SOFTWARE TOOL ARTICLE

REVISED *exprso*: an R-package for the rapid implementation of machine learning algorithms [version 2; referees: 2 approved]

Thomas Quinn[1], Daniel Tylee[2], Stephen Glatt[2]

[1]Bioinformatics Core Research Facility, Deakin University, Victoria, Australia
[2]PsychGENe Lab, SUNY Upstate Medical University, Syracuse, USA

v2 **First published:** 27 Oct 2016, **5**:2588 (doi: 10.12688/f1000research.9893.1)
Latest published: 06 Dec 2017, **5**:2588 (doi: 10.12688/f1000research.9893.2)

Abstract

Machine learning plays a major role in many scientific investigations. However, non-expert programmers may struggle to implement the elaborate pipelines necessary to build highly accurate and generalizable models. We introduce *exprso*, a new R package that is an intuitive machine learning suite designed specifically for non-expert programmers. Built initially for the classification of high-dimensional data, *exprso* uses an object-oriented framework to encapsulate a number of common analytical methods into a series of interchangeable modules. This includes modules for feature selection, classification, high-throughput parameter grid-searching, elaborate cross-validation schemes (e.g., Monte Carlo and nested cross-validation), ensemble classification, and prediction. In addition, *exprso* also supports multi-class classification (through the 1-vs-all generalization of binary classifiers) and the prediction of continuous outcomes.

This article is included in the RPackage gateway.

This article is included in the Machine learning: life sciences collection.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	2
REVISED version 2 published 06 Dec 2017		✓ report
version 1 published 27 Oct 2016	✓ report	? report

1 **Dariusz Plewczynski**, University of Warsaw, Poland
Julian Zubek, University of Warsaw, Poland

2 **Henry Löffler-Wirth**, Interdisciplinary Centre for Bioinformatics, Germany

Discuss this article

Comments (0)

Corresponding author: Thomas Quinn (contacttomquinn@gmail.com)

Author roles: Quinn T: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Tylee D**: Writing – Original Draft Preparation, Writing – Review & Editing; **Glatt S**: Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

How to cite this article: Quinn T, Tylee D and Glatt S. ***exprso*: an R-package for the rapid implementation of machine learning algorithms [version 2; referees: 2 approved]** *F1000Research* 2017, **5**:2588 (doi: 10.12688/f1000research.9893.2)

Grant information: The author(s) declared that no grants were involved in supporting this work.

First published: 27 Oct 2016, **5**:2588 (doi: 10.12688/f1000research.9893.1)

REVISED **Amendments from Version 1**

The new version of this manuscript reflects changes made to the exprso package to improve the ease of data import (via the new exprso function) and the extension of all methods to handle the prediction of continuous outcomes, and revises Figure 1.

See referee reports

Introduction

Supervised machine learning has an increasingly important role in biological studies. However, the sheer complexity of machine learning pipelines poses a significant barrier to expert biologists unfamiliar with the intricacies of machine learning. Moreover, many biologists lack the time or technical skills necessary to establish their own pipelines. Here, we discuss the `exprso` package, a framework for the rapid implementation of high-throughput machine learning, tailored specifically for use with high-dimensional data. As such, this package aims to empower investigators to execute state-of-the-art binary and multi-class classification, as well as regression, with minimal programming experience necessary.

Although R offers a tremendous number of high-quality machine learning packages, there exists only a handful of fully integrated machine learning suites for R. Of these, we recognize here the `caret` package which offers an expansive toolkit for both classification and regression analyses[1]. Otherwise, we acknowledge the `RWeka` package which provides an API to the popular Weka machine learning suite, originally written in Java[2]. While these packages have a vast repertoire of functionality, we believe the `exprso` package has some advantages.

First, this package employs an object-oriented design that makes the software intuitive to lay programmers. In place of a few, elaborate functions that offer power at the expense of convenience, this package makes use of more, simpler functions whereby each constituent event has its own method that users can combine in tandem to create their own custom analytical pipeline. Second, this package contains single functions that execute elaborate high-throughput machine learning pipelines. These, coupled with special argument handlers, manage sophisticated pipelines such as high-throughput parameter grid-searching, Monte Carlo cross-validation[3], and nested cross-validation[4]. Moreover, users can embed these high-throughput modules (e.g., parameter grid-searching) within other modules (e.g., Monte Carlo cross-validation), allowing for infinite possibility. In addition, this package provides an automated way to build ensembles from the results of these high-throughput modules.

