

## Open-Source Tools for Data Mining

Blaz Zupan, PhD<sup>a,b,\*</sup>, Janez Demsar, PhD<sup>a</sup>

<sup>a</sup>University of Ljubljana, Trzaska 25, SI-1000 Ljubljana, Slovenia

<sup>b</sup>Department of Molecular and Human Genetics, Baylor College of Medicine,  
Houston, TX, USA

The history of software packages for data mining is short but eventful. Although the term data mining was coined in the mid-1990s [1], statistics, machine learning, data visualization, and knowledge engineering—research fields that contribute their methods to data mining—were at that time already well developed and used for data exploration and model inference. Obviously, software packages were in use that supported various data mining tasks. But compared with the data mining suites of today, they were awkward, most often providing only command-line interfaces and at best offering some integration with other packages through shell scripting, pipelining, and file interchange. For an expert physician, the user interfaces of early data mining programs were as cryptic as the end of the last sentence. It took several decades and substantial progress in software engineering and user interface paradigms to create modern data mining suites, which offer simplicity in deployment, integration of excellent visualization tools for exploratory data mining, and—for those with some programming background—the flexibility of crafting new ways to analyze the data and adapting algorithms fit to the particular needs of the problem at hand.

Within data mining, there is a group of tools that have been developed by a research community and data analysis enthusiasts; they are offered free of charge using one of the existing open-source licenses. An open-source development model usually means that the tool is a result of a community effort, not necessary supported by a single institution but instead the result of contributions from an international and informal development team. This development style offers a means of incorporating the diverse experiences

---

This work was supported by a Program Grant P2-0209 and a Project Grants J2-9699 and V2-0221 from the Slovenian Research Agency and NIH/NICHD Program Project Grant P01 HD39691.

\* Corresponding author. University of Ljubljana, Trzaska 25, SI-1000 Ljubljana, Slovenia.

*E-mail address:* [blaz.zupan@fri.uni-lj.si](mailto:blaz.zupan@fri.uni-lj.si) (B. Zupan).

and views of multiple developers into a single platform. Open-source data mining suites may not be as stable and visually “finished” as their commercial counterparts but instead may offer high usefulness through alternative, exciting, and cutting-edge interfaces and prototype implementations of the most recent techniques. Being open source they are by definition extendable, and may offer substantial flexibility in handling various types of data.

In the following, we provide a brief overview of the evolution of approaches used in the development of data mining suites, with a particular focus on their user interfaces. We then elaborate on potential advantages of open-source data mining suites as compared with their commercial counterparts, and provide a wish list for techniques any data mining suite should offer to the biomedical data analyst. For our review of tools, we select several representative data mining suites available in open source, present each briefly, and conclude with a summary of their similarities and differences.

### **Evolution of open-source data mining tools**

Early model inference and machine learning programs from the 1980s were most often invoked from a command prompt (eg, from a UNIX or DOS shell), with the user providing the name of the input data file and any parameters for the inference algorithm. A popular classification tree induction algorithm called C4.5 [2] came with such an implementation. C4.5 (the source code is available at <http://www.rulequest.com/Personal>) could also accept a separate input file with cases for testing the model, but included no procedures for sampling-based evaluation of the algorithm. Implementations of early rule-based learning algorithms, such as AQ [3] and CN2 [4], were similar to C4.5 in this respect. Much of the experimental verification of these programs was performed on data sets from medicine, including those related to cancer diagnosis and prediction (see UCI Machine Learning Repository [5]). Such evaluations most often demonstrated that the classification rules inferred had some meaningful medical interpretation and performed well in classification accuracy on a separate test set.

Inference of models from medical data requires elaborate testing, which was not fully integrated into early data mining programs. Researchers typically used a scripting language, such as Perl [6], to separately implement sampling procedures and then execute programs for model inference and testing. To compare different algorithms, such scripts needed to reformat the data for each algorithm, parse textual outputs from each model, and use them to compute the corresponding performance scores. Needless to say, the implementation of such schemata required a substantial amount of programming and text processing.

As an alternative, several research groups started to develop suites of programs that shared data formats and provided tools for evaluation and reporting. An early example of such an implementation is MLC++ [7], a machine learning library in C++ with a command line interface that featured

several then-standard data analysis techniques from machine learning. MLC++ was also designed as an object-oriented library, extendible through algorithms written by a user who could reuse parts of the library as desired.

Command line interfaces, limited interaction with the data analysis environment, and textual output of inferred models and their performance scores were not things a physician or medical researcher would get too excited about. To be optimally useful for researchers, data mining programs needed to provide built-in data visualization and the ability to easily interact with the program. With the evolution of graphical user interfaces and operating systems that supported them, data mining programs started to incorporate these features. MLC++, for instance, was acquired by Silicon Graphics in mid 1990s, and turned into MineSet [8], at that time the most sophisticated data mining environment with many interesting data and model visualizations. MineSet implemented an interface whereby the data analysis schema was in a way predefined: the user could change the parameters of analysis methods, but not the composition of the complete analysis pathway. Clementine (<http://www.spss.com/clementine>), another popular commercial data mining suite, pioneered user control over the analysis pathway by embedding various data mining tasks within separate components that were placed in the analysis schema and then linked with each other to construct a particular analysis pathway. Several modern open-source data mining tools use a similar visual programming approach that, because it is flexible and simple to use, may be particularly appealing to data analysts and users with backgrounds other than computer science.

Flexibility and extensibility in analysis software arise from being able to use existing code to develop or extend one's own algorithms. For example, Weka (<http://www.cs.waikato.ac.nz/ml/weka/>) [9], a popular data mining suite, offers a library of well-documented Java-based functions and classes that can be easily extended, provided sufficient knowledge of Weka's architecture and Java programming. A somewhat different approach has been taken by other packages, including R (<http://www.r-project.org>), which is one of the most widely known open-source statistical and data mining suites. Instead of extending R with functions in C (the language of its core) R also implements its own scripting language with an interface to its functions in C. Most extensions of R are then implemented as scripts, requiring no source-code compilation or use of a special development environment. Recently, with advances in the design and performance of general-purpose scripting languages and their growing popularity, several data mining tools have incorporated these languages. The particular benefit of integration with a scripting language is the speed (all computationally intensive routines are still implemented in some fast low-level programming language and are callable from the scripting language), flexibility (scripts may integrate functions from the core suite and functions from the scripting language's native library), and extensibility that goes beyond the sole use of the data mining suites through use of other packages that interface with that

particular scripting language. Although harder to learn and use for novices and those with little expertise in computer science or math than systems driven completely by graphical user interfaces, scripting in data mining environments is essential for fast prototyping and development of new techniques and is a key to the success of packages like R.

