Concurrent software architectures for exploratory data analysis

Anže Starič,[1] Janez Demšar[1] and Blaž Zupan[1,2]*

Decades ago, increased volume of data made manual analysis obsolete and prompted the use of computational tools with interactive user interfaces and rich palette of data visualizations. Yet their classic, desktop-based architectures can no longer cope with the ever-growing size and complexity of data. Next-generation systems for explorative data analysis will be developed on client–server architectures, which already run concurrent software for data analytics but are not tailored to for an engaged, interactive analysis of data and models. In explorative data analysis, the key is the responsiveness of the system and prompt construction of interactive visualizations that can guide the users to uncover interesting data patterns. In this study, we review the current software architectures for distributed data analysis and propose a list of features to be included in the next generation frameworks for exploratory data analysis. The new generation of tools for explorative data analysis will need to address integrated data storage and processing, fast prototyping of data analysis pipelines supported by machine-proposed analysis workflows, pre-emptive analysis of data, interactivity, and user interfaces for intelligent data visualizations. The systems will rely on a mixture of concurrent software architectures to meet the challenge of seamless integration of explorative data interfaces at client site with management of concurrent data mining procedures on the servers. © 2015 John Wiley & Sons, Ltd.

How to cite this article:
WIREs Data Mining Knowl Discov 2015, 5:165–180. doi: 10.1002/widm.1155

INTRODUCTION

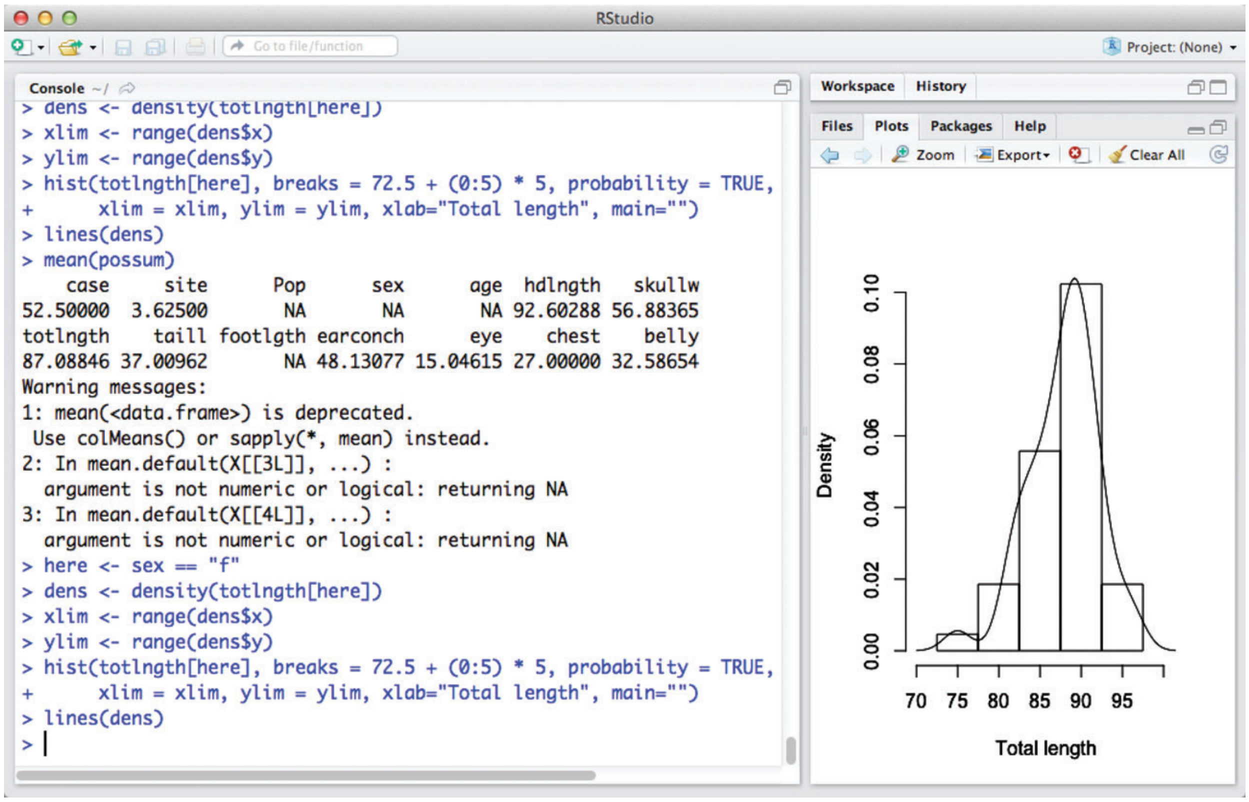
Any data analysis is, in its essence, interactive. We preprocess data, identify the outliers, and develop descriptive and predictive models. But at the same time, we wish to know which parts of data support the reached conclusions and which contradict them. The patterns that we discover—data associations, correlations, interactions—become meaningful only after being traced back to the data. Data analysis involves the analyst in a deep and engaging process of discovery. According to John W. Tukey, 'In exploratory data analysis there can be no substitute for flexibility; for adapting what is calculated—and what we hope plotted—both to the needs of the situation and the clues that the data have already provided'.[1] Data exploration is a dynamic process, in which the interaction with visualizations and other analysis leads the researcher ever deeper into understanding of the underlying data, its structure and interaction between its components. Tukey's introduction of exploratory data analysis inspired the development of many current statistical tools. The early data analysis systems supported interaction and exploratory analysis through scripting. In R[2] (Figure 1), the user issues data processing commands and observes their results through textual output or data visualizations. Domain experts are not necessarily fluent in programming. Graphical environments that retain flexibility through innovative human-computer interfaces, such as visual programming (Figure 2) are an alternative to scripting. In platforms such as KNIME,[3] Orange,[4] and TANAGRA,[5] visual programming or construction of data mining diagrams can provide an engaging experience by offering simple interfaces for execution of

*Correspondence to: blaz.zupan@fri.uni-lj.si

[1]Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia

[2]Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, TX, USA

Conflict of interest: The authors have declared no conflicts of interest for this article.



```
> dens <- density(totlngth[here])
> xlim <- range(dens$x)
> ylim <- range(dens$y)
> hist(totlngth[here], breaks = 72.5 + (0:5) * 5, probability = TRUE,
+      xlim = xlim, ylim = ylim, xlab="Total length", main="")
> lines(dens)
> mean(possum)
    case     site      Pop      sex      age  hdlngth   skullw
52.50000  3.62500       NA       NA       NA 92.60288 56.88365
totlngth    taill footlgth earconch      eye    chest    belly
87.08846 37.00962       NA 48.13077 15.04615 27.00000 32.58654
Warning messages:
1: mean(<data.frame>) is deprecated.
 Use colMeans() or sapply(*, mean) instead.
2: In mean.default(X[[3L]], ...) :
  argument is not numeric or logical: returning NA
3: In mean.default(X[[4L]], ...) :
  argument is not numeric or logical: returning NA
> here <- sex == "f"
> dens <- density(totlngth[here])
> xlim <- range(dens$x)
> ylim <- range(dens$y)
> hist(totlngth[here], breaks = 72.5 + (0:5) * 5, probability = TRUE,
+      xlim = xlim, ylim = ylim, xlab="Total length", main="")
> lines(dens)
> |
```

FIGURE 1 | RStudio has a command line interface to invoke analysis methods and plot data graphs. Visualizations are displayed in a separate window (right), and the next scripting steps often strongly rely on the displayed results.

complex analysis. Visual programming environments rely strongly on data visualization and interactions.[6] Users can interact with visualized data and select data subsets to explore them in the emerging analysis pipeline. Interactive data analysis also takes place in modern spreadsheet applications, where a few tricks suffice to construct powerful computational procedures and exceptional data visualizations.

Desktop data analysis tools work well if the data are small enough to fit into the working memory of a commodity computer. They fail, however, with data's increasing size and complexity. Explorative data analysis requires responsive systems. Clogging and substantial response delays due to computationally complex analysis procedures, resampling-based estimations, or rendering of visualizations of large data sets hinder interactivity and discourage the user from exploring. Exploratory data analysis systems are on the crossroad. Desktop-based systems with interactive data analysis interfaces are failing to cope with larger data sets. Analysis of such data requires concurrent architectures for large-scale data mining,[7,8] but they may lack the necessary front-ends for exploratory analysis. Data mining tools that will offer the best of the two worlds need to address some major software engineering challenges. Computational speed, communication of partial results, accurate reporting on progress, and adaptation to dynamic changes in the analysis pipeline are just a few aspects to consider.

In the next chapter, we review major state-of-the-art software architectures that can support concurrent data analysis. These architectures were designed to speed-up the analysis processes through concurrent execution of computational procedures, buy lack several features that are intrinsic to interactive and exploratory data analysis. For each of the software architectures, we provide a review, expose its main advantages and highlight its shortcomings that are related to its utility in interactive data analysis. Based on the review, we next present an abstract model of a software architecture for interactive data analytics. Components of this model are further discussed in the section that lists advanced features of next-generation exploratory analysis software environments. We summarize our review in the last section of the article and conclude that present software architectures will need a number of specific but gradual adaptations to support interactive, user-engaged data analysis.