In addition, this package facilitates multi-class classification by generalizing binary classification methods to a multi-class context. Specifically, this package automatically executes 1-vs-all classification and prediction whenever working with a dataset that contains multiple class labels. Moreover, this package provides a specialized high-throughput module for 1-vs-all classification with individual 1-vs-all feature selection, an alternative to conventional multi-class classification that has been reported to improve results (at least in the setting of 1-vs-1 multi-class support vector machines)[5]. The `exprso` package also supports the prediction of continuous outcomes.

While we acknowledge that premier machine learning suites, like `caret`, may surpass our package in the breadth of their functionality, we do not intend to replace these tools. Rather, we developed `exprso` as an adjunct, or alternative, tailored specifically to those with limited programming experience, especially biologists working with high-dimensional data. That said, we hope that even some expert programmers find value in our software.

Methods

Implementation

This package uses an object-oriented framework for machine learning. In this paradigm, every unique task, such as data splitting (i.e., creating the training and validation sets), feature selection, and model construction, has its own associated function, called a method. These methods typically work as wrappers for other R functions, structured so that the objects returned by one method will feed seamlessly into the next method.

In other words, each method represents one of a number of analytical modules that provides the user with stackable and interchangeable data processing tools. Examples of these methods include wrappers for popular feature selection methods (e.g., analysis of variance (ANOVA), recursive feature elimination[6,7], empiric Bayes statistic[8], minimum redundancy maximum relevancy (mRMR)[9], and more) as well as numerous models (e.g., support vector machines (SVM)[10], neural networks[11], deep neural networks[12], random forests[13], and more).

We have adopted a nomenclature to help organize the methods available in this package. In this scheme, most functions have a few letters in the beginning of their name to designate their general utility. Below, we include a brief description of these function prefixes along with a flow diagram of the available methods.

- **array:** Modules that import data stored as a `data.frame`, `ExpressionSet` object, or local text file. Alternatively, the `exprso` function imports data in *x*, *y* format (recommended for most users).
- **mod:** Modules that modify the imported data prior to building models.
- **split:** Modules that split these data into training and validation (or test) sets.
- **fs:** Modules that perform feature selection.
- **build:** Modules that build models and ensembles.
- **predict:** Modules that deploy models and ensembles.
- **calc:** Modules that calculate model performance, including area under the receiver operating characteristic (ROC) curve (AUC).