### **Why mine medical data with open-source tools?**

Compared with off-the-shelf commercial data mining suites, open-source tools may have several disadvantages. They are developed mostly by research communities that often incorporate their most recent data analysis algorithms, resulting in software that may not be completely stable. Commercial data mining tools are often closely integrated with a commercial database management system, usually offered by the same vendor. Open-source data mining suites instead come with plug-ins that allow the user to query for the data from standard databases, but integration with these may require more effort than a single-vendor system.

These and other potential shortcomings are offset by several advantages offered by open-source data mining tools. First, open-source data mining suites are free. They may incorporate new, experimental techniques, including some in prototype form, and may address emerging problems sooner than commercial software. This feature is particularly important in biomedicine, with the recent emergence of many genome-scale data sets and new data and knowledge bases that could be integrated within analysis schemata. Provided that a large and diverse community is working with a tool, the set of techniques it may offer can be large and thus may address a wide range of problems. Research-oriented biomedical groups find substantial usefulness in the extendibility of the open-source data mining suites, the availability of direct access to code and components, and the ability to cross-link the software with various other data analysis programs. Modern scripting languages are particularly strong in supporting this type of ad hoc integration. Documentation for open-source software may not be as polished as that for commercial packages, but it is available in many forms and often includes additional tutorials and use cases written by enthusiasts outside the core development team. Finally, there is user support, which is different for open-source than for commercial packages. Users of commercial packages depend on the company's user support department, whereas users of open-source suites are, as a matter of principle, usually eager to help each other. This cooperation is especially true for open-source packages with large and active established user bases. Such communities communicate by online forums, mailing lists, and bug tracking systems to provide encouragement and feedback to developers, propose and prioritize improvements, report on bugs and errors, and support new users.

As these open-source tools incorporate advances in user interfaces and reporting tools, implement the latest analysis methods, and grow their user

bases, they are becoming useful alternatives and complements to commercial tools in medical data mining.

### **Open-source data mining toolbox—a wish list**

To support medical data mining and exploratory analysis, a modern data mining suite should provide an easy-to-use interface for physicians and biomedical researchers that is well supported with data and model visualizations, offers data analysis tools to accommodate interactive search for any interesting data patterns, and allows interactive exploration of inferred models [10–12]. In addition to being simple, the tools have to be flexible, allowing the users to define their own schemata for data analysis. Modern open-source data mining suites are almost by definition extendible; although this may not be a major concern of the users, it is important for data analysts and programmers in biomedical research groups who may need to develop custom-designed data analysis components and schemata.

Most open-source data mining tools today come as comprehensive, integrated suites featuring a wide range of data analysis components. In our opinion, the following set of tools and techniques should be on the wish list of any biomedical data analyst:

- A set of basic statistical tools for primary inspection of the data
- Various data visualization techniques, such as histograms, scatterplots, distribution charts, parallel coordinate visualizations, mosaic and sieve diagrams, and so forth
- Standard components for data preprocessing that include querying from databases, case selection, feature ranking and subset selection, and feature discretization
- A set of techniques for unsupervised data analysis, such as principal component analysis, various clustering techniques, inference of association rules, and subgroup mining techniques
- A set of techniques for supervised data analysis, such as classification rules and trees, support vector machines, naïve Bayesian classifiers, discriminant analysis, and so forth
- A toolbox for model evaluation and scoring (classification accuracy, sensitivity, specificity, Brier score, and other), that also includes graphical analysis of results, such as receiver-operating characteristic curves and lift chart analysis
- Visualizations of inferred models developed from either supervised or unsupervised analysis
- An exploratory data analysis environment, wherein the user can select a set of cases, features, or components of the model and explore the selection in a subsequent data or model visualization component. The emphasis here is on the interplay between data visualization and interaction.

- Techniques for saving the model in some standard format (such as PMML, <http://www.dmg.org/>) for its later use in systems for decision support outside the data mining suite with which the model was constructed
- Reporting, that is, implementation of a notebook-style tool in which the user can save the present results of analysis and associated reports, include any comments, and later retrieve the corresponding analysis schema for further exploration

We use the above list implicitly when reviewing the open-source data mining suites below and when summarizing our impressions of them at the end of the article.

### Selected open-source data mining suites

Below we review several open-source data mining suites, including some of the largest and most popular packages, such as Weka and R. Although representative of different types of user interfaces and implementations, the list is restrictive because the number of other data mining suites in open source is large; because of space limitations we necessarily selected only a small sample. We direct interested readers to web pages, such as KDnuggets (<http://www.kdnuggets.com/>) and Open Directory (<http://dmoz.org>), for more comprehensive lists of open-source data mining tools. For illustrations throughout this article we used a data set on heart disease from UCI Machine Learning Repository [5].

### R

R (<http://www.r-project.org>) is a language and environment for statistical computing and graphics. Most of its computationally intensive methods are efficiently implemented in C, C++, and Fortran, and then interfaced to R, a scripting language similar to the S language originally developed at Bell Laboratories [13]. R includes an extensive variety of techniques for statistical testing, predictive modeling, and data visualization, and has become a de facto standard open-source library for statistics (Fig. 1). R can be extended by hundreds of additional packages available at The Comprehensive R Archive Network (<http://cran.r-project.org>) that cover virtually every aspect of statistical data analysis and machine learning. For those interested in genomic data analysis in bioinformatics, there is an R library and software development project called Bioconductor (<http://www.bioconductor.org>).