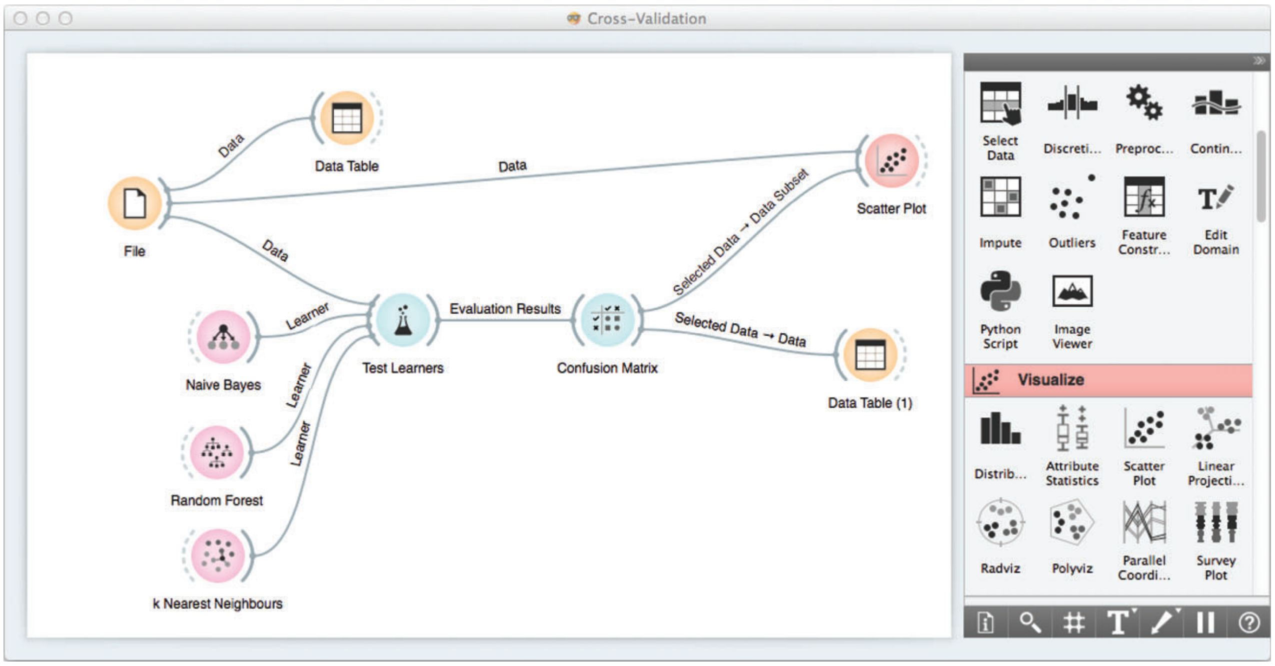


FIGURE 2 | A workflow editor from Orange data mining toolbox, with an example workflow for cross-validation of supervised data mining algorithms and analysis of results. Computational units are represented with icons and communicate through typed communication channels (gray lines).

ARCHITECTURES FOR CONCURRENT DATA ANALYSIS

Analysis of large data sets has gained vast interest from scientists and data miners in recent years. Developers of data mining tools have thus far focused on different aspects of the problem, such as designing algorithms that take advantage of specialized data storage, parallelization of analysis problems, and visualization of large data sets. Various software toolboxes have been crafted to support the data analysis process by allowing users to reuse or upgrade existing analytics components to create new analysis pipelines.

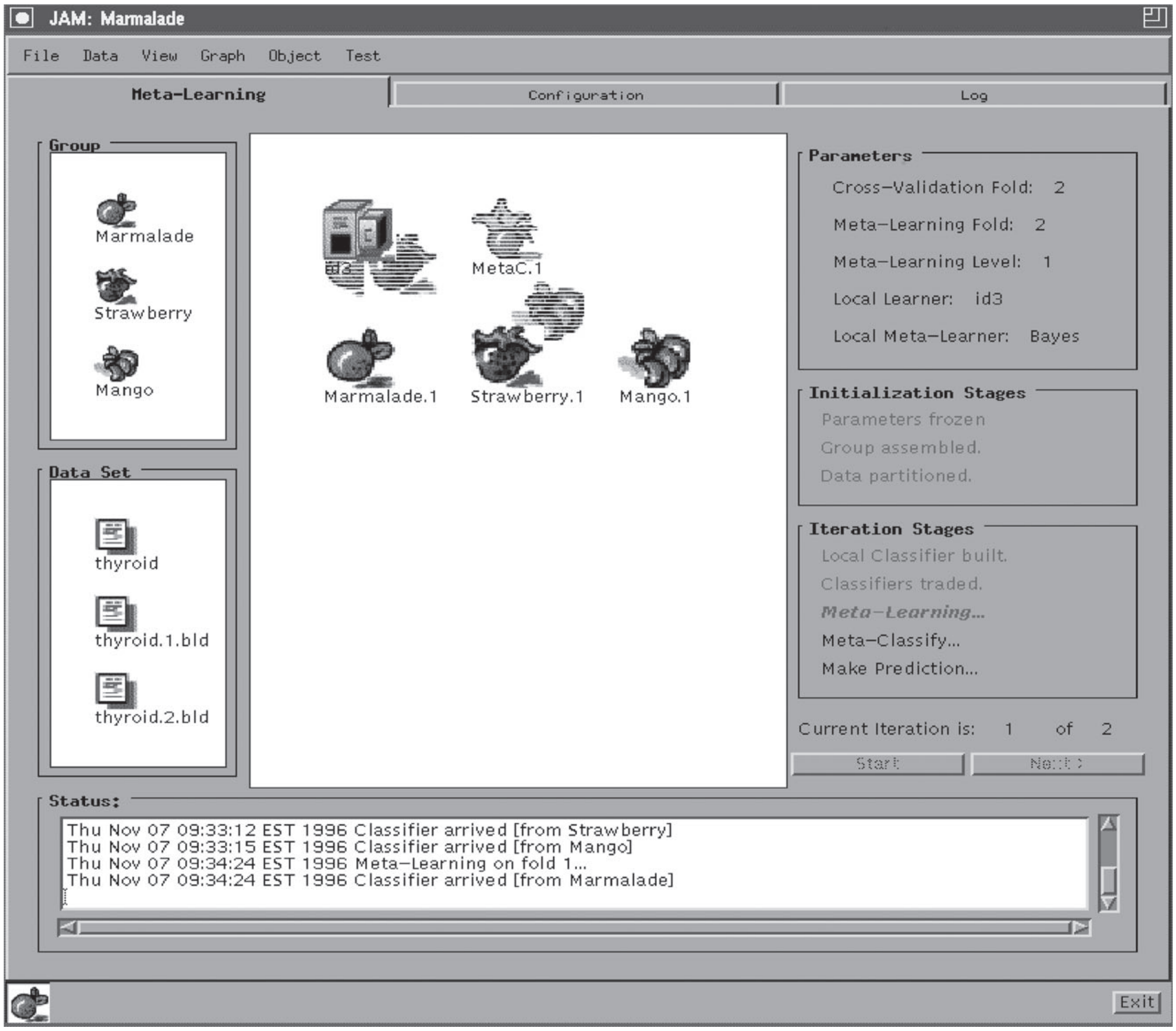
Software Agents

Agent-based architecture is among the oldest distributed architectures for data analysis. Software agents receive instructions from the user or from other agents, independently perform the analysis, and report the results. This approach scales well with the size of data if the modeling can use only small localized data subsets. Agents are employed concurrently, each reporting a model to be fused with others to gain understanding of the entire domain. Agents can also handle distributed data sets that cannot be moved due to privacy or regulatory concerns.[9]

Ideally, agents would learn and adapt over time and produce meaningful results with very limited interaction with the user. This goal has never been achieved. Instead, agents in the present systems are simple and specialized, and often dependent on the user's interaction through graphical user interfaces (Figure 3)[9] or query languages.[10,11] Some systems are also able to graphically present resulting data models, and, e.g., provide visualizations of decision trees[9] or interactive graphs.[11] Some also report on the progress of the agent-based execution of analysis.

Agent-based architectures take various approaches to parallel execution of data mining tasks. JAM[9] is designed to treat data as stationary, and considers data sets at different locations to be private and belonging to various independent organizations. Only the induced models are communicated between agents and fused into a final, resulting model. Papyrus,[10] however, can move data from one location to another, and considers various trade-offs between local computation and transfer of the data to multiple computers to optimize computational load.

Each data analysis agent produces its own model. In principle, and when only predictions are important, the models may be combined through ensembles by, say, stacking[12] or averaging. Fused models are very hard to interpret, however, and special procedures like data visualizations and feature scoring[13] are needed for their appropriate inclusion into exploratory data analysis.



Parameters
Cross-Validation Fold: 2
Meta-Learning Fold: 2
Meta-Learning Level: 1
Local Learner: id3
Local Meta-Learner: Bayes

Initialization Stages
Parameters frozen
Group assembled.
Data partitioned.

Iteration Stages
Local Classifier built.
Classifiers traded.
Meta-Learning...
Meta-Classify...
Make Prediction...

