Executive Summary

How much information is there in your data? How much is hidden from you, because you don’t have access to the right analysis tools? What kind of insights are you trying to extract from your data? For example:

Do you need to predict a result, based on the values of other variables?
Are you trying to understand which collection of variables gives you the most predictive power?
Is there some measure of performance that you’re trying to optimize?

Other companies are driving business planning and decision making by using analysis techniques which answer questions such as these, and which have been useful in transforming their data into valuable information. However, such techniques can be difficult to develop from scratch for in-house analysis applications, whilst the use of open source packages can be unreliable in a world of varying quality. An alternative approach is to use a commercially available implementation of analysis methods which can be effortlessly incorporated into your in-house code. Developers are thereby freed up to concentrate on adding value to your application, whilst the numerical heavy-lifting and algorithmic nuts and bolts are provided by experts in the field of numerical algorithms. This white paper describes how application developers are using various methods to incorporate powerful analytics functionality into their code, thereby providing a competitive advantage to their clients who are using their data to discover valuable information.

Introduction

Personalization, mobility, and the rapidly growing use of sensor technology are generating raw data at previously unimaginable rates. This so-called “big data” contains valuable insights about business processes and customer behaviors, but the full value of this data is all too often untapped. Transforming data into meaningful insight requires the expert utilization of sophisticated analytical tools – including techniques such as optimization, correlation and regression analysis, time series analysis and data mining – which can be a daunting task.

Regarding current trends for the use of such techniques, authors Davenport and Harris comment in their seminal publication “Competing on Analytics”:

“Virtually every major company uses some form of statistical or mathematical analysis, but some take analytics much further than others. In our research on the topic, we have identified several key attributes of firms that compete on analytics, [including the] widespread use of not just descriptive statistics, but predictive modeling and complex optimization techniques.”

According to a report from the International Data Corporation, the worldwide market for business analytics software is expected to grow to $33.9 billion in 2012. As the demand for advanced analytics continues to expand, more and more software vendors are searching for ways to enhance capabilities and remain competitive with market leaders. By embedding powerful “analytic engines”, leading software vendors are now extending the analytical capability of their applications to deliver unique value to customers and achieve differentiation from their competition.
A key decision for software vendors that want to make analytics part of their application is how to embed the mathematics and statistics that enables analytics. Three alternatives are discussed in the next section.

Options for embedding Analytics

There are a variety of alternative approaches for software vendors who are seeking to add analytic functionality to their applications. The three options which are commonly considered represent a slight variation on the well-worn theme of “build, borrow, or buy”:

- **Build** and integrate analytics functions using in-house development resources
- **Borrow** analytic functionality by leveraging and integrating available open source code
- **Buy** a commercially available integrated solution or components

Each option has advantages and disadvantages, which will be explored in the following sections.

Build – and integrate in-house

The first option which comes to mind for many software vendors is the in-house development of analytics functionality. This is an obvious choice because it leverages the skills of their employees who are adept programmers that already know how to develop code for their applications. The technical literature (for example, the Numerical Recipes series of books) contains detailed descriptions of analytics algorithms which can be to solve problems in a wide range of fields; in principle, developers can then translate the mathematics into an implementation of the algorithm written in their programming language of choice, and then incorporate that into their application.
However, using the specification of an algorithm in a journal article or textbook to write an implementation from scratch can be a considerable challenge for software developers if the goal is to create code which is numerically robust, computationally efficient and commercially viable. Moreover, development of this kind is typically serial in nature, and so adding more resource will not shorten the development cycle. If time-to-market is a consideration for the vendor, then the effort required to develop in-house analytics solutions may prove to have a high cost associated with it.

Looking at the option of in-house development in more detail, it is clear that both direct and ongoing costs must be borne in mind. Firstly, there is the initial, direct cost of the development time to write, test and document the algorithm. Although this will be low for sufficiently simple problems, development will take longer for complex algorithms or custom solutions: months, or even years in some cases. Secondly, there is the ongoing cost of maintaining algorithms; this includes upgrading code for new hardware or software platforms, migrating code to new environments where required, and updating the code to incorporate new features as needed. The cost of maintenance can be many times more than that of the initial development process.

Besides bearing in mind the costs of in-house development, it is important to consider whether assigning internal resources to the development of analytical solutions (in addition to the working on all other aspects of the vendor’s core application) represents the most appropriate allocation of effort. Since the main task of the internal development team is to add value to the core application – often by making use of domain-specific knowledge which is unique to the vendor – and that mathematical and statistical algorithms can be leveraged from elsewhere (see below), it could be argued that it doesn’t.

**Borrow – from open source analytics**

In an effort to avoid the costs associated with in-house development of algorithms, some software vendors have investigated one of the many open source systems, of which the **GNU Scientific Library** is but one example. The most commonly cited advantages of open source are that its source code is available and its development proceeds in a public, collaborative fashion.