The preferred interface to R is its command line and use through scripting. Scripting interfaces have distinct advantages: the data analysis procedure is stated clearly and can be saved for the later reuse. The downside is that scripting requires some programming skills. Users lacking them can use R through extensions with graphical user interfaces. R Commander

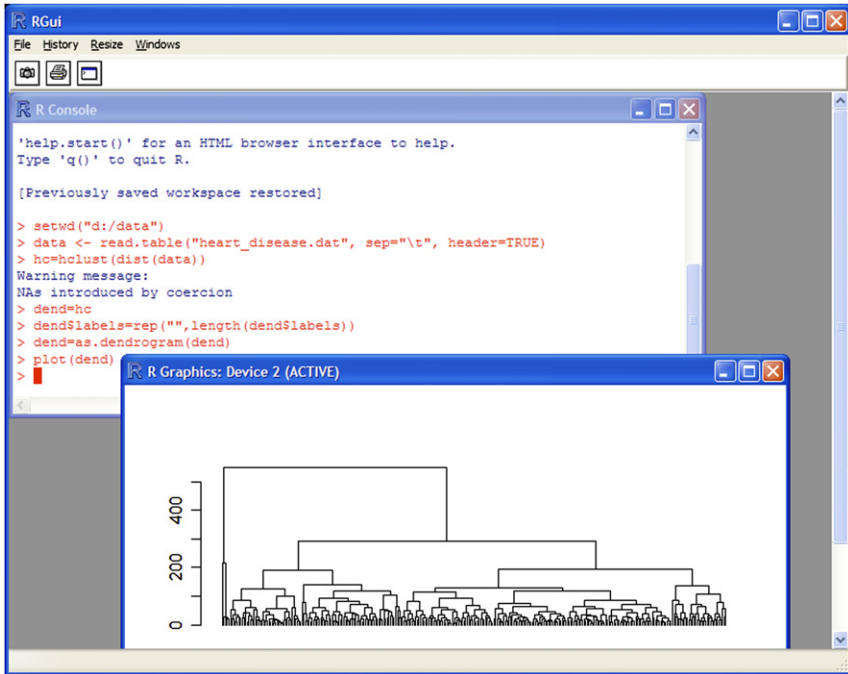


Fig. 1. Snapshot of the basic R environment (RGui) with an example script that reads the data, constructs an object that stores the result of hierarchical clustering, and displays it as a dendrogram in a separate window.

(<http://socserv.mcmaster.ca/jfox/Misc/Remdr/>), for instance, implements a graphical user interface to compose and issue commands in R script. Rattle (<http://rattle.togaware.com>), another interface extension of R, is implemented as an R library and provides a graphical user interface to many of R's data analysis and modeling functions.

### *Tanagra*

Tanagra (<http://eric.univ-lyon2.fr/~ricco/tanagra/>) is a data mining suite built around a graphical user interface wherein data processing and analysis components are organized in a tree-like structure in which the parent component passes the data to its children (Fig. 2). For example, to score a prediction model in Tanagra, the model is used to augment the data table with a column encoding the predictions, which is then passed to the component for evaluation.

Although lacking more advanced visualizations, Tanagra is particularly strong in statistics, offering a wide range of uni- and multivariate parametric and nonparametric tests. Equally impressive is its list of feature selection techniques. Together with a compilation of standard machine learning

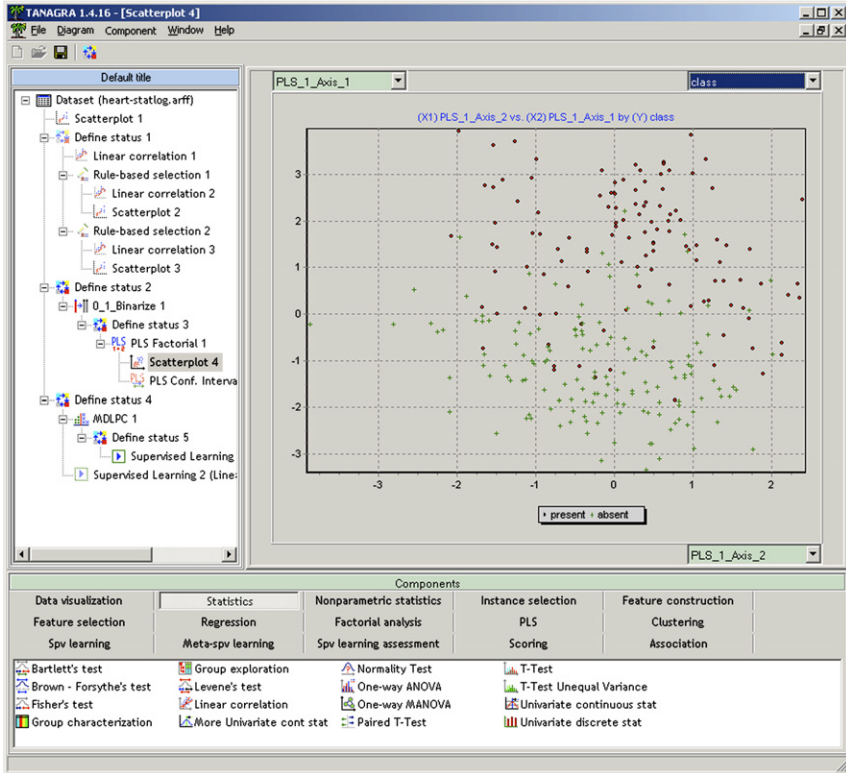


Fig. 2. Snapshots of Tanagra with an experimental setup defined in the left column, which loads the data (Dataset), shows a scatterplot (Scatterplot 1), selects a set of features (Define status 1), computes linear correlations (Linear correlation 1), selects a subset of instances based on a set of conditions (Rule-based selection 1), computes the correlation and a scatterplot for these instances, and so on. The components of the data processing tree are dragged from the list at the bottom (Components); the snapshot shows only those related to statistics. The scatterplot on the right side shows the separation of the instances based on the first two axes as found by the partial least squares analysis, where each symbol represents a patient, with the symbol's shape corresponding to a diagnosis.

techniques, it also includes correspondence analysis, principal component analysis, and the partial least squares methods. Presentation of machine learning models is most often not graphical, but instead—unlike other machine learning suites—includes several statistical measures. The difference in approaches is best illustrated by the naïve Bayesian classifier, whereby, unlike Weka and Orange, Tanagra reports the conditional probabilities and various statistical assessments of importance of the attributes (eg,  $\chi^2$ , Cramer's  $V$ , and Tschuprow's  $t$ ). Tanagra's data analysis components report their results in a nicely formatted HTML.