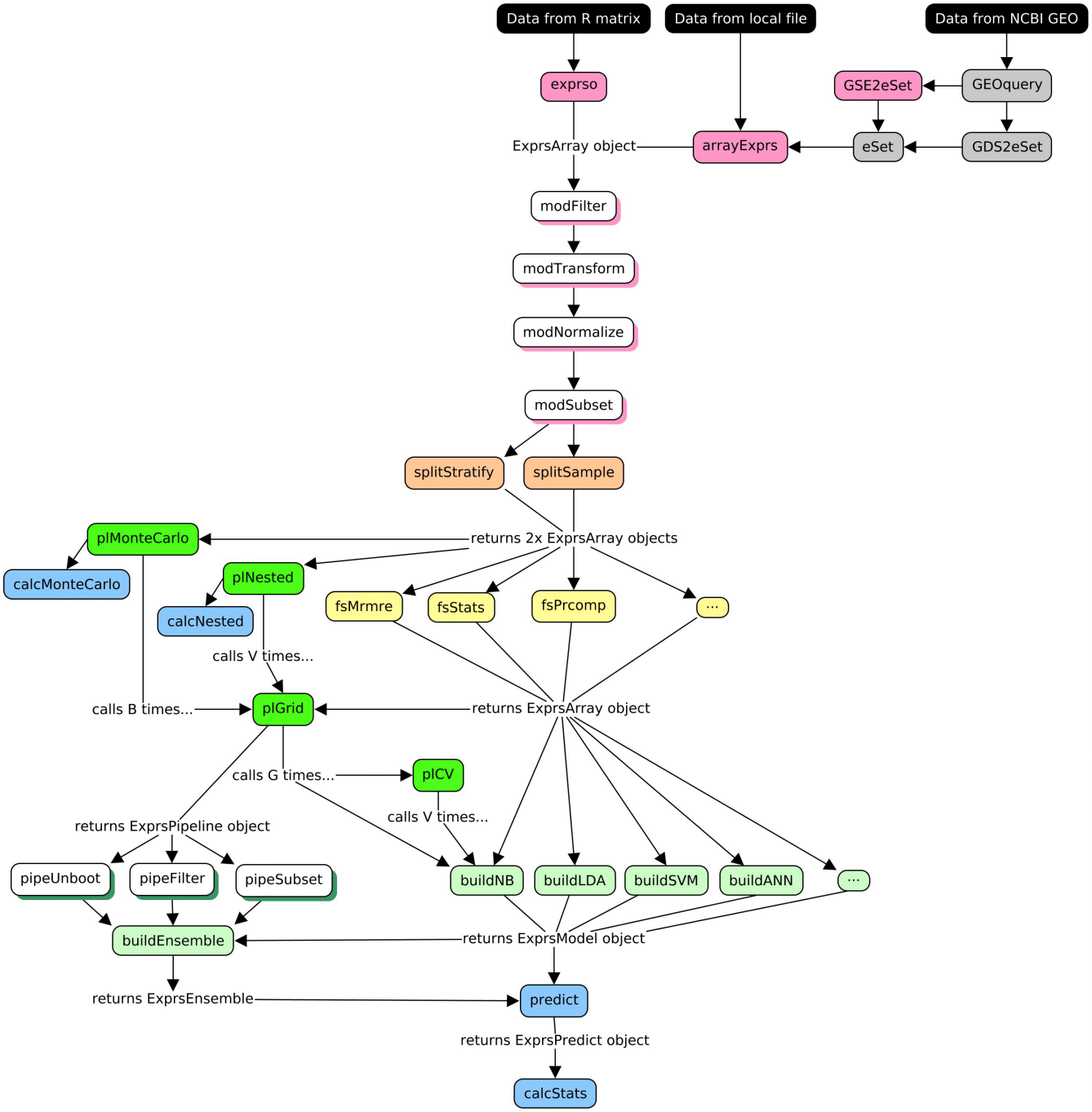

Figure 1. A directed graph of all modules included in the `exprso` package and how they might relate to each other in practice. Elements colored grey exist outside of this package and instead refer to natively compatible components from the `GEOquery`[14] and `Biobase`[15] packages. Elements colored black indicate possible data sources.

- **pl:** Modules that manage pipelines, including high-throughput parameter grid-searches, Monte Carlo cross-validation, and nested cross-validation.
- **pipe:** Modules that filter the pipeline results.

We refer the reader to the package vignette, "An Introduction to the `exprso` Package," hosted with the package on the Comprehensive R Archive Network (CRAN), for a detailed description of the functions available from this package[16].

Operation

Specific computer hardware requirements will depend on the size of the dataset under study and the functions used. For the most part, however, a standard laptop computer with the latest version of R installed will handle most applications of the `exprso` package.

Use cases

To showcase this package, we make use of the publicly available hallmark Golub 1999 dataset to differentiate (i.e., classify) acute lymphocytic leukemia (ALL) from acute myelogenous leukemia (AML) based on gene expression as measured by microarray technology[17]. We begin by importing this dataset as an `ExpressionSet` object from the package `GolubEsets` (version 1.16.0)[18]. Then, using the `arrayExprs` function, we load the `ExpressionSet` object into `exprso`. Note that, alternatively, one could use the `exprso` function to import the data in *x*, *y* format (recommended for most users).

Next, using the `modFilter`, `modTransform`, and `modNormalize` methods, we threshold filter, log2 transform, and standardize the data, respectively, reproducing the pre-processing steps taken by the original investigators[19]. To keep the code clear and concise, we make use of the %>% function notation from the `magrittr` package[20]. In short, this function passes the result from the previous function call to the first argument of the next function, known as piping.

```
library(exprso)
library(golubEsets)
library(magrittr)
data(Golub_Merge)
array <-
  arrayExprs(Golub_Merge,
             colBy = "ALL.AML",
             include = list("ALL","AML"))%>%
  modFilter(20, 16000, 500, 5) %>%
  modTransform %>%
  modNormalize
```

