

JAVA — The Next Generation of Statistical Computing?

Martin Theus
Information Science Research
AT&T Labs
Florham Park, NJ, 07932

Abstract

Statistical computing started with the implementation of simple FORTRAN, APL etc. routines. Those routines often run on specific hardware and used specific data-formats.

Packages like S started to gather routines as well as data in a common environment. Researchers also have been able to add their own functions and provide them to other users of S. But still package and data were tied to specific platforms.

Given JAVA, any routine runs on any computer supporting JAVA. Data as well as methods can be accessed remotely. The remote data-access even includes the direct access of databases. JAVA's graphical user interface offers the same functionality on every platform.

This paper will investigate past and recent developments in the area of statistical computing in order to judge the possible influence of the upcoming JAVA-technology.

1 Introduction

This paper does not intend to give a precise forecast of what statistical computing will look like in the future. The intention is more to call to mind the different stages of statistical computing in the light of a new powerful technology — JAVA. Thus we will look at various phenomena and try to understand their importance, their limitations and their lastingness. Based on this insight we then can judge what impact JAVA will/can have on the further development of statistical computing.

2 Looking back to see the Future

This section provides a brief outline of the history of statistical computing, developed along three threads. The threads in figure 1 are chosen as:

- Data Storage

- Programming Languages
- Interaction (with the computer-system)

There is no time-scale provided, since the actual times when certain technologies were available for the different communities of users may vary. Although figure 1

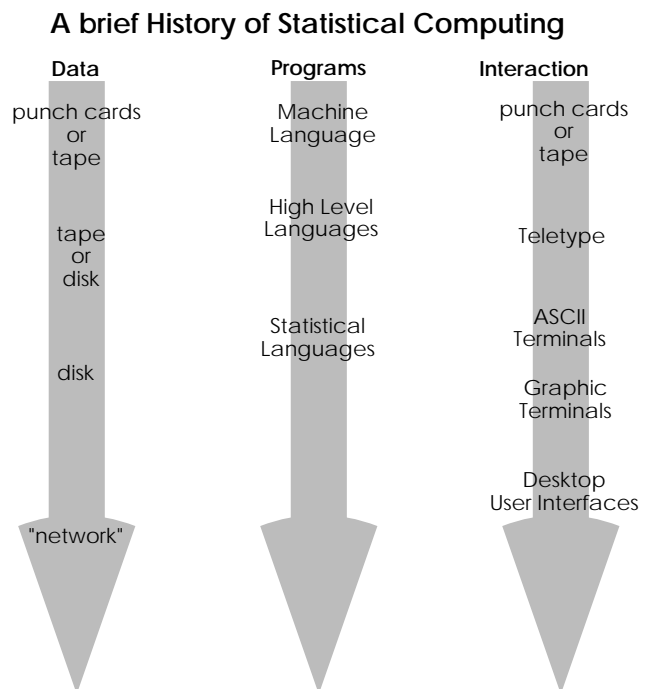


Figure 1: A brief history of statistical computing along three threads

represents the whole spectrum of progress in software and hardware technologies, it does not reflect the way statisticians think of statistical computing. Given a lag of about ten years between the introduction of new tools and technologies by computer scientists — as claimed by Becker (1989) — statisticians are endangered to not always use the appropriate tools and techniques in implementing solutions for statistical problems. Although

this lag might get smaller in terms of time it certainly is not getting smaller in terms of technology.

2.1 Kinds of Software

Roughly following the categorization of Becker (1989) we can distinguish about four kinds of statistical software:

- **Special Purpose Stand-alone Application**
This class comprises a huge variety of applications, which includes e.g. even UNIX-tools like `sort` or `uniq`. No matter which technologies are available, there will always be a need of these, mostly small, applications which are either of broad use and simple functionality (like `sort`), or have a very special purpose and thus a narrow use.
- **Collection of Subroutines**
Collections of subroutines are often the building blocks for statistical applications. In times of high popularity of FORTRAN, libraries like LINPACK and ICEPACK have been essential to statistical computing.
The S-language, today a broad statistical programming environment, evolved out of a loose collection of statistical subroutines.
- **Special Purpose Packages (closed)**
In contrast to the first group mentioned, this category includes closed software-packages which are not programmable, but yet offer a *variety* of methods in a certain field.
Almost all packages, which offer high interactions with plots and data through a graphical user interface (GUI) fall into this category.
- **General Purpose Packages (open)**
Only a few systems have been designed to offer a very general support in solving statistical problems. The best know member of this category is S/S-Plus. Other general purpose packages are ISP (more popular in the 80s) or XploRe. Packages like SAS are extensible as well, but have their strength more in already implemented statistical methods.