Current Iteration is: 1 of 2

Status:
```
Thu Nov 07 09:33:12 EST 1996 Classifier arrived [from Strawberry]
Thu Nov 07 09:33:15 EST 1996 Classifier arrived [from Mango]
Thu Nov 07 09:34:24 EST 1996 Meta-Learning on fold 1...
Thu Nov 07 09:34:24 EST 1996 Classifier arrived [from Marmalade]
```

FIGURE 3 | Interface for meta learning in JAM architecture showing the progress of the analysis execution (on the right).

Agent based architectures have been used in wide range of data analysis applications that include text mining,[11] medical data analysis,[10] and credit card fraud detection.[9] Agents were found to be of particular benefit when the datasets were spread across multiple sites and could not be moved or combined due to the regulatory restrictions.

Web Services

Web services are a popular technology that allows us to take advantage of remote computational resources.[14] Services based on Web Service Definition Language (WSDL) allow easy discovery by providing a description of the interface as a part of the service. In contrast, Representational State Transfer (REST) services strive toward simplicity and rely on the features included in the HTTP protocol.

Several modern data analysis systems use web services to execute the analysis on remote computers. Weka4WS[15] has the same graphical user interface as Weka,[16] but uses web services to remotely analyze data. Taverna[17] is a workflow management system for creating and executing scientific workflows. It contains references to more than 3500 services that can be added to the user-designed data analysis pipeline. Taverna consists of Taverna Workbench (Figure 4), a desktop application for workflow design, and Taverna Server for remote execution of workflows.

Taverna workflows strongly depend on services, which are its sole means of data processing. There is no local, client-based data analysis. In this way,

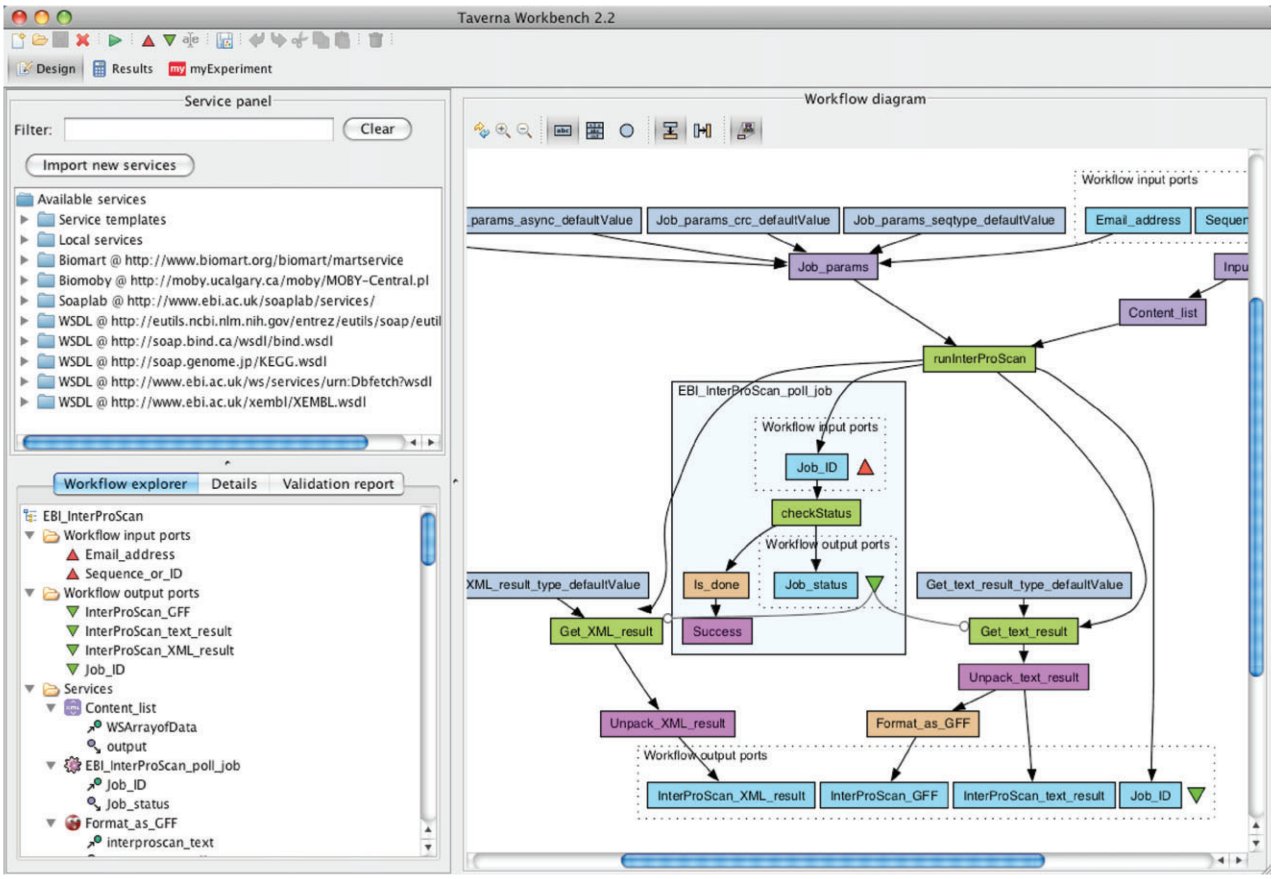


FIGURE 4 | Taverna workbench can be used to design complex workflows (on the right) from the list of available services (on the left).

Taverna simplifies and elegantly unifies the analytics architecture. The main bottleneck of the approach is data communication. Services need to exchange data, and if these are large, their transfers can take much longer than the actual computations. Orange4WS,[18] however, can construct workflows from components that are executed either locally or through webservice calls to remote components. Based on the workflow manager in Orange,[4] each component of the workflow executes its tasks as soon as it has all the necessary inputs, which enables the user to get preliminary results during construction of the analysis pipeline. Another distinctive feature of Orange4WS is automatic construction of workflows. The analytic components are annotated with terms from a Knowledge Discovery Ontology[19] and can be assembled into analysis workflow from the user's specification of desired inputs and outputs.

Service oriented architectures for data analysis also have several shortcomings.[17] In an analysis composed of multiple services, each of them has its own processing queue, which is usually hidden from the user. As a result, execution times are hard to predict when some of the services are experiencing heavy load. Service descriptions are sufficient to establish a connection, but their readability depends on the service author and is not subject to a standard. Service inputs with names such as '*arg1*' or parameters of the type such as '*string*' suffice for generating a valid request, but are not informative. Orange4WS tackles this problem with additional annotations constructed manually and added independently of the service provider. Problems also arise when combining services from different providers. The input and output data types of services may not match and the user has to manually provide the necessary conversions. Services also need to transfer all the data related to requests, which is unfeasible for large data sets. The Web Service Resource Framework (WSRF, http://www.globus.org/wsrf/specs/ws-wsrf.pdf) partially solves this by storing the data as a resource and using resource identifiers in the requests. Weka4WS supports WSRF and further provides a WSRF-based service that can be executed on grids.

Web services have found their utility in various application areas, including astronomy,[20] biology,[21] chemistry,[22] and text mining.[23] BioCatalogue (https://www.biocatalogue.org), e.g., includes over 1000 different web services from different areas of life sciences, where a substantial number of them deal with data access, analysis, and visualization. These and similar web services can be composed into workflows using open source tools such as Taverna (http://www.taverna.org.uk) or Triana (http://www.trianacode.org/). Using the WSRF standard, web services can take advantage of grids to spread the work on multiple computation nodes.

Grids

Grids solve the problem of controlled and coordinated resource sharing and resource use in dynamic, scalable virtual organizations.[24] Multiple organizations can share data, analysis procedures, and computational resources. Grids also implement various security and access policies.

Grid software architectures for data analysis are developed on the existing grid services for data transfer and management, allocation, monitoring, and control of computational resources. These provide the framework to construct additional layers of services that implement various data mining procedures. Grid services expose their interfaces and usage policies. A high-level client can discover and combine them into an analysis pipeline.

Many current grid-based data analysis frameworks offer graphical interfaces to assemble analysis pipelines. In workflow editors, we can select different services, set their parameters, and establish communication, while services on the grid execute the data analysis tasks. The pipeline editor in Discovery net[25] can verify the validity of the workflow from service metadata. When multiple sites provide the same service, Discovery net users can manually map tasks to specific computational resources. An alternative to designing implementation-specific graphical editors is to couple grid services with open source workflow managers. These fetch information on available services from a service catalog provided by the grid. The DataMiningGrid,[26] e.g., uses the Triana[27] workflow editor.

Job scheduling and parallel execution on grids are managed by a resource broker, which receives requests and delegates them to the underlying grid execution system, such as HTCondor.[28] Independent services concurrently run on different machines, and data-aware scheduling minimizes data transfer.

Grids have been used for Gene Annotation,[25] Ecosystem Modeling,[26] and Text Analysis.[26] Data analysis pipeline can be designed in workflow editors such as Taverna (http://www.taverna.org.uk/) or Triana (http://www.trianacode.org/) and executed concurrently on the grid.

MapReduce and Hadoop

MapReduce[7] is a popular approach for processing large amounts of data. By limiting the programmer to the tasks that can be expressed as a series of map and reduce steps, MapReduce provides a high level of parallelism on a large number of commodity machines. The core of MapReduce technology is a distributed file system where the data are redundantly stored on multiple machines.