## Weka

Weka (Waikato Environment for Knowledge Analysis, <http://www.cs.waikato.ac.nz/ml/weka/>) [9] is perhaps the best-known open-source machine learning and data mining environment. Advanced users can access its components through Java programming or through a command-line interface. For others, Weka provides a graphical user interface in an application called the Weka KnowledgeFlow Environment featuring visual programming, and Weka Explorer (Fig. 3) providing a less flexible interface that is perhaps easier to use. Both environments include Weka's impressive array of machine learning and data mining algorithms. They both offer some functionality for data and model visualization, although not as elaborate as in the other suites reviewed here. Compared with R, Weka is much weaker in classical statistics but stronger in machine learning techniques. Weka's community has also developed a set of extensions ([http://weka.sourceforge.net/wiki/index.php/Related\\_Projects](http://weka.sourceforge.net/wiki/index.php/Related_Projects)) covering diverse areas, such as text mining, visualization, bioinformatics, and grid computing. Like R in statistics,

The screenshot shows the Weka Explorer application window. The 'Classify' tab is active, and the 'NaiveBayes' classifier is selected. The 'Test options' section shows 'Cross-validation' with 'Folds' set to 10. The 'Classifier output' pane displays the following results:

```

exercise_induced_angina: Normal Distribution. Mean = 0.55 StandardDev = 0.4975
oldpeak: Normal Distribution. Mean = 1.584 StandardDev = 1.2751 WeightSum = 120
slope: Normal Distribution. Mean = 1.8167 StandardDev = 0.5625 WeightSum = 120
number_of_major_vessels: Normal Distribution. Mean = 1.15 StandardDev = 1.03 We
thal: Normal Distribution. Mean = 6.7667 StandardDev = 1.7736 WeightSum = 120 P

```

Time taken to build model: 0.25 seconds

```

=== Stratified cross-validation ===
=== Summary ===

```

Correctly Classified Instances	226	83.7037 %
Incorrectly Classified Instances	44	16.2963 %
Kappa statistic	0.6683	
Mean absolute error	0.1835	
Root mean squared error	0.3598	
Relative absolute error	37.163 %	
Root relative squared error	72.4003 %	
Total Number of Instances	270	

```

=== Detailed Accuracy By Class ===

```

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.873	0.208	0.84	0.873	0.856	absent
0.792	0.127	0.833	0.792	0.812	present

```

=== Confusion Matrix ===

```

a	b	-- classified as	
131	19	a = absent	

Fig. 3. Weka Explorer with which we loaded the heart disease data set and induced a naïve Bayesian classifier. On the right side of the window are the results of evaluation of the model using 10-fold cross-validation.

Weka became a reference package in the machine learning community, attracting a number of users and developers. Medical practitioners would get the easiest start by using Weka Explorer, and combining it with extensions for more advanced data and model visualizations.

### YALE

Among the reviewed graphical user interface environments, the visual programming in YALE (Yet Another Learning Environment, <http://rapid-i.com>) is the closest to the traditional sense of the word “programming”: the user defines an experiment by placing the functions (eg, components for reading the data, cross-validation, applying a chain of operators, and so forth)

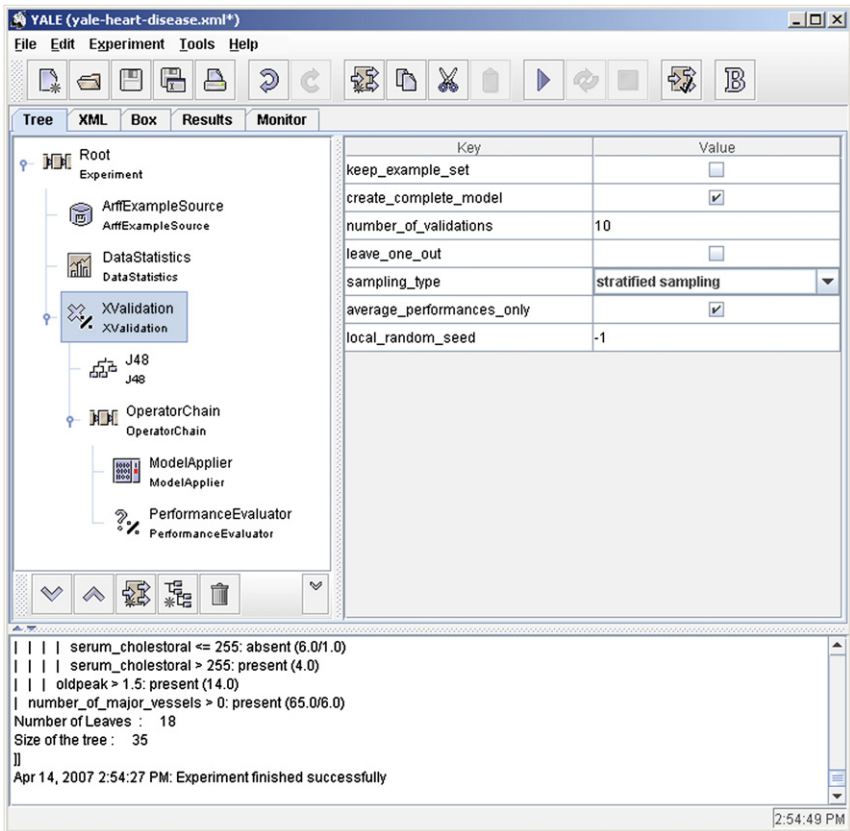


Fig. 4. A snapshot of YALE with the experimental setup for cross-validation that reads the data, computes some basic statistics about the features, and then cross-validates a classification tree inducer J48. Selection of any component from the list on the left of the window provides access to its parameters; those for cross-validation are displayed in the snapshot. The experiment log is displayed in the bottom part of the window. After executing the experiment, the results of experiments are available in the Results tab.

into a treelike structure and runs the program (Fig. 4). Internal nodes of the tree represent functions in which their children are the arguments (which may in turn be—and usually are—functions). For example, an operator “XValidation” performs cross-validation and requires two child nodes. The first must be able to handle an “ExampleSet” and deliver a “Model.” The second child node gets an ExampleSet and a Model and outputs a “PerformanceVector.” The second child would typically be an operator chain consisting of a “ModelApplier,” which uses the prediction Model on an ExampleSet, resulting in a table of predictions and actual classes and a “PerformanceEvaluator,” which takes the table and computes the corresponding classifier scores.