Then, using the `splitSample` method, one of the `split` methods shown in the above diagram, we partition the data into a training and a test set through random sampling without replacement. Next, we perform a series of feature selection methods on the extracted training set. Through the `fs` modules `fsStats` and `fsPrcomp`, we pass the top 50 features as selected by the Student's t-test through dimension reduction by principal components analysis (PCA).

```
splitSets <- splitSample(array, percent.include = 67)
array.fs <-
  trainingSet(splitSets) %>%
  fsStats(how = "t.test") %>%
  fsPrcomp(top = 50)
```

With feature selection complete, we can build the classifier model. For this example, we use the `buildSVM` method to train a linear kernel support vector machine (SVM) (with default parameters) using the top 5 principal components. Then, we deploy the trained machine on the test set from above. Note that, by design, each feature selection event, including the rules for dimension reduction by PCA, gets passed along automatically at every step up until model deployment. This ensures that the test set always undergoes the same feature selection history as the training set. The `calcStats` function allows us to calculate classifier performance as sensitivity, specificity, accuracy, or area under the curve (AUC)[21,22].

```
pred <-
  array.fs %>%
  buildSVM(top = 5, kernel = "linear") %>%
  predict(testSet(splitSets))
calcStats(pred)
```

When constructing a model using a `build` module, we can only specify one set of parameters at a time. However, investigators often want to test models across a vast range of parameters. For this reason, we provide methods like `plGrid` to automate high-throughput parameter grid-searches. These methods not only wrap model construction, but also model deployment. In addition, they accept a `fold` argument to toggle leave-one-out or v-fold cross-validation.

Below, we show a simple example of parameter grid-searching, whereby the top 3, 5, and 10 principal components, as established above, get used to construct linear and radial kernel SVMs with costs of 1, 101, and 1001. In addition, we calculate a (biased) 10-fold cross-validation accuracy to help guide our choice of the final model parameters. (Note that we call this accuracy biased because we are performing cross-validation on a dataset that has already undergone feature selection. Although this approach gives a poor assessment of absolute classifier performance[23], it may still have value in helping to guide parameter selection in a statistically valid manner. As an alternative to this biased cross-validation accuracy, users can instead call the `plNested` method in which feature selection is performed anew with each data split that occurs during the leave-one-out or v-fold cross-validation.)

```
pl <-
  plGrid(array.fs, testSet(splitSets),
         top = c(3, 5, 10),
         how = "buildSVM",
         kernel = c("linear", "radial"),
         cost = c(1, 101, 1001),
         fold = 10)
```

Finally, we show an example for the `plMonteCarlo` method, an implementation of Monte Carlo cross-validation. Compared to the `plGrid` method which iteratively builds and deploys models on a validation (or test) set, `plMonteCarlo` wraps multiple iterations of data splitting, feature selection, and parameter grid-searching. The final result therefore contains a summary of the model performances as measured across any number of bootstraps carved out from the initial dataset. Argument handler

functions help organize the arguments supplied to the splitting, feature selection, and high-throughput methods of the `plMonteCarlo` method call. (Note that when using the Monte Carlo cross-validation method (or any of the other `pl` modules), the user may iterate over any `build` method provided by `exprso`, not only `buildSVM`. This includes the `buildDNN` method for deep neural networks as implemented via `h2o`[12]. Also note that the user can embed other cross-validation methods, such as another Monte Carlo or nested method, within the cross-validation method call, allowing for endless combinatory possibility.)

In the first section of the code below, we define the argument handler functions for the `plMonteCarlo` call. As suggested by their names, the `ctrlSplitSet`, `ctrlFeatureSelect`, and `ctrlGridSearch` handlers manage arguments to data splitting, feature selection, and high-throughput grid-searching, respectively. In this example, we set up arguments to split the unaltered training set through random sampling with replacement, perform the two-step feature selection process from above, and run a high-throughput parameter grid-search with biased cross-validation. The unaltered dataset is processed this way 10 times, as directed by argument `B`.

```
ss <-
  ctrlSplitSet(func = "splitSample",
               percent.include = 67,
               replace = TRUE)
fs <-
  list(ctrlFeatureSelect(func = "fsStats",
                         how = "t.test"),
       ctrlFeatureSelect(func = "fsPrcomp",
                         top = 50))
gs <-
  ctrlGridSearch(func = "plGrid",
                 top = c(3, 5, 10),
                 how = "buildSVM",
                 kernel = c("linear", "radial"),
                 cost = c(1, 101, 1001),
                 fold = 10)
boot <-
  plMonteCarlo(trainingSet(splitSets),
               B = 10,
               ctrlSS = ss,
               ctrlFS = fs,
               ctrlGS = gs)
```