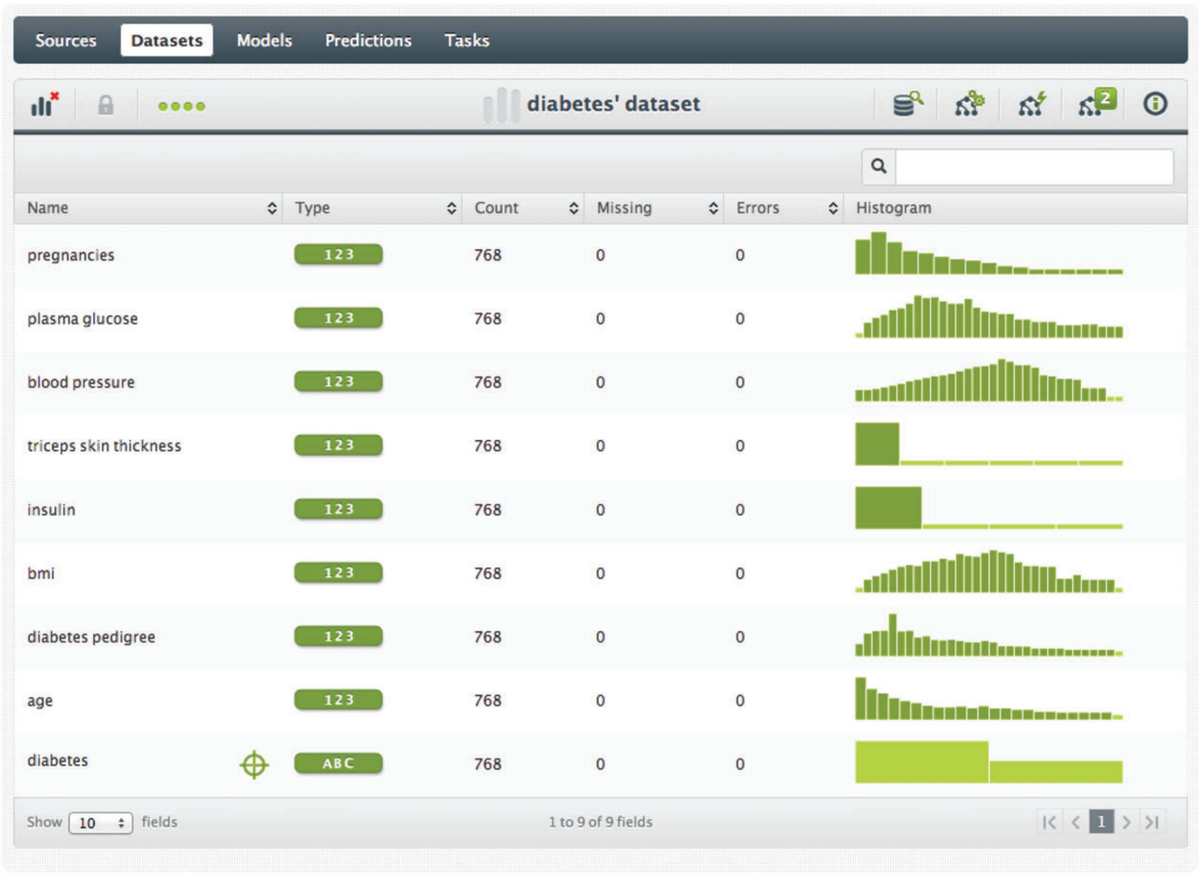
Data analysis algorithms can be expressed as a series of map and reduce steps. The map operation is applied to the entire data set and yields intermediary results. The distributed nature of the underlying file system ensures that processing on different parts of the data can be performed with different machines without moving any data. Outputs of the map step are assembled and processed in the reduce step to yield the final result.

MapReduce clusters can contain thousands of machines. The probability of hardware failure rises with the size of cluster. The software architecture thus contains mechanisms to circumvent failures and slow or unresponsive workers by reassigning the corresponding jobs to another machine that holds duplicate data.

Apache Hadoop (http://hadoop.apache.org) is a popular open source implementation of the MapReduce architecture and a general-purpose framework. Its data analysis extension is implemented within multiple libraries; Apache Mahout (http://mahout.apache.org) for data mining, BioPig[29] and SeqPig[30] for sequencing data and Pegasus[31] for mining graphs. Data analysis code can be written in Java, PIG, SQL,[32] or R programming language.[33,34] Commercial third party tools such as Pentaho Business Analytics or IBM InfoSphere provide some limited support for graphical design of MapReduce analysis pipelines.

While MapReduce approach works for batch processing of data, other approaches are used when data needs to be queried because of their short response times. Apache HBase is a data store inspired by Google BigTable.[35] It is built on top of the Hadoop File System and supports real-time read/write access to the data. Interactive queries of the data are possible with Dremel[36] (marketed as Google BigQuery). It uses a combination of columnar storage and hierarchical processing to execute aggregate queries on petabytes of data in a few seconds. Project Apache Drill aims to provide an Open Source implementation of functionality provided by Dremel.

An alternative approach is open source cluster computing system Spark.[37] Its responsiveness is a result of keeping the data in memory between requests,



Name	Type	Count	Missing	Errors	Histogram
pregnancies	123	768	0	0	
plasma glucose	123	768	0	0	
blood pressure	123	768	0	0	
triceps skin thickness	123	768	0	0	
insulin	123	768	0	0	
bmi	123	768	0	0	
diabetes pedigree	123	768	0	0	
age	123	768	0	0	
diabetes	ABC	768	0	0	

Show 10 fields 1 to 9 of 9 fields 1

FIGURE 5 | The web application BigML allows users to upload their data sets, build and visualize decision tree models, and use them on new data. Each of the application's tabs (top row) is associated with one of the available tasks carried out within a simple interface also suitable for users with little or no data mining experience.

which can benefit exploratory data analysis. Tools such as RABID[38] can be used to execute R code on Spark clusters.

MapReduce can deal with large amounts of data, but is it ready for exploratory data analysis? Researchers perform exploratory studies on Hadoop[39] implementations of interactive querying of data have shown that interactive analysis of the larger data sets is possible given enough computers to parallelize the job. However, survey of Hadoop usage in research workflows[40] shows that high level tools are still not in regular use and that majority of analyses is performed by writing low-level MapReduce code. Optimization of Hadoop for small jobs[41] and algorithms for data visualization on MapReduce architectures[42] are two example research areas that are paving the way for exploratory analysis on MapReduce clusters.

Cloud Applications

Cloud computing offers a number of opportunities for data analysis applications. Perhaps the most promising is horizontal scaling,[43] where the number of rented workers dynamically changes according to the requirements of the analysis procedures.

Several applications allow users to upload data and use cloud computing for data analysis. BigML (https://bigml.com) implements a set of basic data mining procedures that are accessed through a simple graphical interface (Figure 5), making this web application also suitable for users with no formal background in data analytics. BigML includes several advanced procedures that support interactive analysis of big data. For instance, data histograms are first rendered on a data sample and then updated when the entire data set is loaded and processed. Decision trees are a supervised machine-learning technique that starts with the root classification node, which is then iteratively refined. Decision trees in BigML are rendered on the client application dynamically and evolve as their nodes are induced on the computational server.

Google Prediction API (https://developers.google.com/prediction) does not support interactivity, but provides an interface to supervised data analysis

TABLE 1 | Five Features of Concurrent Software Architectures That Play an Important Role in Implementations of Exploratory Data Analysis Environments, and Their Support by a Set of Existing Architectures. A Partial Support Is Noted Where the Software Architecture Was Not Designed for the Task But Could in Principle Be Adapted for It

	Fine-Grained Parallel Execution	Interactive Visualization	Communication of Intermediate Results	High-Level Execution Environment	Data Stays on Premise
Agents	No	Yes	Yes	No	Yes
Web services	No	Yes	No	Yes	No
Grids	Yes	No	Partially	Yes	Partially
Map reduce	Yes	No	No	Partially	Partially
Cloud	Yes	Yes	Yes	Yes	No

on the cloud. This is one of the earliest cloud-based data mining solutions, but it is not suitable for explorative data analysis as it returns only predictions and not models.

Cloud-based architectures are a low-cost option that enables large-scale data analysis for organizations that by themselves do not own the adequate computational resources. The downside is the need to transfer the data to and from the cloud. Data transfer takes time and may be in part restricted by corporate policies or government regulations. From the viewpoint of explorative data analysis, the principal bottleneck of cloud-based architectures is the lack of their support for responsive visualizations. Existing JavaScript libraries, such as InfoViz (http://thejit.org), D3 (http://d3js.org), or Highcharts (http://www.highcharts.com), support beautiful visualizations, but all assume that the entire data set is locally available for rendering. Resulting visualizations can be interactive, but again only if the entire data set is already present on the client side. Visualizations with multiple levels of detail required for exploration of big data are currently not available. Existing data visualization applications[44] solve this problem with native clients by locally rendering the data they dynamically fetch from the cloud.