Whilst there are many examples of successful and useful open source projects, software vendors who are considering the use of this option to add analytics to their applications should also bear the risks associated with it in mind. For example, the license associated with many open source systems requires that the source code for any application that makes use of that system must also be made available, which is not a viable option for many vendors. The use of other open source systems which incorporate third-party obligations will be unacceptable to some customers, who will not accept applications that contain open source code, because they do not want to deal with multiple companies and organizations for support.

In addition, it should be pointed out that not all open source projects are easy to use from the development point of view. Some are only sparsely documented (occasionally in foreign languages) and contain few, if any, examples. Although the availability of source is viewed as an advantage in some circumstances, it sometimes means (if source is the only thing that’s distributed in the project) that the user is obliged to compile the necessary binaries themselves. This can be daunting if the developer wants to use the system on a new platform (i.e. one for which it hasn’t yet been implemented) – for example, if the project has been designed for Linux, but the
developers can then invoke the appropriate method at the point in the application where analysis is required. They are thus freed up to concentrate on adding domain-specific value to the application, without having to worry about the algorithmic nuts and bolts which are being provided by experts in the field. Using a commercial solution empowers users to add analytical functionality to their application without having to re-invent the wheel by writing their own versions of complicated mathematical or statistical routines, or introducing risk by using untested, unsupported open source software in their application.

Using a commercial solution has other advantages also. The third party will have a focus on developing, testing, documenting, maintaining and selling analytical algorithms for a wide range of languages and platforms, and will be experienced at working with thousands of customers around the globe. The vendor who chooses the commercial option can immediately make use of all of this expertise.

Finally, it has been found that open source projects often stagnate after an initial burst of activity. This is usually because there is no impetus for continued development once the original developers have got the software working for the solution of their specific problem.

**Buy – from commercial suppliers**

The third option is to use a commercial collection of analytics methods that have been developed by a third party – preferably one that is expert in the field of numerical analysis. Such a collection of methods is most conveniently made available in the form of a software library which can be linked in with the vendor’s application at build time; developers can then invoke the appropriate method at the point in the application where analysis is required. They are thus freed up to concentrate on adding domain-specific value to the application, without having to worry about the algorithmic nuts and bolts which are being provided by experts in the field. Using a commercial solution empowers users to add analytical functionality to their application without having to re-invent the wheel by writing their own versions of complicated mathematical or statistical routines, or introducing risk by using untested, unsupported open source software in their application.

Using a commercial solution has other advantages also. The third party will have a focus on developing, testing, documenting, maintaining and selling analytical algorithms for a wide range of languages and platforms, and will be experienced at working with thousands of customers around the globe. The vendor who chooses the commercial option can immediately make use of all of this expertise.

<table>
<thead>
<tr>
<th>Embedded Analytic Approaches</th>
<th>Key Advantages</th>
<th>Key Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Build – and integrate in-house</strong></td>
<td>Analytic functionality developed is owned by the software vendor</td>
<td>Investment in time to write, test, document and maintain algorithms</td>
</tr>
<tr>
<td></td>
<td>Code is free to modify and enhance</td>
<td>Diversion of developer’s effort from core application development</td>
</tr>
<tr>
<td><strong>Borrow – from open source analytics</strong></td>
<td>Many algorithms are freely available</td>
<td>Third-party obligations and viral-type license agreements may be unacceptable to customers</td>
</tr>
<tr>
<td></td>
<td>Less development time than in house option</td>
<td>Documentation, testing and support are variable</td>
</tr>
<tr>
<td></td>
<td>Developer freed up to concentrate on core application</td>
<td></td>
</tr>
<tr>
<td><strong>Buy – from commercial suppliers</strong></td>
<td>Depth and breadth of available algorithms</td>
<td>Cost to license</td>
</tr>
<tr>
<td></td>
<td>Reduced in-house development time and risk through use of tested, supported and widely used algorithms</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Leveraging of commercial algorithmic expertise</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Developer freed up to concentrate on core application</td>
<td></td>
</tr>
</tbody>
</table>

This table summarizes the key advantages and disadvantages of these alternatives for embedding analytics into a software application.
Introduction to the NAG Library

With more than forty years’ experience in the development of algorithms in numerics, statistics and analysis, The Numerical Algorithms Group (NAG) is one of many vendors who offer a commercial library option for embedding analytics. The NAG Library enjoys an unrivalled reputation for quality, accuracy and performance, and its methods are available from many computing environments, including standard languages such as C, C++, Java, C#, Python and Fortran, as well as packages like R, Excel and MATLAB. Its analytics algorithms represent a wide range of functionality, have been subject to extensive and thorough testing in a variety of environments, and are extensively documented to facilitate their use by developers. In addition, many of them have been specially tuned for use on multicore systems, and have found use in a variety of HPC applications, where they are routinely used in the generation and analysis of big data.