2.2 Groups of Users

The categorization in section 2.1 leads also to a categorization of the users of statistical software. This is basically done upon their skills in computer science and statistics:

- The first group includes the developer of statistical software, which is used by other statisticians and domain experts. Obviously people in this group need

extended knowledge of computer science and latest technologies.

- Most statisticians fall into the category of users of statistical software. They use programmable systems as well as closed systems to solve their problems, and usually would not dare to write software from scratch, but rather extend existing systems.
- The last group of users of statistical software includes everybody who has almost no skills in programming and thus uses tailor made software to solve specific problems.

3 Two Threads of Interactive Computing

The terminology of "Interactive Computing" has two understandings according to the background it is used on. When statisticians moved to systems where they didn't have to write complete programs and submit a stack of punch-cards, but were able to interact with a piece of software via a teletype or ASCII-terminal, these environments were certainly worth to be called interactive.

Text Interface:

The user interacts with a statistical processor via a textual command language.

The system responds immediately to the user input.

Single commands are executed step by step, no program has to be written in advance.

Tasks can be gathered into functions.

Graphics Interface:

The user interacts with graphical representations of the data like plots, icons etc.

The system responds immediately to the user selections.

Single actions are performed step by step.

Usually there is no way of recording or abstracting actions.

Extendability

Figure 2: The two understandings of interactive statistical computing.

But after the broad introduction of graphical user interfaces in the mid 80s, a new understanding of interactive computing came up. Interactivity was now connected

Splius:

```
> fit <- glm(Number ~ Muser + Temperature + WaterSoftness + Preference,
             family=poisson)
> fit
...
Degrees of Freedom: 24 Total; 18 Residual
Residual Deviance: 42.92866
> add1(fit, ~ .^2)
Single term additions

Model:
Number ~ Muser + Temperature + WaterSoftness + Preference
              Df Sum of Sq      RSS      Cp
      <none>                43.87621 73.12701
Muser:Temperature  1  1.25286 42.62335 76.74928
Muser:WaterSoftness 2  1.07484 42.80136 81.80243
Muser:Preference  1 20.50510 23.37111 57.49704
Temperature:WaterSoftness 2  6.07932 37.79688 76.79795
Temperature:Preference  1  4.35622 39.51998 73.64592
WaterSoftness:Preference 2  0.39516 43.48104 82.48211

> glm(Number ~ Muser + Temperature + WaterSoftness + Preference +
       Muser:Preference, family = poisson)
...
Degrees of Freedom: 24 Total; 17 Residual
Residual Deviance: 22.34719
```

Graphical Interface:

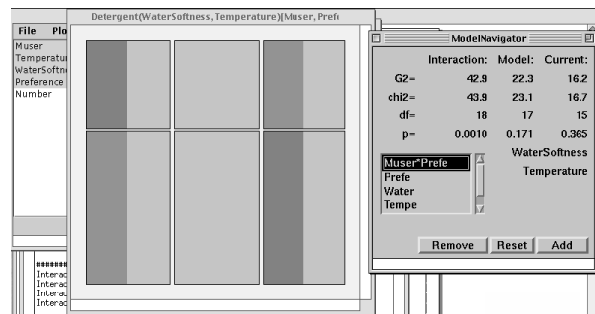


Figure 3: Stepwise interactive modeling of a log-linear model. On the left text based inside S-Plus. On the right graphics based inside Mondrian. The detergent dataset used here can be found in e.g. (Venables & Ripley, 1994)

with the direct manipulation of graphical representations of the data like plots and icons. E.g. parameters now could be represented by sliders and a change of a parameter usually would be propagated to all instances which are affected by this change. The basic difference between software packages following the one or the other paradigm is the fact that most graphical oriented packages are not extendible.

Figure 3 highlights the two threads using the example of setting up a log-linear model. The left part of Figure 3 shows the necessary steps in S-Plus. Most readers will be familiar with the modeling language of S-Plus and the corresponding output. On the right hand side an interactive graphical modeling is presented. The model is set up by choosing the different interactions found in the mosaic plot. Single keystrokes or mouse-clicks are used to add or delete interactions or to find out how the model would change by adding or deleting certain interactions.

This graphical stepwise modeling is implemented inside the Mondrian packages, which is entirely written in JAVA. For more information on the modeling technique and the Mondrian software see (Theus, 1998).