Summary

Table 1 summarizes the features of the reviewed architectures. We focus on the five aspects that are most important for exploratory data analysis. The first one is support for fine-grained parallel execution. Concurrent execution of a single task on multiple CPUs can speed up the execution to the degree were interactive analysis of large data becomes feasible. The second one, interactive visualization, requires constant communication with the computational server. Even if computation is lengthy, the user has to be updated on the progress and, ideally, should see any intermediate results, our third aspect in the Table 1, to allow users to focus the analysis on the interesting subsets of the data. Notice that while Grids and Map Reduce provide speed-ups with concurrent execution, these architectures engage computational workers that do not directly communicate with the client and are less suitable for interactive exploration and communication of intermediate results.

The fourth aspect examines if the high-level execution environment can be coupled with the chosen architecture. Such environment should allow domain experts to analyze large amounts of data without specifically addressing the intricacies of parallel environments. The final, fifth aspect, is related to data security which is important in industrial or clinical settings or alike. We distinguish between architectures where data stays on a local premise or is spread across the network not necessary managed by the data owner. Grids and Map Reduce can support data locality only if the institution has the appropriate, often costly computational clusters.

Obviously, none of the current software architectures was designed with exploratory data analysis in mind, and hence none is ideal for its implementation. Yet, they all contain basic building blocks with which we could construct a suitable architecture. We briefly review these building blocks in the next chapter, and continue with a list of features that are specific to exploratory data analysis.

COMPONENTS OF DATA ANALYTICS SOFTWARE ARCHITECTURE

The five concurrent software architectures that can support data analysis, which we reviewed in the previous section all decouple data processing from the user interface. Regardless of their substantial differences, we may abstract their architecture with a few interconnected components (Figure 6). The user, who may be a field expert and not necessary prolific in computer science, uses a client to issue data analysis tasks to the rest

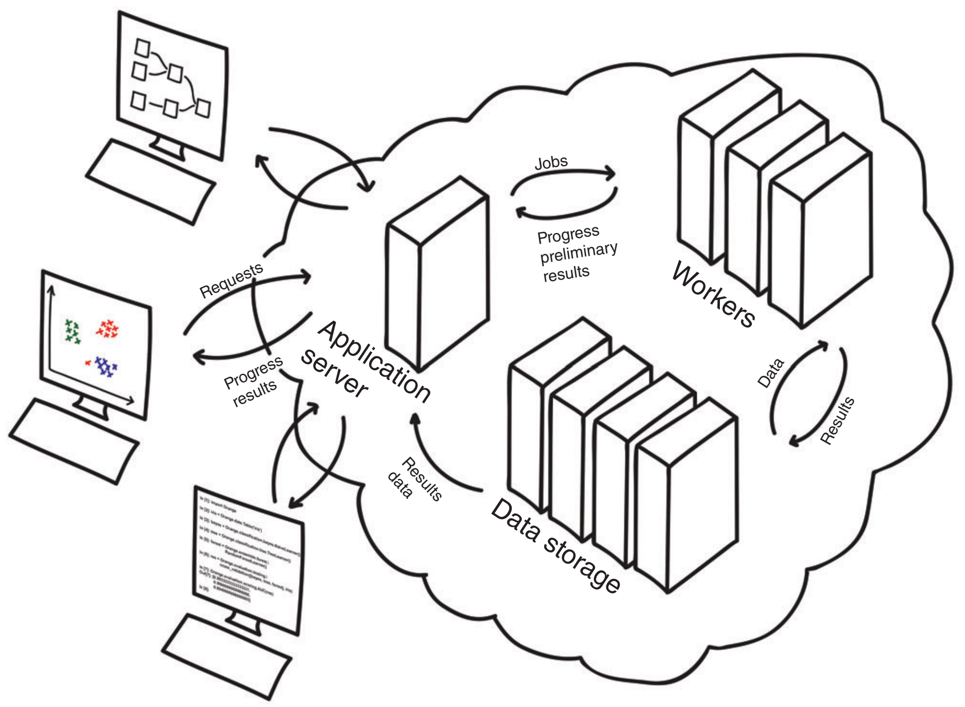


FIGURE 6 | General architecture for data analysis. Clients with user interface send data analysis requests to application servers that engage workers. These communicate with a data repository to load data and store the results of the analysis. Application servers render the results and communicate them back to the clients.

of the system. The client is either a standalone program or a web application that converts analysis requests to a set of instructions and presents the results to the user. The most common ways to design an analysis are workflows,[45] script-based programs or sets of statements in a query language.[46] The client should also be responsible for informing the user about the progress of the analysis and about any errors that occur during the execution of the analysis.

The responsiveness of the analysis system mostly depends on the computational backend. Often referred to as computational nodes, servers, or workers, they do the heavy lifting required to execute the analysis. Depending on the variety of tasks executed, workers may be of general-purpose or task-specialized.

General-purpose workers are able to execute any of the requested data analysis tasks. Their utilization depends on the data set locality and computational demands. In cloud-based setups, the number of general-purpose workers can be modified in order to meet the application load.

Specialized workers can only perform a limited number of tasks, but with greater efficiency than general-purpose workers. For instance, machines equipped with graphical processing units can efficiently execute some data analysis algorithms in parallel.[47,48] Similarly, clusters equipped with a high-speed InfiniBand network can significantly speed up execution of jobs using OpenMPI.[49]

ADVANCED FEATURES OF NEXT-GENERATION EXPLORATORY ANALYSIS SOFTWARE ENVIRONMENTS

In the Introduction, we have exposed the key components of exploratory data analysis systems: responsive user interfaces, interaction with analytics engine, graphical data displays, and dynamic workflows. Following we list several core functions that should be present in next-generation explorative data analysis systems and would distinguish these from the current desktop-based systems. We also highlight that some of this functionality have already been prototyped in existing data analysis software libraries, but it is their collective implementation within analytic systems that will present the biggest challenge to the evolution of concurrent data analysis architectures. We group the proposed features from the view-point of a user into those that address the speed-ups of the algorithms and data analysis procedures, support the smooth use of data analysis workflows, deal with procedures for data visualization, and support collaborative data exploration.

Concurrent Processing and Algorithms

An obvious gain from concurrent architectures for data analysis is greater speed. Users of toolboxes for explorative data analysis expect responsive interfaces and short execution times of underlying data analysis algorithms. These can be achieved by minimizing the data transfer between computational units, parallelization of data mining algorithms, and algorithmic prediction of the future user's requests to guide preemptive analysis and compute results for the most likely future queries.

Integrated Storage and Processing

Transmission of the data from storage to processing component of data analysis architecture is time consuming. Modern approaches store the data on the machines that also perform analysis, either by using a distributed file system[50,51] or sharding.[52] Such architectures scale well, as support for growing data volume is achieved by providing an increased number of computers. The downside of this architecture is that the location of the data files determines the nodes to process the data, which can be problematic when multiple users use the system simultaneously.

Potential solutions should consider meta information about storage and processing needs of the computational components of the workflow. These should be considered within task scheduler and load balancing algorithms. Load balancing strategies have been developed within grid computing[53] and agent-based systems,[54] and would need to be adapted for data mining tasks in which only rough estimates of processing time are available.

Parallelization of Standard Data Mining Tasks

Procedures such as scoring of the prediction systems (e.g., cross-validation), parameter estimation, or various optimization methods are standard data mining tasks.[55] Their parallel implementation is often straightforward and many of them are embarrassingly parallel. Some data mining software suites already support this kind of concurrency,[56] and its implementation should be present in any modern explorative data analysis system.

Algorithms based on iterative optimization are harder to parallelize, as each step of the procedure relies heavily on the previous steps. Parallel implementation of support vector machines[57] decomposes the model into a hierarchy of submodels. Submodel are trained in parallel and later joined to produce a single prediction model. Training of deep neural networks is still done sequentially, but takes advantage of specialized hardware (GPUs) to speed up the execution of each iteration. Other algorithms, such as mixed effects models are yet to be parallelized.

Preemptive Analysis

Explorative data analysis is most often based on descriptive statistics and their appropriate rendering. We request the computation of averages, standard deviations, and box plots for almost any examined data set. If computational resources are available, such statistics can be computed in advance, even before the user explicitly requests them. Preemptive execution of data analysis procedures can largely improve the perceived responsiveness of the system.

Several existing technologies already preprocess data to speed-up the analysis. In online analytical processing[58] (OLAP), transactional multidimensional data are transformed to OLAP cubes that are suitable for rapid filtering, grouping and computation of sums, averages, and other descriptive statistics. With appropriate design and intelligent workflow engines, other, more complex data analysis procedures may be executed in advance. Choosing which procedures to run will require predictions that reveal the most likely data analysis tasks to be executed according to the current pipeline, data, and user's choices when dealing with similar analysis problems in the past. For instance, loading a class-labeled data set could trigger a preemptive cross-validation with the user's favorite supervised data analysis methods that he previously applied to similar data sets. Preemptive analysis may have high computational costs and need to be tailored for particular users. The main challenge in this field is automatic identification of analysis tasks that require preemptive treatment, and automatic selection of types of preemptive analysis given the history of data analysis actions. The field could borrow predictive methods from recommender systems.[59]

Design and Use of Workflows

Workflows and their construction through visual programming are a perfect match for explorative data analysis. Workflows consist of computational units where users process, visualize, filter, and select data. The output of one computational unit is passed to another unit, in this way instructing the machine about the required analysis steps. Combined with interactive data visualizations, workflows can provide intuitive means to describe potentially complex data analysis procedures. Concurrent architectures can provide parallel implementations of workflows to speed-up analysis of larger data sets and data collections. Yet, these will still require implementation of procedures to report on the progress of analysis and inform the user by returning any intermediate results. For complex data analysis tasks and where computational components for data analysis are abundant, user may benefit from techniques that provide assistance in workflow construction and that propose the likely workflow extension or completion of analysis branches based on their current state and type of analyzed data.