YALE incorporates a reasonable number of visualizations ranging from the basic histograms to multidimensional RadViz [14] projections. YALE is written in Java and is built on top of Weka, thus including its vast array of data analysis components. Although data miners with a background in programming easily grasp its visual functional programming concepts,

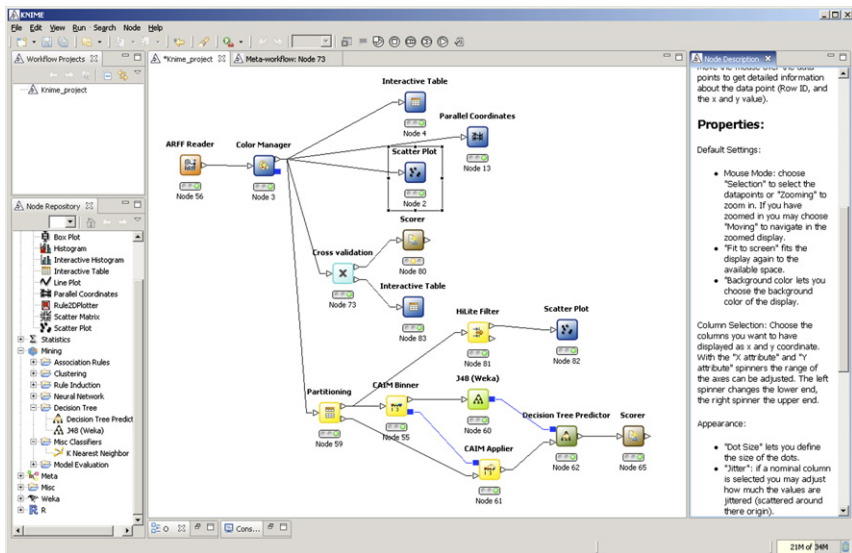


Fig. 5. Screenshot of KNIME. The central part of the window shows the experimental setup with several interconnected nodes; the right part contains a useful description of the selected node. The screenshot shows an experiment in which we loaded the data, colored the instances according to their class and showed them in a table, and used parallel coordinates and a scatterplot for visualization. In the middle of the graph we placed the nodes for testing the performance of a classification tree inducer; node “Cross-validation” has an internal workflow with the definition of the evaluated learning algorithm. At the bottom part of the graph are nodes for random partitioning of the data set, binning of the training set, and derivation of a classification tree used to predict the classes of the test set and obtain the related performance scores. In addition, we visualized the training set in a scatterplot, but put the instances through the HiLite Filter. With this setup, we can pick a node in the classification tree “J48 Weka” and see the corresponding examples in the “Scatter Plot.”

medical practitioners and researchers with limited knowledge of computer science may find them somewhat complicated to understand and manage.

### KNIME

KNIME (Konstanz Information Miner, <http://www.knime.org>) is a nicely designed data mining tool that runs inside the IBM's Eclipse development environment. The application is easy to try out because it requires no installation besides downloading and unarchiving. Like YALE, KNIME is written in Java and can extend its library of built-in supervised and

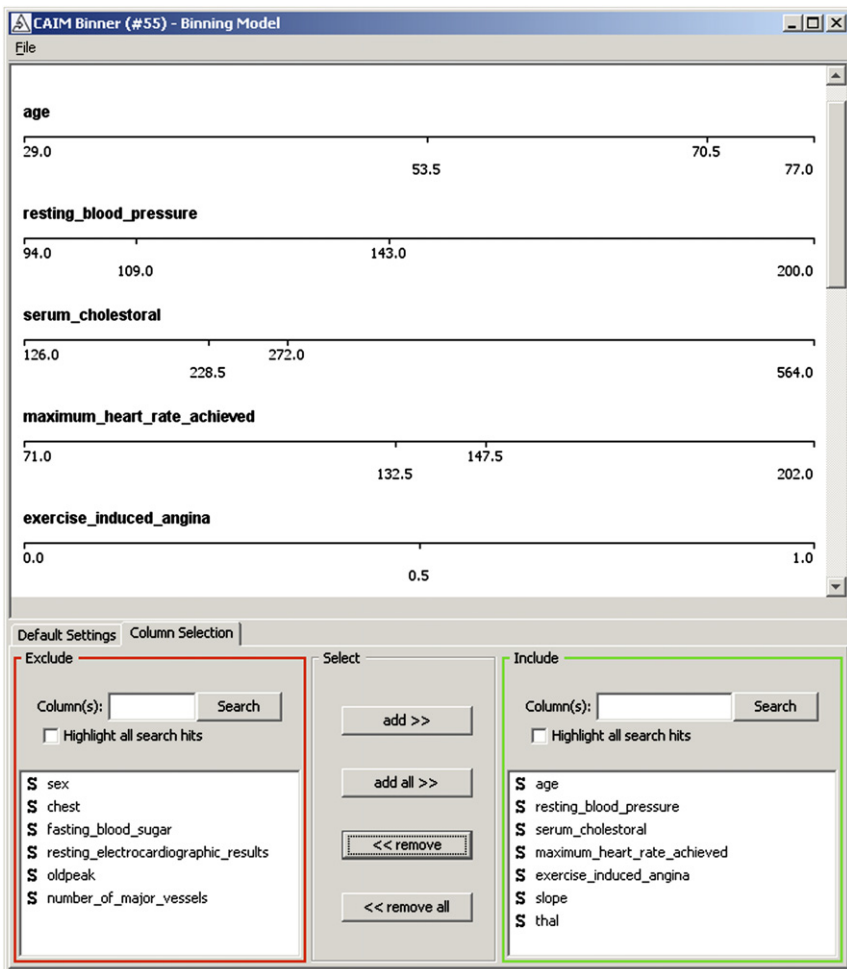


Fig. 6. A dialog of the node “CAIM Binner” (from Fig. 5) that transforms continuous features into discrete features (discretization). Features to be discretized are selected in the bottom part of the window, with the top part of the window displaying the corresponding split points.

unsupervised data mining algorithms with those provided by Weka. But unlike that of Yale, KNIME's visual programming is organized like a data flow. The user "programs" by dragging nodes from the node repository to the central part of the benchmark (Fig. 5). Each node performs a certain function, such as reading the data, filtering, modeling, visualization, or similar functions. Nodes have input and output ports; most ports send and receive data, whereas some handle data models, such as classification trees. Unlike nodes in Weka's KnowledgeFlow, different types of ports are clearly marked, relieving the beginner of the guesswork of what connects where.