4 The S/S-Plus Success Story

The most often used statistical package in academia and research is doubtlessly S/S-Plus. There are several reasons why this is the case:

- Many basic linear algebra and statistics functions are accessible inside S/S-Plus. Thus new statistical methods can usually be programmed in S/S-Plus with little effort at a very high abstraction level.

- Unlike other systems the programming language in S/S-Plus is a well defined language based on a grammar.
- For more abstract tasks in data analysis (Chambers & Hastie, 1992) a even more abstract language enables a quick set-up of models and graphs (Trellis Display library).
- Since S/S-Plus keeps all its data on disk, the user never has to bother whether data is loaded or not. A more abstract way of handling data is to store datasets into Dataframes, which allows to access data of different modes.
- Although PostScript graphics is nothing special any longer, it was 12 years ago. But even today the fact that S/S-Plus graphics are extendible to suit the users needs is a big plus. However, the basic drawing routines in S celebrate their 30th anniversary this year, which does not leave much hope for any high interaction graphics!
- The most wide spread platform amongst universities and research organizations are UNIX boxes. Since S/S-Plus was developed under UNIX it was easy to port to a variety of different machines and thus reached almost all research communities.
- Given the widely accepted statistical computing environment of S/S-Plus, many researchers contributed new statistical methods to the build in functions of S/S-Plus. The probably largest collection of S/S-Plus function is StatLib at <http://lib.stat.cmu.edu/>. E.g. Wavelets and Neural Networks, which became popular amongst

statisticians in the last years, both could be obtained as free implementations for S/S-Plus.

5 The Desktop Revolution

The first interactive graphical systems came up in the early 70s. Unlike today these systems were of almost unaffordable costs. Thus the number of users was in many cases restricted to the research group, who implemented the software and the institution, which owned the hardware.

In the mid 80s desktop systems like the Macintosh came up, and X became the standard GUI for UNIX workstations. Some years later PCs followed with Microsoft Windows. The costs in hardware and software development dropped dramatically, and new interactive systems were implemented. The most successful of them are:

- DataDesk
(<http://www.lightlink.com/datadesk/>)
- JMP
(<http://www.jmpdiscovery.com/>)
- SAS-Insight
(<http://www.sas.com/rnd/app/stat/insight/insight.html>)

But as often the case with revolutions, the intended impact remains small, unless the underlying circumstances really change. Interactive graphical techniques did not become part of the curricula of statistics departments and statistical computing was not much influenced by new ideas. Reviewing Cleveland & McGill (1988) shows a clear stagnation in the field of interactive statistical graphics — although a few exceptions carry on this powerful paradigm.

Many statistical packages used GUIs only to provide multiple windows, and a menu driven invocation of commands. Thus these packages made use of the new GUI technology only in a superficial way, and did not include new ideas of statistical data analysis which were based on this technology.

6 The JAVA Hype

Surveying the biggest online bookstore we get (5/26/98) the following number of books on the programming languages FORTRAN, COBOL, PASCAL and JAVA¹ shown in figure 4. Since so much has been written on

¹The used search-engine failed to give reliable numbers for C/C++

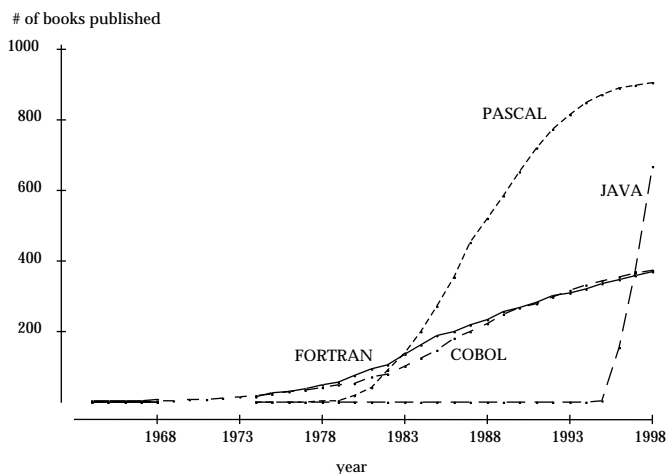


Figure 4: The number of books published on certain programming languages in the last 30 years.

JAVA already, many attributes are connected with the language, which might not characterize it very well.

JAVA is not a particular easy-to-learn programming language. However, it is a well defined object oriented language, which had an ISO standard before it was widely in use — in contrast to C++, which got its ISO standard in late 1997. JAVA is not a language especially designed for network tasks, but a general purpose language well aware of problems in network programming.