Optionally, one can use these results to build an ensemble of the best models from each bootstrap, then deploy that censemble on the withheld test set. Analogous to how random forests will deploy an ensemble of decision trees[24], this method, which we dub "random plains", will deploy an ensemble of SVMs.

```
ens <- buildEnsemble(boot, colBy = "valid. acc", top = 1)
pred <- predict(ens, testSet(splitSets))
```

The above procedure, including building and deploying ensembles, also works for multi-class classification and continuous outcome prediction. We refer the reader to the package vignettes, "An Introduction to the `exprso` Package" and "Advanced Topics for the `exprso` Package", both hosted with the package on the Comprehensive R Archive Network (CRAN), for a detailed description of all methods included in this package[16].

Summary

Here we introduce the R package `exprso`, a machine learning suite tailored specifically to working with high-dimensional data. Unlike other machine learning suites, we have prioritized simplicity of use over expansiveness. As such, `exprso` provides a fully interchangeable and modular programming interface that allows for the rapid implementation of classification and regression pipelines. We have included in this framework functions for executing some of most popular feature selection methods and machine learning algorithms. In addition, `exprso` also contains a number of modules that perform high-throughput parameter grid-searching in conjunction with sophisticated cross-validation schemes. Owing to its ease-of-use and extensive documentation, we hope `exprso` will serve as a helpful resource, especially to scientific investigators with limited prior programming experience.

Software availability

Software available from: http://cran.r-project.org/web/packages/exprso/

Latest source code: http://github.com/tpq/exprso

Archived source code as at time of publication: http://dx.doi.org/10.5281/zenodo.1069113[25]

Software license: GNU General Public License, version 2

Author contributions

TQ designed and implemented the tool, applied the tool to the use case, and drafted the article. DT and SG helped design the tool and drafted the article. DT contributed code and performed extensive beta testing.

Competing interests

No competing interests were disclosed.

Grant information

The author(s) declared that no grants were involved in supporting this work.