Support for Fast Prototyping

Prototyping is an efficient way to assemble analysis pipelines and apply them to new data.[60] It allows an expert analyst to try out different methods and adapt them to the data at hand. Prototyping can be done in visual programming interfaces[3,4,60] or through scripting, possibly in the interactive console.[2] Each approach has its own merits. Visual interfaces are easier to learn, yet they usually limit the choice of available components. Scripting however enables expert users to finely tune their queries and change advanced parameters. PIG complements Java interface of Apache Hadoop with a high level language that can be used to combine existing map and reduce jobs, while Dremel uses queries similar to SQL to query underlying data.

Workflows define a collection and succession of data analysis tasks to be executed on a computational server. Workflow paths by their nature imply parallelism, making concurrent processing architectures ideal for this type of specification of tasks in explorative data analysis. Many current data analysis systems support workflow design, with no consensus on the ‘right’ granularity of tasks they implement and combine. Taverna, e.g., can embed either a simple string-processing routine or a complex genome analysis procedure with a single component. An open challenge is what granularity is best for the user of such systems.

Progress Tracking

Even with the fast implementation of data analysis algorithms and their concurrent execution, results of data analysis may not be available instantaneously. Execution tracking and estimation of remaining time are important for analyses that run for longer periods. Estimating the execution time is difficult, in particular for heuristic and iterative algorithms, which abound in data analysis.

Estimating execution of parallel queries in a general setup is a challenging problem. JAM architecture[9] displays status of execution on used agents. Pig/Hadoop’s progress estimator shows a percentage-remaining estimate under assumption that all operators take the same time to complete. Another solution for execution time estimation of MapReduce tasks is ParaTimer,[61] which provides better estimates by considering distribution over different nodes, concurrent execution, failures, and data skew.

Display of Intermediate Results

From the user’s viewpoint, rendering of intermediate results for some data analysis procedure is primarily the feature of the client’s interface. The main complexity of the implementation, however, lies within the architecture of the server. Here, the distribution of the tasks on the server has to include handling of requests for intermediate results and potential requests for abortion of the execution.

Time-consuming approaches in data mining, such as induction of classification trees in a large random forest or inference of deep neural networks,[62] are iterative. Partial or converging models may be meaningful even before the analysis is completed. For classification trees, nodes close to the root that are inferred first may already hold potentially interesting information. In a random forest, a smaller subset of trees may already provide useful predictions. Early stages of development of a deep neural network may be already sufficiently accurate. Application clients should thus be able to receive and render partial results of the data analysis as they arrive. Exposing them to the user lets him reconsider the course of analysis and potentially change it before the ongoing computation is completed. Display of intermediate results depends on the data analysis method, which would need to be appropriately adapted and implemented on the servers. Another challenge for the software architecture is also to provide means to request or push the intermediate results and report on the status of analysis.

Intelligent Data Analysis Interfaces

Analysis of complex and heterogeneous data sources requires nontrivial analysis pipelines. Future generation explorative data analysis frameworks will be able to suggest the components or analysis pathways based on the data characteristics, previous actions of the analysts, and typical analysis patterns of the broader community. Technologies for such an endeavor may stem from meta-learning,[63] social computing, and recommender systems.[59] Prediction of workflow components (Figure 7) is a recent area of study[64] that also borrows from techniques of network analysis.[65]

Data Visualization

Data visualization provides an essential mean of communication with the user of explorative data analysis

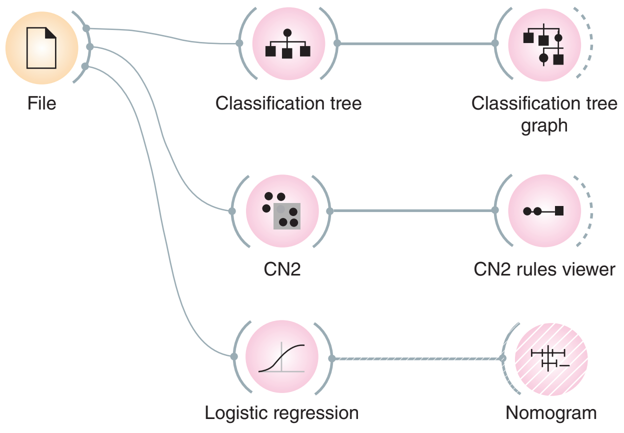


FIGURE 7 | Machine prediction of workflow components. The user constructs a workflow with two modeling algorithms (Classification Tree and a rule-based classifier CN2), each followed by components for model visualization (Classification Tree Graph and CN2 Rules Viewer). After he adds another modeling algorithm (Logistic Regression), the framework can anticipate that it will be followed by a visualization component (Nomogram) as well.

environment. For exploration, the visualizations need to support interactions, and to sustain the speed, the server has to be aware of the limitations of the data transfer. Instead of serving the entire data sets, data projects are computed and even partially rendered on a server. Concurrent algorithms on a server may also sip through the vast space of different projections to find those most informative and worthy to be displayed to the user.

Interactive Visualizations

Visualizations are an efficient way to gain insight into large data sets. Visual analytics[66] thus plays a very important role in exploratory data analysis. Visualization of large data sets is computationally intensive and cannot be performed on the client alone. Visualizations can be rendered on the server and transferred to the client in a form of a video stream.[67] In such a hybrid architecture, the server transforms the data into a representation of manageable size, which is transferred to the client and rendered there.[68] Visualizations on the client may be updated continuously as the stream of the processed data arrives from the server. Depending on the type of visualization and its processing needs, the data projection methods need to be adapted for parallel computation. An example for the later is an adaptation of multidimensional scaling and its parallel implementation.[55]

Intelligent Data Visualizations

When data includes many variables, the number of possible projections and data views is vast. Intelligent data visualization copes with this by engaging computational algorithms to propose the most informative visualizations. Various data mining methods already address this problem, although on a smaller scale. These include a dynamic walk through interesting visualizations (GrandTour[69]), or techniques such as VizRank that find and rank the most interesting projections[70] (Figure 8). Enumeration of interesting visualizations and search for paths to explore them is computationally expensive; different visualizations have to be explored through concurrent search and optimization. Processes that run on the server may depend on actions of the users through, say, choice of visualizations or choice of a data subset. This again requires a responsive software architecture that is able to run, abort, and reinitiate tasks on the server, based on the events received from the client site.

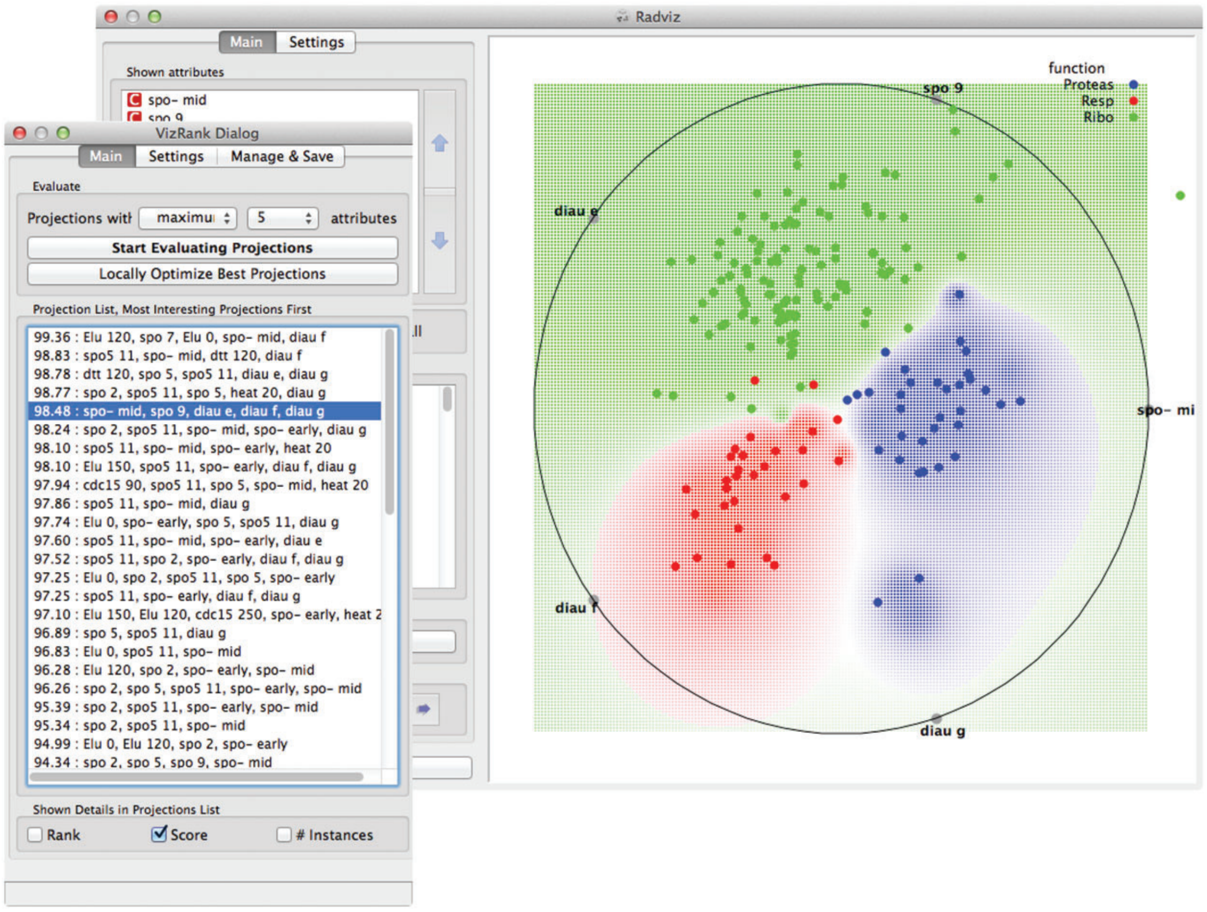
Another problem that could be solved with intelligent approaches is scaling the visual representations that work on small data sets to data of larger volume. Scatterplot, a popular visualization technique for small data sets, is useless for huge number of data points. For larger data sets, processes on the server would need to estimate the distributions from an informative data sample,[72] and communicate these to client to render a visualization of incomplete data set that potentially uncovers interesting patterns. It is the task of procedures for intelligent visualization to score the interestingness and, through concurrent processing, find data subsets that maximizes this score.[73]