A primary concern about JAVA shared by many programmers is the matter of speed. Obviously JAVA virtual machines (VM) can not reach the speed of native C/C++ compiled code. However, the speed of VMs is steadily improving due to intensive research and more optimized implementations. Most of the tasks statisticians implement can be processed fast enough with today's VMs.

The main advantage of JAVA over other programming environments is its portability. JAVA classfiles can be run on any implementation of the JAVA VM given the necessary classfiles of the JAVA-packages used. Implementations of the JAVA Development Kit (JDK) can be found on any major system. The latest implementation of the JDK can be found for SUN's Solaris operating system and for Windows PCs. Other platforms usually suffer a delay of several weeks or months in releasing the latest JDK.

Incompatibilities between platforms can usually be explained by bugs in the particular implementation.

Java is a simple language with only 59 reserved words,

from which 11 are not even used in JDK 1.1. Thus the many features JAVA offers are all part of the JAVA API. The JAVA API of the JDK 1.1 consist of 23 packages. Although many people might associate JAVA only with Applets, only one out of these packages deals specifically with Applets.

The most important packages for statistical computing tasks are:

- AWT and Swing (part of JDK 1.2) for a unified GUI
- JDBC, the JAVA database interface, which allows direct data access to databases (remote and local).
- 2D, (part of JDK 1.2) 3D which offers almost Display PostScript capabilities and tools for scientific data visualization.

7 Where we stand right now

7.1 Applets

Most incarnations of statistical software in JAVA today are Applets. The two best known shall be investigated more closely here.

The first is WebStat at <http://www.stat.sc.edu/~west/webstat/> by Webster West, which provides various graphs and some basic statistical tests. Figure 5 gives an example of a 2x2 layout of the graphics window, showing four different graphs of the Iris dataset. WebStat's graphics output is customizable to a certain degree. It is a free package.

Statlets is a commercial package, and a demo version (limited to 100 observations) can be obtained at <http://www.statlets.com/>. Although Statlets implements more functionality than WebStat, it is still far from being an industrial strength data analysis package. Figure 6 shows a Statlets window containing a stacked barchart for the first 100 observations from the Cars dataset taken from the DataDesk example datasets. Besides pure JAVA applets several commercial packages implement JAVA frontends to their packages. The most advanced example today is the XploRe (Härdle et al., 1995) package. These frontends allow a distributed usage of packages, where different users may use arbitrary platforms. They connect to the same server running only once in the intranet on a specific machine via a web-browser.

Searching the web we find various tiny Applets which demonstrate properties of statistical methods. A nice example is the Applet at <http://www.stat.sc.edu/>

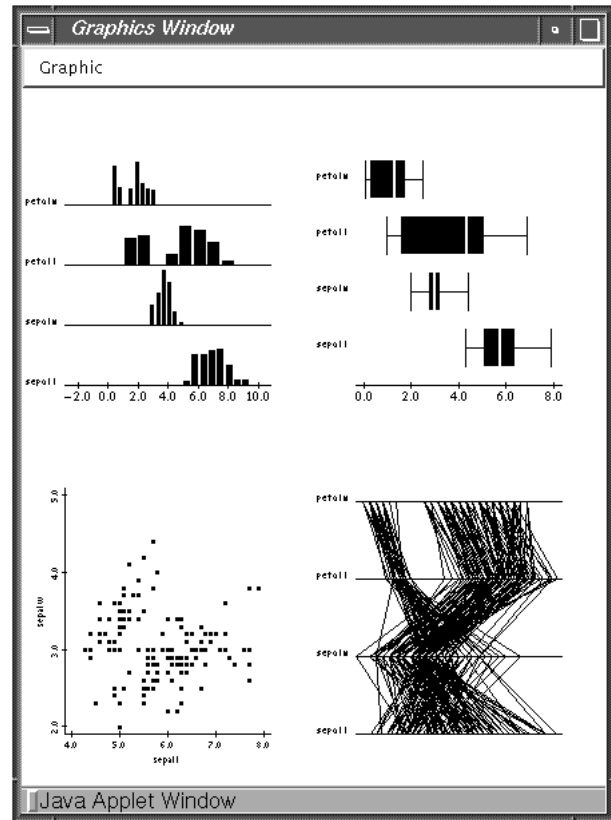


Figure 5: An example of a WebStat graphics window.

[~west/javahtml/Regression.html](http://www.stat.sc.edu/~west/javahtml/Regression.html) showing the leverage of a single point in a simple regression setting. Nonetheless products like Velleman's ActiveStats (<http://www.awl-ile.com/stats/activstats/>) seem to offer a more solid basis for an interactive support in classroom teaching.