Typical nodes in KNIME's KnowledgeFlow have two dialog boxes, one for configuring the algorithm or a visualization and the other for showing its results (Fig. 6). Each node can be in one of the three states, depicted with a traffic-light display: they can be disconnected, not properly configured, or lack the input data (red); be ready for execution (amber); or have finished the processing (green). A nice feature called HiLite (Fig. 7) allows the user to select a set of

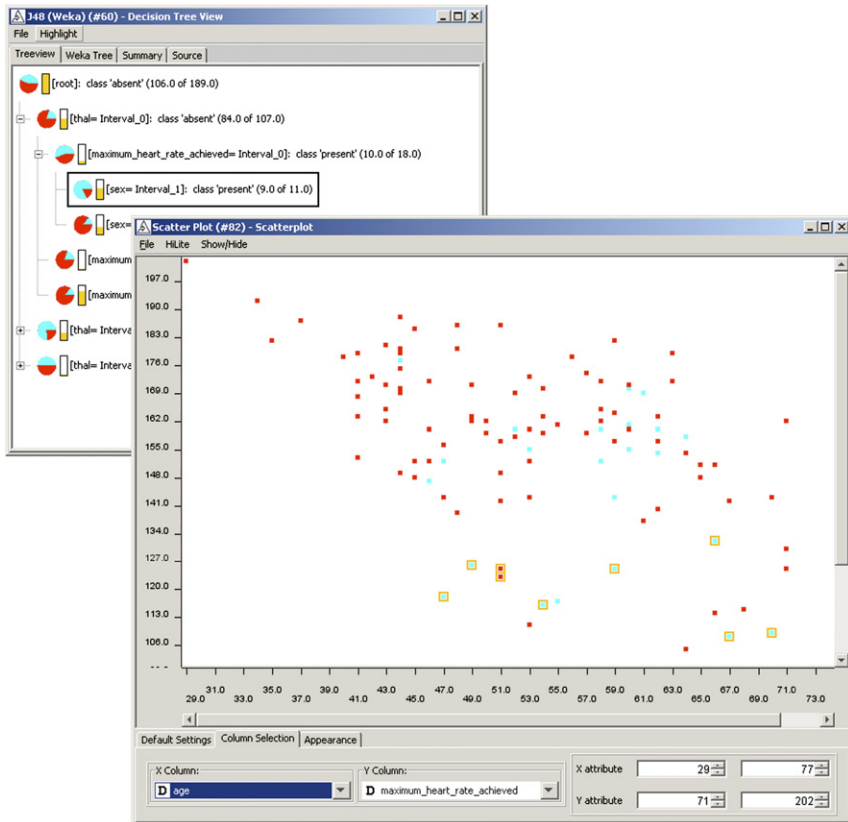


Fig. 7. KNIME HiLiteing (see Fig. 5), where the instances from the selected classification tree node are HiLited and marked in the scatterplot.

instances in one node and have them marked in any other visualization in the current application, in this way further supporting exploratory data analysis.

*Orange*

Orange (<http://www.ailab.si/orange>) is a data mining suite built using the same principles as KNIME and Weka KnowledgeFlow. In its graphical environment called Orange Canvas (Fig. 8), the user places *widgets* on a *canvas* and connects them into a schema. Each widget performs some basic function,

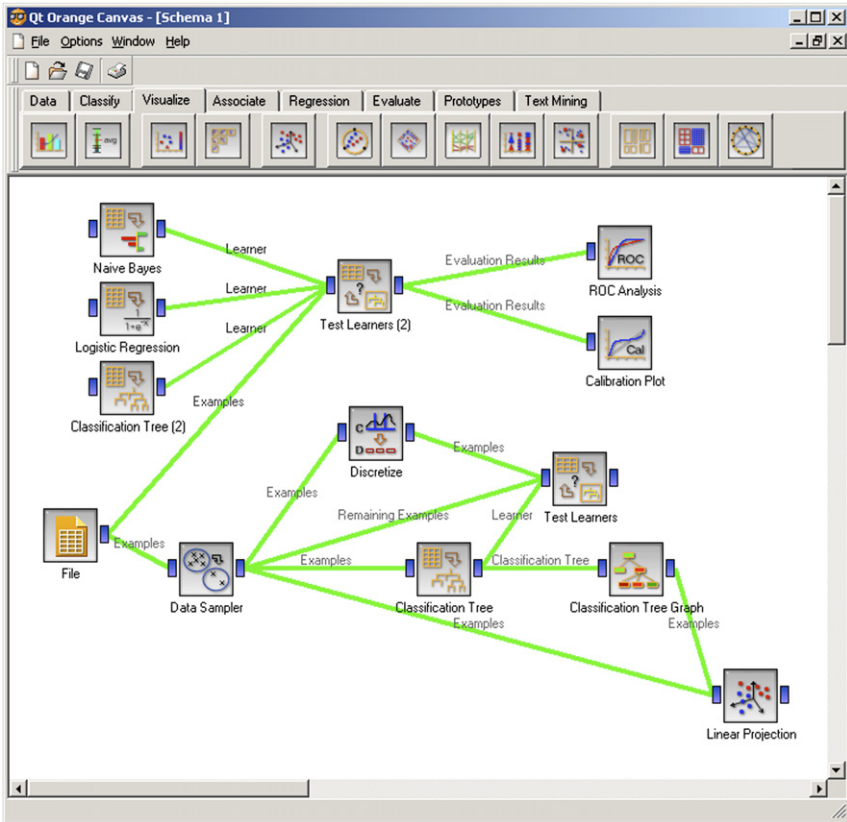


Fig. 8. Snapshot of the Orange canvas. The upper part of the schema centered around “Test Learners” uses cross-validation to compare the performance of three classifiers: naïve Bayes, logistic regression, and a classification tree. Numerical scores are displayed in “Test Learners,” with evaluation results also passed on to “ROC Analysis” and “Calibration Plot” that provide means to graphically analyze the predictive performance. The bottom part contains a setup similar to that in KNIME (see Fig. 5): the data instances are split into training and test sets. Both parts are fed into “Test Learners,” which, in this case, requires a separate test set and tests a classification tree built on the training set that is also visualized in “Classification Tree Graph.” “Linear Projection” visualizes the training instances, separately marking the subset selected in the “Classification Tree Graph” widget.