The NAG Library currently contains more than 1,600 methods for solving mathematical and statistical problems, which can be embedded in any type of application by developers. The table below presents the areas covered by the NAG Library.

<table>
<thead>
<tr>
<th>Mathematics</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Root Finding</td>
<td>· Simple Statistics</td>
</tr>
<tr>
<td>· Summation of Series</td>
<td>· Correlation &amp; Regression Analysis</td>
</tr>
<tr>
<td>· Quadrature</td>
<td>· Multivariate Methods</td>
</tr>
<tr>
<td>· Ordinary Differential Equations</td>
<td>· Analysis of Variance</td>
</tr>
<tr>
<td>· Partial Differential Equations</td>
<td>· Random Number Generators</td>
</tr>
<tr>
<td>· Numerical Differentiation</td>
<td>· Univariate Estimation</td>
</tr>
<tr>
<td>· Integral Equations</td>
<td>· Nonparametric Statistics</td>
</tr>
<tr>
<td>· Mesh Generation</td>
<td>· Smoothing in Statistics</td>
</tr>
<tr>
<td>· Interpolation &amp; Approximation</td>
<td>· Contingency Table Analysis</td>
</tr>
<tr>
<td>· Curve and Surface Fitting</td>
<td>· Survival Analysis</td>
</tr>
<tr>
<td>· Optimization</td>
<td>· Time Series Analysis</td>
</tr>
<tr>
<td>· Special Function Approximation</td>
<td>· Operations Research</td>
</tr>
<tr>
<td>· Linear Algebra</td>
<td>· Data Mining</td>
</tr>
</tbody>
</table>

The specific methods that will be chosen by the developer depend on the requirement of the application. The following sections highlight some examples of recent use of Library methods by NAG customers to solve specific problems in business analytics. The following case studies describe various ways software vendors have leveraged NAG’s commercially available library of mathematical and statistical components to successfully embed analytics into their software applications.
Case studies: NAG and analytics

Modeling customer survey returns

Many companies need to use analytic functionality to extract meaningful information from their data. For example, one NAG user is modeling returns from a customer survey using NAG’s data mining methods. More specifically, their application builds a predictive model to help improve returns from the survey, offering a variety of learning algorithms (decision tree, logistic regression, support vector machine, etc.) to use in the model. The application reads in the survey results and randomly partitions them into training and validation sets. Supervised learning is then employed: this runs the learning algorithm on training set, and then optimizes its performance on the validation set. The improvement gained by using the predictive model is displayed graphically (see figure).

General purpose business analytics

Another NAG user is employing their methods in their business analytics application, which is looking at offering answers to various questions that might be asked by clients. Some of these are listed below, along with the NAG method that is being used to answer them:

- Why do people spend more for car insurance?
  NAG method: Stepwise linear regression

- Are stock prices correlated together?
  NAG method: Correlation matrix

- Who are your customers (and who aren't)?
  NAG method: Generalized linear model with binomial errors

- Can data be transformed from a standard representation to cumulative normal?
  NAG method: Special functions

The data sets that are analyzed by the user’s application come from various sources, including statistics on municipalities, intra-day stock prices and customer behavior profiles.
Other example applications for NAG and analytics

There are several other analytics application areas where NAG's methods are relevant, including:

- retail and consumer: forecasting, stock optimization and data fitting
- marketing: surveys, campaign optimization, marketing mix models
- pharmaceutical: forecast simulations and lifecycle management
- logistics: routing, loading and manpower
- finance: credit score, marketing, operations and financial modeling
- telecoms: traffic analysis, tariff models, technology roll-out optimization
- manufacturing: supply chain, stock and production control
- health: theatre capacity, epidemic planning and behavior simulations
- government: deployment of services, pollution, air safety, and criminal justice policy
- transport: scheduling, ticketing, and planning

Conclusions

By embedding analytic functionality into their applications, software vendors can help their customers and users to extract new insights from their data, solve business problems and make informed decisions. Whilst building in-house solutions could be viewed as diverting resources from core application development, and borrowing from open source distributions raises issues of quality, using a commercial option provides an optimum solution for many vendors because of the way it leverages the supplier’s experience in developing, testing, documenting and supporting analytical algorithms. Vendors who incorporate commercial analytics into their applications can help their customers and users to extract new insights from their data, solve business problems and make informed decisions.
applications free up their developers to concentrate on adding domain-specific value, whilst the analytical heavy lifting is being performed by someone else who can be relied upon to provide a more dependable solution than that offered by open source.

NAG has over forty years’ experience in developing algorithms for numerics, statistics and analysis, which have enjoyed an unrivalled reputation for quality, accuracy and performance. NAG methods offer a “cure for the common code” which can be invoked from many computing environments, and many of them have been optimized for use in HPC applications. Embedding NAG analytics in an application enables it to be used to transform business data into valuable information, thereby providing a competitive edge to the user.