References

1. Kuhn M: **Building predictive models in r using the caret package.** *J Stat Softw.* ISSN 1548-7660. 2008; **28**(1): 1–26.
Publisher Full Text
2. Hornik K, Buchta C, Zeileis A: **Open-source machine learning: R meets Weka.** *Computation Stat.* ISSN 0943-4062, 1613-9658. 2008; **24**(2): 225–232.
Publisher Full Text
3. Picard RR, Cook RD: **Cross-Validation of Regression Models.** *J Am Stat Assoc.* ISSN 0162-1459, 1537-274X. 1984; **79**(387): 575–583.
Publisher Full Text
4. Varma S, Simon R: **Bias in error estimation when using cross-validation for model selection.** *BMC Bioinformatics.* ISSN 1471-2105. 2006; **7**: 91.
PubMed Abstract | Publisher Full Text | Free Full Text
5. Huang PX, Fisher RB: **Individual feature selection in each One-versus-One classifier improves multi-class SVM performance**.
Reference Source
6. Guyon I, Weston J, Barnhill S, *et al.*: **Gene Selection for Cancer Classification using Support Vector Machines.** *Mach Learn.* ISSN 0885-6125, 1573-0565. 2002; **46**(1–3): 389–422.
Publisher Full Text
7. Johannes M: **pathClass: Classification using biological pathways as prior knowledge.** R package version 0.9.4. 2013.
Reference Source
8. Ritchie ME, Phipson B, Wu D, *et al.*: ***limma* powers differential expression analyses for RNA-sequencing and microarray studies.** *Nucleic Acids Res.* 2015; **43**(7): e47.
PubMed Abstract | Publisher Full Text | Free Full Text
9. De Jay N, Papillon-Cavanagh S, Olsen C, *et al.*: **mRMRe: an R package for parallelized mRMR ensemble feature selection.** *Bioinformatics.* ISSN 1367-4803, 1460-2059. 2013; **29**(18): 2365–2368.
PubMed Abstract | Publisher Full Text
10. Meyer D, Dimitriadou E, Hornik K, *et al.*: **e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071). TU Wien.** R package version 1.6-7. 2015.
Reference Source
11. Venables WN, Ripley BD: **Modern Applied Statistics with S.** Springer, New York, fourth edition. ISBN 0-387-95457-0. 2002.
Publisher Full Text
12. Aiello S, Kraljevic T, Maj P, *et al.*: **h2o: R Interface for H2O.** R package version 3.10.0.6. 2016.
Reference Source
13. Liaw A, Wiener M: **Classification and regression by randomforest.** *R News.* 2002; **2**(3): 18–22.
Reference Source
14. Davis S, Meltzer PS: **GEOquery: a bridge between the Gene Expression Omnibus (GEO) and Bioconductor.** *Bioinformatics.* 2007; **23**(14): 1846–1847.
PubMed Abstract | Publisher Full Text
15. Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor.** *Nat Methods.* 2015; **12**(2): 115–121.
PubMed Abstract | Publisher Full Text | Free Full Text
16. Quinn T: **exprso: Rapid Implementation of Machine Learning Algorithms for Genomic Data.** R package version 0.1.7. 2016.
Reference Source
17. Golub TR, Slonim DK, Tamayo P, *et al.*: **Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.** *Science.* ISSN 0036-8075. 1999; **286**(5439): 531–537.
PubMed Abstract | Publisher Full Text
18. Golub T: **golubEsets: exprSets for golub leukemia data.** R package version 1.12.0.
Reference Source
19. Deb K, Raji Reddy A: **Reliable classification of two-class cancer data using evolutionary algorithms.** *Biosystems.* ISSN 0303-2647. 2003; **72**(1–2): 111–129.
PubMed Abstract | Publisher Full Text
20. Bache SM, Wickham H: **magrittr: A Forward-Pipe Operator for R.** R package version 1.5. 2014.
Reference Source
21. Bradley AP: **The use of the area under the ROC curve in the evaluation of machine learning algorithms.** *Pattern Recognit.* ISSN 0031-3203. 1997; **30**(7): 1145–1159.
Publisher Full Text
22. Sing T, Sander O, Beerenwinkel N, *et al.*: **ROCR: visualizing classifier performance in R.** *Bioinformatics.* 2005; **21**(20): 3940–1.
PubMed Abstract | Publisher Full Text
23. Simon R, Radmacher MD, Dobbin K, *et al.*: **Pitfalls in the use of DNA microarray data for diagnostic and prognostic classification.** *J Natl Cancer Inst.* ISSN 0027-8874, 1460-2105. 2003; **95**(1): 14–18.
PubMed Abstract | Publisher Full Text
24. Breiman L: **Random Forests.** *Mach Learn.* ISSN 0885-6125, 1573-0565. 2001; **45**(1): 5–32.
Publisher Full Text
25. Quinn T: **tpq/exprso: exprso-0.2.6 (Version exprso-0.2.6).** *Zenodo.* 2017.
Publisher Full Text

Open Peer Review

Current Referee Status: ✔ ✔

Version 2

Referee Report 28 February 2018

doi:10.5256/f1000research.14476.r28684

✔ **Henry Löffler-Wirth**
Leipzig University, Interdisciplinary Centre for Bioinformatics, Leipzig, 04107, Germany

The authors addressed my comments in their new manuscript and software versions.
In particular, the program seems more convenient in its current state, and it is also under continuous development.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Referee Report 17 February 2017

doi:10.5256/f1000research.10664.r20319

? **Henry Löffler-Wirth**
Leipzig University, Interdisciplinary Centre for Bioinformatics, Leipzig, 04107, Germany

The authors present a new tool integrating established algorithms into one R package. This tool is intended to provide access to state-of-the-art statistical analysis to users with limited programming experience. As the manuscript is rather a package vignette than an independent article, I focus my review mainly on the software and usability.

In general, the intention of providing comprehensive analysis options for non-experts is desirable, however a bit overachieving. In particular, I have several years of programming experience in R, but I was not able to apply the presented methods to another (multiclass) data with acceptable effort of time. Non-expert programmers, as biologists or doctors, also will not be able to correctly use the software in its present form.

I therefore recommend extensive improvement of usability, user guidance and error feedback. I also acknowledge how challenging implementation of such tools is and encourage the authors to continue development of “exprso”.