but unlike in KNIME with two data types—models and sets of instances—the signals passed around Orange’s schemata may be of different types, and may include objects such as learners, classifiers, evaluation results, distance matrices, dendrograms, and so forth. Orange’s widgets are also coarser than KNIME’s nodes, so typically a smaller number of widgets is needed to accomplish the same task. The difference is most striking in setting up a cross-validation experiment, which is much more complicated in KNIME, but with the benefit of giving the user more control in setting up the details of the experiment, such as separate preprocessing of training and testing example sets.

Besides friendliness and simplicity of use, Orange’s strong points are a large number of different visualizations of data and models, including intelligent search for good visualizations, and support of exploratory data analysis through interaction. In a concept similar to KNIME’s HiLiteing (yet subtly different from it), the user can select a subset of examples in a visualization, in a model, or with an explicit filter, and pass them to, for instance, a model inducer or another visualization widget that can show them as a marked subset of the data (Fig. 9).

Orange is weak in classical statistics; although it can compute basic statistical properties of the data, it provides no widgets for statistical testing. Its

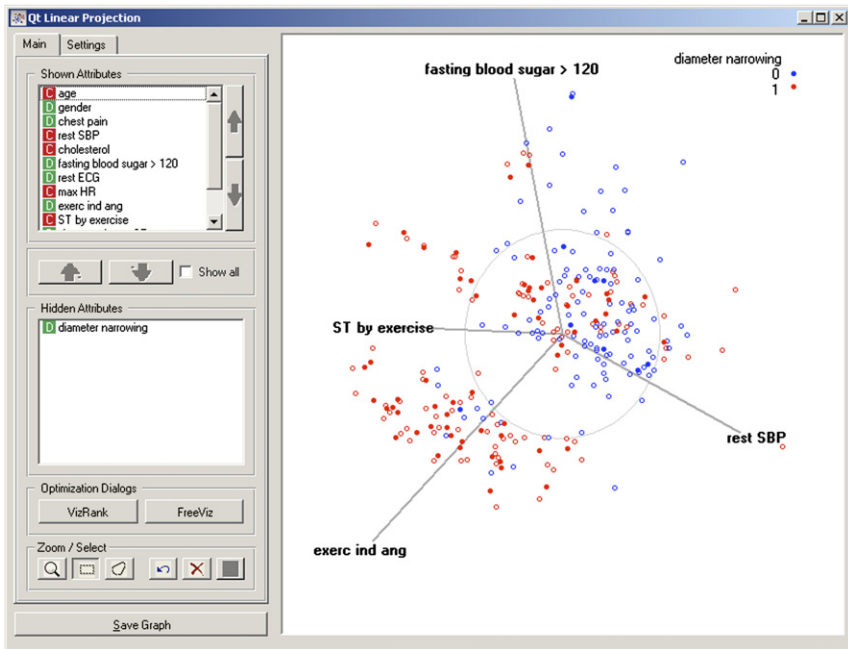


Fig. 9. The linear projection widget from Orange displaying a two-dimensional projection of data, where the x and y axis are a linear combination of feature values whose components are delineated with feature vectors. Coming from the schema shown in Fig. 8, the points corresponding to instances selected in the classification tree are filled and those not in the selection are open.

reporting capabilities are limited to exporting visual representations of data and models. Similar to R, the computationally intensive parts of Orange are written in C++, whereas the upper layers are developed in the scripting language Python, allowing advanced users to supplement the existing suite with their own algorithms or with routines from Python's extensive scientific library (<http://www.scipy.org>).

### GGobi

Data visualization was always considered one of the key tools for successful data mining. Particularly suited for data mining and explorative data analysis, GGobi (<http://www.ggobi.org>) is an open-source visualization program featuring interactive visualizations through, for instance, *brushing* (Fig. 10), whereby a user's selection is marked in all other opened visualizations, and *grand tour* (Fig. 11) [15], which uses two-dimensional visualizations and in a movie-like fashion shifts between two different projections. GGobi can also plot networks, a potentially useful feature for analysis of larger volumes of data, such as those from biomedicine. By itself GGobi is only intended for visualization-based data mining, but can be nicely integrated with other statistical and data mining approaches when used as a plug-in for R or used through interfaces for the scripting languages Perl and Python.

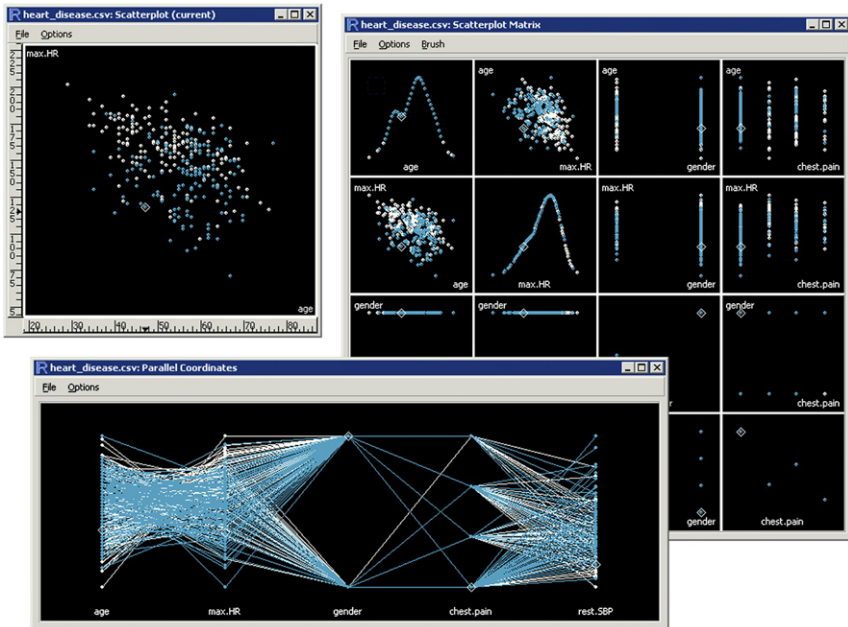


Fig. 10. Scatterplot, a matrix of scatterplots and parallel coordinates as displayed by GGobi. The instances selected in one visualization (scatterplot, in this case) are marked in the others.