Collaboration

Analysis of real-life data is often a complex process spanning across different areas of expertise. Data analysis projects are becoming collaborative efforts. Collaboration interfaces should borrow from the ideas of social and gaming platforms. Collaborative data exploration has to be mapped to concurrent execution of data analytics tasks to avoid redundancies in data processing and offer the speed that stems from sharing the results that were rendered for one user to his partners. Collaborative explorative data analysis environments are in its infancy[74] and we have yet to gain experience in this field to propose efficient concurrent solutions.

CONCLUSIONS

Of the software architectures we have reviewed, it is their combination that will most likely be present in the future systems for explorative data analysis. The key components will likely be implemented within computer grids, perhaps augmented with specialized hardware for large-scale parallel execution. Current



```
99.36 : Elu 120, spo 7, Elu 0, spo- mid, diau f
98.83 : spo5 11, spo- mid, dtt 120, diau f
98.78 : dtt 120, spo 5, spo5 11, diau e, diau g
98.77 : spo 2, spo5 11, spo 5, heat 20, diau g
98.48 : spo- mid, spo 9, diau e, diau f, diau g
98.24 : spo 2, spo5 11, spo- mid, spo- early, diau g
98.10 : spo5 11, spo- mid, spo- early, heat 20
98.10 : Elu 150, spo5 11, spo- early, diau f, diau g
97.94 : cdc15 90, spo5 11, spo 5, spo- mid, heat 20
97.86 : spo5 11, spo- mid, diau g
97.74 : Elu 0, spo- early, spo 5, spo5 11, diau g
97.60 : spo5 11, spo- mid, spo- early, diau e
97.52 : spo5 11, spo 2, spo- early, diau f, diau g
97.25 : Elu 0, spo 2, spo5 11, spo 5, spo- early
97.25 : spo5 11, spo- early, diau f, diau g
97.10 : Elu 150, Elu 120, cdc15 250, spo- early, heat 2
96.89 : spo 5, spo5 11, diau g
96.83 : Elu 0, spo5 11, spo- mid
96.28 : Elu 120, spo 2, spo- early, spo- mid
96.26 : spo 2, spo 5, spo5 11, spo- early, spo- mid
95.39 : spo 2, spo5 11, spo- early, spo- mid
95.34 : spo 2, spo5 11, spo- mid
94.99 : Elu 0, Elu 120, spo 2, spo- early
94.34 : spo 2, spo 5, spo 9, spo- mid
```

FIGURE 8 | Intelligent data visualization by VizRank. The window on the right shows Radviz visualization of gene expression profiles from Ref. 71. Vizrank can score Radviz data projections by degree of separation between data items of different classes and offer them in browsable ranked list (window on the left).

architectures would need much improvement, however, to accommodate the advanced tasks that will be supported by the next generation of concurrent explorative data analysis architectures. They will need to address a decrease of latency, continuous streaming of results, and improvements at the client site and user interfaces. Latency could be reduced by incremental analysis of the data that would offer intermediate results prior to completion of the analysis on the entire data set, or even prior to the user's request for the analysis. Software architectures should stream graphical results, as data visualization is one of the key components of exploratory data analysis. Workflow interfaces on the client side need to be adapted for dynamically changing environments, where the design and execution are interwoven and intermediate results impact the analysis choices of the user.

Future systems for data analytics will predict the course of analysis and guide users in the design of the analytics pipeline. Computational servers will anticipate the next steps of analysis and execute some selected procedures in advance by balancing between the cost of computational resources and response time requirements. Collaborative analysis will add an extra dimension to this problem and will require an additional optimization layer to minimize computation by jointly serving the needs of the community of users.

Ever growing volume of data will greatly impact the design computer systems for exploratory data analysis. Existing software architectures are no longer adequate; we need to adapt them and invent new ones. But as this review shows, the process can be gradual and based on excellent foundations of concurrent processing and software architectures developed throughout a wide scope of information technologies.

REFERENCES

1. Jones LVE. *The Collected Works of John W. Tukey: Philosophy and Principles of Data Analysis 1949–1964*, vol. III & IV. Chapman & Hall; 1986, London.
2. Ihaka R, Gentleman R. R: a language for data analysis and graphics. *J Comput Graph Stat* 1996, 5:299–314.
3. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Sieb C, Thiel K, Wiswedel B. KNIME: the Konstanz information miner. *ACM SIGKDD Explor Newslett* 2009, 11:26–31.
4. Curk T, Demšar J, Xu Q, Leban G, Petrovič U, Bratko I, Shaulsky G, Zupan B. Microarray data mining with visual programming. *Bioinformatics* 2005, 21:396–398.
5. Rakotomalala R. TANAGRA: un logiciel gratuit pour l'enseignement et la recherché. In: *Actes de EGC'2005, RNTI-E-3*, Paris, France, 2005, 697–702.
6. Keim DA, Mansmann F, Schneidewind J, Ziegler H. Challenges in visual data analysis. In: *Proceedings of the IEEE Tenth International Conference on Information Visualisation,* 2006, 9–16.
7. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. *Commun ACM* 2008, 51:107–113.
8. Ekanayake J, Pallickara S, Fox G. MapReduce for data intensive scientific analyses. In: *Proceedings of the 4th IEEE International Conference on eScience*, Indianapolis, Indiana, US, 2008, 277–284.
9. Stolfo S, Prodromidis A, Tselepis S, Lee W. JAM: Java agents for meta-learning over distributed databases. In: *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, US, 1997, 74–81.
10. Bailey S, Grossman R, Sivakumar H, Turinsky A. Papyrus: a system for data mining over local and wide area clusters and super-clusters. In: *Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, ACM, Portland, Oregon, US, 1999, 63.
11. Kargupta H, Hamzaoglu I, Stafford B. Scalable, distributed data mining—an agent architecture. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California, US, 1997, 211–214.
12. Džeroski S, Ženko B. Is combining classifiers with stacking better than selecting the best one? *Mach Learn* 2004, 54:255–273.
13. Štrumbelj E, Kononenko I. An efficient explanation of individual classifications using game theory. *J Mach Learn Res* 2010, 11:1–18.
14. Papazoglou M. *Web Services: Principles and Technology*. Upper Saddle River, New Jersey: Prentice Hall; 2008.
15. Talia D, Trunfio P, Verta O. The Weka4WS framework for distributed data mining in service-oriented grids. *Concur Comput Pract Exp* 2008, 20:1933–1951.
16. Hall M, Eibe F, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. *ACM SIGKDD Explor Newslett* 2009, 11:10–18.
17. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, Oinn T. Taverna: a tool for building and running workflows of services. *Nucleic Acids Res* 2006, 34:29–32.
18. Podpečan V, Zemenova M, Lavrač N. Orange4WS environment for service-oriented data mining. *Comput J* 2011, 55:82–98.
19. Zakova M, Kremen P, Zelezny F, Lavrač N. Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Trans Autom Sci Eng* 2011, 8:253–264.
20. Raicu I, Foster I, Szalay A, Turcu G. Astroportal: a science gateway for large-scale astronomy data analysis. In: *Proceedings of the TeraGrid Conference*, Indianapolis, Indiana, US, 2006, 12–15.
21. Kalderimis A, Lyne R, Butano D, Contrino S, Lyne M, Heimbach J, Hu F, Smith R, Štěpán R, Sullivan J, et al. InterMine: extensive web services for modern biology. *Nucleic Acids Res* 2014, 42:468–472.
22. Kiss R, Sandor M, Szalai F. http://Mcule.com: a public web service for drug discovery. *J Cheminform* 2012, 4(Suppl 1):17–17.
23. Korkontzelos I, Mu T, Ananiadou S. ASCOT: a text mining-based web-service for efficient search and assisted creation of clinical trials. *BMC Med Inform Decis Mak* 2012, 12(Suppl 1):1–12.
24. Foster I. The anatomy of the grid: enabling scalable virtual organizations. *Int J High Perform Comput Appl* 2001, 15:200–222.
25. AlSairafi S, Emmanouil FS, Ghanem M, Giannadakis N, Guo Y, Kalaitzopoulos D, Osmond M, Rowe A, Syed J, Wendel P. The design of discovery net: towards open grid services for knowledge discovery. *Int J High Perform Comput Appl* 2003, 17:297–315.
26. Stankovski V, Swain M, Kravtsov V, Niessen T, Wegener D, Kindermann J, Dubitzky W. Grid-enabling data mining applications with DataMiningGrid: an architectural perspective. *Future Gener Comput Syst* 2008, 24:259–279.
27. Taylor I, Shields M, Wang I, Harrison A. The Triana workflow environment: architecture and applications. In: *Workflows for e-Science*. Heidelberg: Springer; 2007, 320–339.
28. Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the condor experience. *Concur Comput Pract Exp* 2005, 17:323–356.
29. Nordberg H, Bhatia K, Wang K, Wang Z. BioPig: a Hadoop-based analytic toolkit for large-scale sequence data. *Bioinformatics* 2013, 29:3014–3019.
30. Schumacher A, Pireddu L, Niemenmaa M, Kallio A, Korpelainen E, Zanetti G, Heljanko K. SeqPig: simple