Particular comments:

- The *arrayExps*-function should accept also standard matrix containing expression values, the groups may then be given as character vector.
- Handling of most functions can be improved by providing standard parameter values, e.g:

1. in *arrayExps* (may simply include all groups)
2. *percent.include* in *splitSampe* (66%)
3. and so on

- Help the users to find errors in their function calls by giving appropriate warning/error messages. E.g. “fsStats cannot be applied in multi-class studies”. No non-programmer will know what an “inherited method” is.
- Addressing the same point: *fsANOVA* returned with an error not clear to me (my data includes 4 groups):

"contrasts can be applied only to factors with 2 or more levels"

- I found no information about this in the help file of the fs-methods. In general, documentation should be split into one individual file for each function.
- In the manuscript and the package vignette, the authors provide an overview figure of all functions implemented in the package. I suggest revision of this figure: a more systematic flowchart layout of the functions graph would help to find the starting point (GEO, local file, R matrix) and to follow the analysis workflow. Top-left part of the figure is not clear to me.

Competing Interests: No competing interests were disclosed.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 07 Dec 2017

Thomas Quinn, Deakin University, Australia

Dear Henry,

Thank you so much for taking the time to perform a detailed and critical review of the software. I regret that professional obligations have delayed me from addressing your feedback in a timely manner. However, I am pleased to say that I have heavily revised the exprso package, incorporating most, if not all, of your suggestions.

Key changes include:
* Easier data import with the `exprso` function which imports data in x, y format
* Software now supports continuous outcome prediction (when importing data via `exprso`)
* More default values (e.g., split modules) to simplify user experience
* Created custom errors for every split, feature selection, build function. Most other functions now have custom errors as well
* Documentation is split up by unique function. ?mod, ?split, ?fs, ?build, etc. all open a table of contents that overviews the unique functions available
* Figure 1 simplified with data sources clearly labeled in black
* Manuscript adjusted to reflect other changes

Also, you should not encounter any error with mutli-class classification when importing data using the new `exprso` function.

(PS: I am aware of a trivial warning in this version that is triggered by predict. It is already fixed in the developmental branch on GitHub).

Cheers,
Thom

Competing Interests: No competing interests were disclosed.

Referee Report 09 December 2016

doi:10.5256/f1000research.10664.r18380

Dariusz Plewczynski [1], **Julian Zubek** [2]
[1] Centre of New Technologies, University of Warsaw, Warsaw, Poland
[2] Center of New Technologies, University of Warsaw, Warsaw, Poland

This is a short software tool article presenting a new R package for implementing machine learning pipelines. According to the authors it is targeted specifically at non-expert programmers analyzing high-dimensional biological data. The article briefly describes design goals and implementation details of the package, and then provides an example of building full machine learning pipeline for well-known microarray data set.

The main contribution of this work lies in the prepared software and not in the accompanying manuscript. Because of this in the article there are no clear hypotheses nor conclusions. The software package does not provide any novel algorithms or functionalities. It is conceived as a wrapper which should make existing methods more accessible. This goal is of practical rather than scientific nature. Ultimately, the usefulness of the package can only be confirmed by its wider adoption by the users. This situation makes it hard to write a conclusive review.

I agree with the motivation behind this software. R package infrastructure can be indeed confusing and it is notoriously hard to navigate through it for a beginner programmer. I find the interface adopted by exprso package relatively clean and unambiguous. The way the evaluation methods are implemented is consistent with best machine learning practices. Eventual success of this package depends on how well it will be integrated with other popular packages. I hope that the authors will have the resources for further development of exprso.

Below I present more detailed comments for the authors:

- caret is mentioned as the R package with similar goals to exprso. I would find a more detailed comparison between these two package useful, both in terms of general design and specific use cases.

- It is commendable when newly created packages integrate with existing infrastructure. As I understand, exprso has some limited integration with GEOquery and Biobase packages allowing easy data import. However, after data is loaded all operations use special ExprsArray objects,

distinct from native R types such as DataFrame or Matrix. I understand this design choice, but I have to note that it limits interoperability. Should the additional processing in the middle of exprso pipeline be required, data needs to be converted between formats.

- I find the design in which transformations applied on the training set are automatically applied on the testing set controversial. It obscures the pipeline and may not be intuitive for beginner programmers. Moreover, sometimes transformations of the testing set differ slightly from the transformations of the training set.

- It is not obvious to me why simple train-test split is implemented as "split" module and cross-validation as "pl" module. There are not very much different. The way cross-validation is currently implemented does not allow detailed control over individual folds, which is sometimes useful.

Competing Interests: No competing interests were disclosed.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.