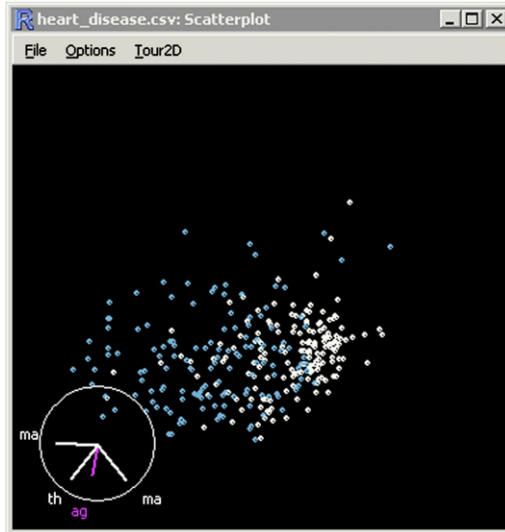


Fig. 11. GGobi's Grand tour shows a projection similar to the Linear Projection in Orange (see Fig. 9) but animates it by smoothly switching between different interesting projections, which gives a good impression of positions of the instances in the multidimensional space.

## Summary

State-of-the-art open-source data mining suites of today have come a long way from where they were only a decade ago. They offer nice graphical interfaces, focus on usability and interactivity, support extensibility through augmentation of the source code or (better) through use of interfaces for add-on modules. They provide flexibility through either visual programming within graphical user interfaces or prototyping by way of scripting languages. Major toolboxes are well documented and use forums or discussion groups for user support and exchange of ideas.

The degree to which all of the above is implemented of course varies from one suite to another, but in the packages we have reviewed in this article most of the above issues were addressed and we could not find a clear winner in supporting all of the aspects in the best way. For a medical practitioner or biomedical researcher starting with data mining the choice for the right suite may be guided by the simplicity of the interface, whereas for research teams a choice of implementation or integration language (Java, R, C/C++, Python, and so forth) may be important. For the wish list of data mining techniques we find that all packages we have reviewed (with the exception of GGobi focusing on visualization only) cover most of the standard data mining operations, ranging from preprocessing to modeling, with some providing better support for statistics and others for visualization.

There are many open-source data mining tools available, and our intention was only to demonstrate the ripeness of the field through exemplary

implementations. We covered only general-purpose packages, and also because of space limitations did not discuss any of the specialized software tools dealing with biomedical data analysis, such as text mining, bioinformatics, microarray preprocessing, analysis in proteomics, and so forth, some of which are addressed in other articles in this issue. The number of such tools is large, with new development projects being established almost on a daily basis. Not all of these will be successful in the long term, but many of them are available, stable, and have already been used in a large number of studies. With growing awareness that in science we should share the experimental data and knowledge, along with the tools we build to analyze and integrate them, open-source frameworks provide the right environment for community-based development, fostering exchange of ideas, methods, algorithms and their implementations.

## References

- [1] Fayyad U, Piatetsky-Shapiro G, Smyth P, et al, editors. *Advances in knowledge discovery and data mining*. Menlo Park (CA): AAAI Press; 1996.
- [2] Quinlan JR. *C4.5: programs for machine learning*. San Mateo (CA): Morgan Kaufmann Publishers; 1993.
- [3] Michalski RS, Kaufman K. Learning patterns in noisy data: the AQ approach. In: Paliouras G, Karkaletsis V, Spyropoulos C, editors. *Machine learning and its applications*. Berlin: Springer-Verlag; 2001. p. 22–38.
- [4] Clark P, Niblett T. The CN2 induction algorithm. *Machine Learning* 1989;3:261–83.
- [5] Asuncion A, Newman DJ. UCI Machine Learning Repository. Available at: <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Accessed April 15, 2007. Irvine, CA: University of California, Department of Information and Computer Science; 2007.
- [6] Wall L, Christiansen T, Orwant J. *Programming Perl*. 3rd edition. Sebastopol, CA: O'Reilly Media, Inc.; 2000.
- [7] Kohavi R, Sommerfield D, Dougherty J. Data mining using MLC++: a machine learning library in C++. *International Journal on Artificial Intelligence Tools* 1997;6:537–66.
- [8] Brunk C, Kelly J, Kohavi R. MineSet: an integrated system for data mining. In *Proc. 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, Menlo Park (CA). p. 135–8.
- [9] Witten IH, Frank E. *Data mining: practical machine learning tools and techniques with Java implementations*. 2nd edition. San Francisco (CA): Morgan Kaufmann; 2005.
- [10] Zupan B, Holmes JH, Bellazzi R. Knowledge-based data analysis and interpretation. *Artif Intell Med* 2006;37:163–5.
- [11] Bellazzi R, Zupan B. Predictive data mining in clinical medicine: current issues and guidelines. *Int J Med Inform* 2006; in press.
- [12] Cios KJ, Moore GW. Uniqueness of medical data mining. *Artif Intell Med* 2002;26:1–24.
- [13] Becker RA, Chambers JM. *S: an interactive environment for data analysis and graphics*. Pacific Grove (CA): Wadsworth & Brooks/Cole; 1984.
- [14] Hoffman PE, Grinstein GG, Marx KE. DNA visual and analytic data mining. Phoenix (AZ): In *Proc. IEEE Visualization*; 1997. p. 437–41.
- [15] Asimov D. The grand tour: a tool for viewing multidimensional data. *SIAM J Sci Statist Comput* 1985;6:128–43.