and scalable scripting for large sequencing data sets in Hadoop. *Bioinformatics* 2014, 30:119–120.

31. Kang U, Tsourakakis CE, Faloutsos C. Piranha:Pegasus: mining peta-scale graphs. *Knowl Inf Syst* 2011, 27:303–325.
32. Russell J. *Getting Started with Impala: Interactive SQL for Apache Hadoop*. O'Reilly Media, Inc.; 2014, Sebastopol, California, US.
33. Prajapati V. *Big Data Analytics with R and Hadoop*. Packt Publishing; 2013, Birmingham, UK.
34. Yejas ODL, Zhuang W, Pannu A. Big R: large-scale analytics on Hadoop using R. In: *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Anchorage, Alaska, US, 2014, 570–577.
35. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: a distributed storage system for structured data. *ACM Trans Comput Syst* 2008, 1–26:4.
36. Melnik S, Gubarev A, Long JJ, Romer G, Shivakumar S, Tolton M, Vassilakis T. Dremel: interactive analysis of web-scale datasets. In: *Proceedings of the 36th International Conference on Very Large Data Bases*, Singapore, 2010, 330–339.
37. Zaharia M, Chowdhury M, Franklin M, Shenker S, Stoica I. Spark: cluster computing with working sets. In: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, USENIX Association, Boston, MA, US, 2010, 10.
38. Lin H, Yang S, Midkiff SP. RABID: a distributed parallel R for large datasets. In: *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Anchorage, Alaska, US, 2014, 725–732.
39. Zhou X, Petrovic M, Eskridge T, Carvhalo M. Exploring Netfow data using Hadoop. In: *Proceedings of the Second ASE International Conference on Big Data Science and Computing*, Academy of Science and Engineering (ASE), Stanford, CA, US, 2014.
40. Ren K, Kwon Y, Balazinska M, Howe B. Hadoop's adolescence: an analysis of Hadoop usage in scientific workloads. *Proc VLDB Endowment* 2013, 6:853–864.
41. Elmeleegy K. Piranha: optimizing short jobs in hadoop. *Proc VLDB Endowment* 2013, 6:985–996.
42. Wang Z, Chen C, Zhou J, Liao J, Chen W, Maciejewski R. A novel visual analytics approach for clustering large-scale social data. In: *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, Santa Clara Marriott, CA, USA, 2013, 79–86.
43. Vaquero LM, Rodero-Merino L, Buyya R. Dynamically scaling applications in the cloud. *ACM SIGCOMM Comput Commun Rev* 2011, 41:45–52.
44. Chen K, Xu H, Tian F, Guo S. Cloudvista: visual cluster exploration for extreme scale data in the cloud. In: *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*, Portland, OR, US, 2011, 332–350.
45. Deelman E, Gannon D, Shields M, Taylor I. Workflows and e-science: an overview of workflow system features and capabilities. *Future Gener Comput Syst* 2009, 25:528–540.
46. Džeroski S, Lavrač N. *Relational Data Mining*. Berlin: Springer; 2001.
47. Lowe EW, Butkiewicz M, Woetzel N and Meiler J. GPU-accelerated machine learning techniques enable QSAR modeling of large HTS data. In: *Proceedings of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, San Diego, CA, US, 2012, 314–320.
48. Sharp T. Implementing decision trees and forests on a GPU. In: *Proceedings of the 10th European Conference on Computer Vision*, Marseille, France, 2008, 595–608.
49. Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, Sahay V, Kambadur P, Barrett B, Lumsdaine A et al. Open MPI: goals, concept, and design of a next generation MPI implementation. In: *Proceedings of the 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, 2004, 97–104.
50. Ghemawat S, Gobioff H, Leung ST. The Google file system. *ACM SIGOPS Oper Syst Rev* 2003, 37:29–43.
51. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop distributed file system. In: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Lake Tahoe, Nevada, US, 2010, 1–10.
52. Cattell R. Scalable SQL and NoSQL data stores. *ACM SIGMOD Rec* 2011, 39:12–27.
53. Qin X. Design and analysis of a load balancing strategy in data grids. *Future Gener Comput Syst* 2007, 23:132–137.
54. Mary MEL, Saravanan V. Predictive load balancing for data mining in distributed systems. *J Theor Appl Inf Technol* 2013, 53:13–23.
55. Bae SH, Choi JY, Qiu J, Fox GC. Dimension reduction and visualization of large high-dimensional data via interpolation. In: *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ACM, Chicago, Illinois, US, 2010, 203.
56. Bekkerman R, Bilenko M, Langford J. *Scaling up machine learning: parallel and distributed approaches*. Cambridge: Cambridge University Press; 2012.
57. Sun Z, Fox G. Study on parallel SVM based on MapReduce. In: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, CSREA Press, Las Vegas, Nevada, US, 2012, 16–19.
58. Chaudhuri S, Dayal U. An overview of data warehousing and olap technology. *ACM SIGMOD Rec* 1997, 26:65–74.
59. Tavakolifard M, Almeroth K. Social computing: an intersection of recommender systems, trust/reputation systems, and social networks. *IEEE Network* 2012, 26:53–58.

60. Mierswa I, Wurst M, Klinkenberg R. YALE: rapid prototyping for complex data mining tasks. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Philadelphia, PA, US, 2006, 935–940.

61. Morton K, Balazinska M and Grossman D. Paratimer: a progress indicator for mapreduce dags. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ACM, Indianapolis, Indiana, US, 2010, 507–518.

62. Murphy KP. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: The MIT Press; 2012.

63. Prodromidis A, Chan P and Stolfo S. Meta-learning in distributed data mining systems: issues and approaches. *Advances in Distributed and Parallel Knowledge Discovery*, AAAI Press/MIT Press, 2000, 81–114.

64. Polajnar M, Demšar J. Small network completion using frequent subnetworks. *Intell Data Anal* 2015, 19:89–108.

65. Newman M. *Networks: An Introduction*. Oxford, UK: Oxford University Press; 2010.

66. Keim D, Andrienko G, Fekete JD, Carsten G. *Visual Analytics: Definition, Process, and Challenges*. Berlin, Germany: Springer-Verlag; 2008.

67. Lamberti F, Sanna A. A streaming-based solution for remote visualization of 3D graphics on mobile devices. *IEEE Trans Vis Comput Graph* 2007, 13: 247–260.

68. Ahrens J, Woodring J, DeMarle D, Patchett J, Maltrud M. Interactive remote large-scale data visualization via prioritized multi-resolution streaming. In: *Proceedings of the Workshop on Ultrascale Visualization*, ACM, Portland, OR, US, 2009, 1–10.

69. Asimov D. The grand tour: a tool for viewing multidimensional data. *SIAM J Sci Stat Comput* 1985, 6:128–143.

70. Leban G, Zupan B, Vidmar G, Bratko I. Vizrank: data visualization guided by machine learning. *Data Min Knowl Discov* 2006, 13:119–136.

71. Brown MP, Grundy WN, Lin D, Cristianini N, Sugnet CW, Furey TS, Ares M, Haussler D. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc Natl Acad Sci USA* 2000, 97:262–267.

72. Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica I. BlinkDB: queries with bounded errors and bounded response times on very large data. In: *Proceedings of the 8th ACM European Conference on Computer Systems*, ACM, Prague, Czech Republic, 2013, 29–42.

73. Ma KL. Machine learning to boost the next generation of visualization technology. *IEEE Comput Graph Appl* 2007, 27:6–9.

74. Willett W, Heer J, Hellerstein J, Agrawala M. CommentSpace: structured support for collaborative visual analysis. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, BC, Canada, 2011, 3131–3140.

FURTHER READINGS

Heer J, Kandel S. Interactive analysis of big data. *XRDS* 2012, 19:50–54.

Witten IH, Frank E. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc.; 2005, Burlington, Massachusetts.

Ward MO, Grinstein G, Keim D. Interactive Data Visualization: Foundations, Techniques, and Applications (2nd Edition), A K Peters/CRC Press, 2015.