

flexible feature selection capability by demonstrating feature selection in both a differential expression (DE) setting, as well as differential variability (DV). The concept of DV was first introduced in Ho *et al.* (2008) and it selects genes with different variability between two classes of interest. This potentially selects different features to DE and offers a different perspective for identifying interesting biomarkers. The selection of DV features will be explored and classification performance examined for DV.

2 Implementation

2.1 Classification framework

The framework is implemented as four components. The data transformation stage takes a matrix of data and returns transformed matrix of data. The feature selection stage takes in a matrix of training data and returns a vector of indices indicating which features were selected. Training uses the training samples and the selected features to build a classifier. Finally, the prediction stage takes a matrix of test data and returns a vector of class predictions. The order that stages are executed in is user-defined.

Parameters are stored in classes. This allows type checking, and ensures the user provides all necessary variables for each stage of classification. The four stages each have an associated parameter class, with their own requirements (Supplementary Table S1). Full support is present for the Bioconductor class `ExpressionSet`, allowing popular benchmark datasets, such as CuratedOvarianData (Ganzfried *et al.*, 2013), to seamlessly be imported into the framework.

2.2 Selection and classification algorithms

Some interfaces to common R algorithms are provided, such as `limmaSelection` for feature selection by DE ranking. Other R functions work directly with the framework, with no need for an interface, such as `dlda` from package `sparsediscrim`. DV analysis is supported by a function which implements Levene feature selection and another that implements Fisher's LDA.

2.3 Result evaluation

Once predictions are obtained, the package allows easy exploration of results. Functions are available that calculate classification performance measures, sample-wise errors, feature inclusion frequency and a variety of plots to compare different performance aspects between analyses. No other package provides such an end-to-end classification solution.

3 Results and Discussion

We illustrate the package on a dataset of 47 patients with stage III melanoma that had their gene expression measured by microarrays (Jayawardana *et al.*, 2015). Patients are divided into two classes: *poor prognosis* ($n = 22$) and *good prognosis* ($n = 25$).

A comparison of classification using DE and DV, which has previously only been used in one DNA methylation study (Teschendorff and Widschwendter, 2012), is made. The provided feature selection functions `limmaSelection` and `leveneSelection` were used, respectively. Using the convenience function `calcError` to obtain balanced error rates, it was found that error rates were similar for both feature types (Supplementary Table S2). The function `distribution` was used to find out which features were selected most often (Supplementary Table S3). The top feature of each type was plotted using the function `plotFeatureClasses` (Fig. 1A). Limma feature selection was much more stable than Levene selection. Sample-wise error rates were plotted simultaneously using `errorMap` (Fig. 1B), showing similar error rates for DE and DV classification. The framework enables seamless use of multiple processors to speed up CV. Supplementary Table S4 shows the time taken to do 100 replications (using random permutations of the samples) of 5-fold CV and leave-2-out CV. When using 16 cores, the processing time is about eight times less than using a single core.

In summary, `ClassifyR` is a framework that allows both convenient and extensible classification and exploration of results. It takes advantage of the numerous cores in modern computers to significantly reduce time required for thorough CV.

Acknowledgements

We are thankful to Rebecca Barter for testing this software.

Funding

ARC Early Career Award DE130101670 (J.O.); ARC Future Fellowship (FT0991918) (J.Y.H.Y.); Australian Postgraduate Award (D.S.).

Conflict of Interest: none declared.

References

Ganzfried,B.F. *et al.* (2013) curatedOvarianData: clinically annotated data for the ovarian cancer transcriptome. *Database*, **2013**, bat013.

Hastie,T. *et al.* (2013) pamr: Pam: prediction analysis for microarrays.

Ho,J.W.K. *et al.* (2008) Differential variability analysis of gene expression and its application to human diseases. *Bioinformatics*, **24**, i390–i398.

Hothorn,T. (2014) CRAN Task View: Machine Learning & Statistical Learning.

Jayawardana,K. *et al.* (2015) Determination of prognosis in metastatic melanoma through integration of clinico-pathologic, mutation, mRNA, microRNA, and protein information. *Int. J. Cancer*, **136**, 863–874.

Kuhn,M. (2008) Building predictive models in R using the caret package. *J. Stat. Softw.*, **28**, 1–26.

Meyer,D. *et al.* (2014) e1071: Misc functions of the department of statistics, TU Wien.

Mitchell,L. *et al.* (2014) Parallel classification and feature selection in microarray data using SPRINT. *Concurr. Comput. Pract. Exp.*, **26**, 854–865.

Teschendorff,A.E. and Widschwendter,M. (2012) Differential variability improves the identification of cancer risk markers in DNA methylation studies profiling precursor cancer lesions. *Bioinformatics*, **28**, 1487–1494.