Although JAVA Applets and applications differ only by a couple of lines in their source code, JAVA Applets suffer some major weaknesses:

- Applets usually are much slower than their equivalents running as application. This is mainly due to limited bandwidth on the network and security issues on the network.
- The data input and output is very restricted for security reasons. Although these limitations perfectly make sense for most Applets to protect the user from unauthorized access by an Applet, they force any data access of a statistical Applet to go through the HTTP-daemon of the host where the dataset is stored.
- The different progress of the different implementations of the JDK already seems to be a restriction

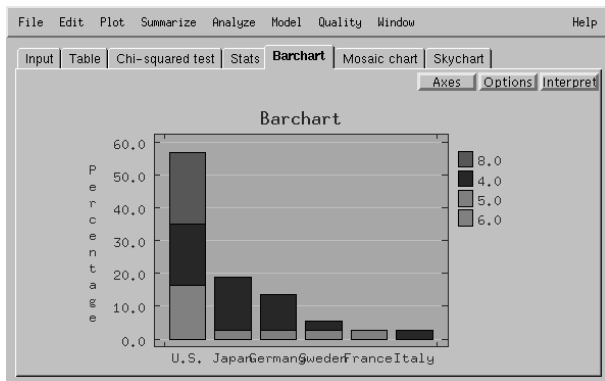


Figure 6: The Statlets window showing a stacked bar chart.

to really universal code, the differences of the implementations inside various browsers add one more dimension of possible problems.

Since Statlets can be run as an application as well, the reader may experience the performance difference between Applets and applications himself/herself.

7.2 Applications

Since JAVA can be used to build stand-alone applications just like any other programming language people will start to build programs in JAVA as they did in FORTRAN, C or C++.

Mondrian (<http://www.research.att.com/~theus/Mondrian/Mondrian.html>) is a data-visualization system written as a JAVA-application. The main emphasis of Mondrian is on visualization techniques for categorical data and geographical data. All plots in Mondrian are fully linked, and offer various interrogations. Any observation selected in any plot in Mondrian is highlighted in all other plots. Mondrian is research software, and was initially started to exploit the possibilities of implementing high interaction graphics in JAVA. Currently implemented plots comprise mosaic plots, maps, barcharts, parallel coordinate plot, boxplots and scatterplots. Mosaic plots in Mondrian are fully interactive. This includes not only linking, highlighting and interrogations, but also an interactive graphical modeling technique for log-linear models already showed in figure 3. Mondrian works with data in ASCII files, or connects directly to databases, given a JDBC-driver.

7.3 Packages

As already mentioned in section 6 the core building blocks of JAVA are packages. Packages gather methods

which can be applied to certain objects.

JNL is a numerical library for JAVA by Visual Numerics (<http://www.vni.com/>). JNL is a set of classes for the most important numerical functions missing in JAVA. The library contains a numerical type class: **Complex**, and three categories of numerical functions classes: the special functions class, the linear algebra classes, and the statistics class. All classes use double precision floating point as the underlying float type. More information can be found at <http://www.vni.com/products/wpd/jnl/JNL/docs/overview.html>. The following list shows the methods of the statistics class inside the JNL:

- average(double [])**
Returns the average (mean).
- FCdf(double, double, double)**
Returns the value of the cummulative F distribution.
- inverseFCdf(double, double, double)**
Returns the inverse of the cummulative F distrib.
- inverseNormalCdf(double)**
Returns the inverse of the cummulative normal dist.
- inverseTCdf(double, double)**
Returns the inverse of the cummulative t distrib.
- kurtosis(double [])**
Returns the kurtosis.
- linearFit(double [], double [])**
Returns the the least squares fit of a line to the data.
- maximum(double [])**
Returns the maximum value.
- median(double [])**
Returns the median value.
- minimum(double [])**
Returns the minimum value.
- normalCdf(double)**
Returns the value of the cummulative normal distrib.
- range(double [])**
Returns the range (xmax-xmin).
- skew(double [])**
Returns the skew.
- slope(double [], double [])**
Returns the the slope of the best least squares fit of a line passing through the origin to the data.
- standardDeviation(double [])**
Returns the sample standard deviation.
- tCdf(double, double)**
Returns the value of the cummulative t distrib.
- variance(double [])**
Returns the sample variance.

The linear algebra class provides various manipulations on double precision or complex matrices including Cholesky, LU, QR and SVD decompositions. The class **Sfun** complements JAVA's **Math** class.

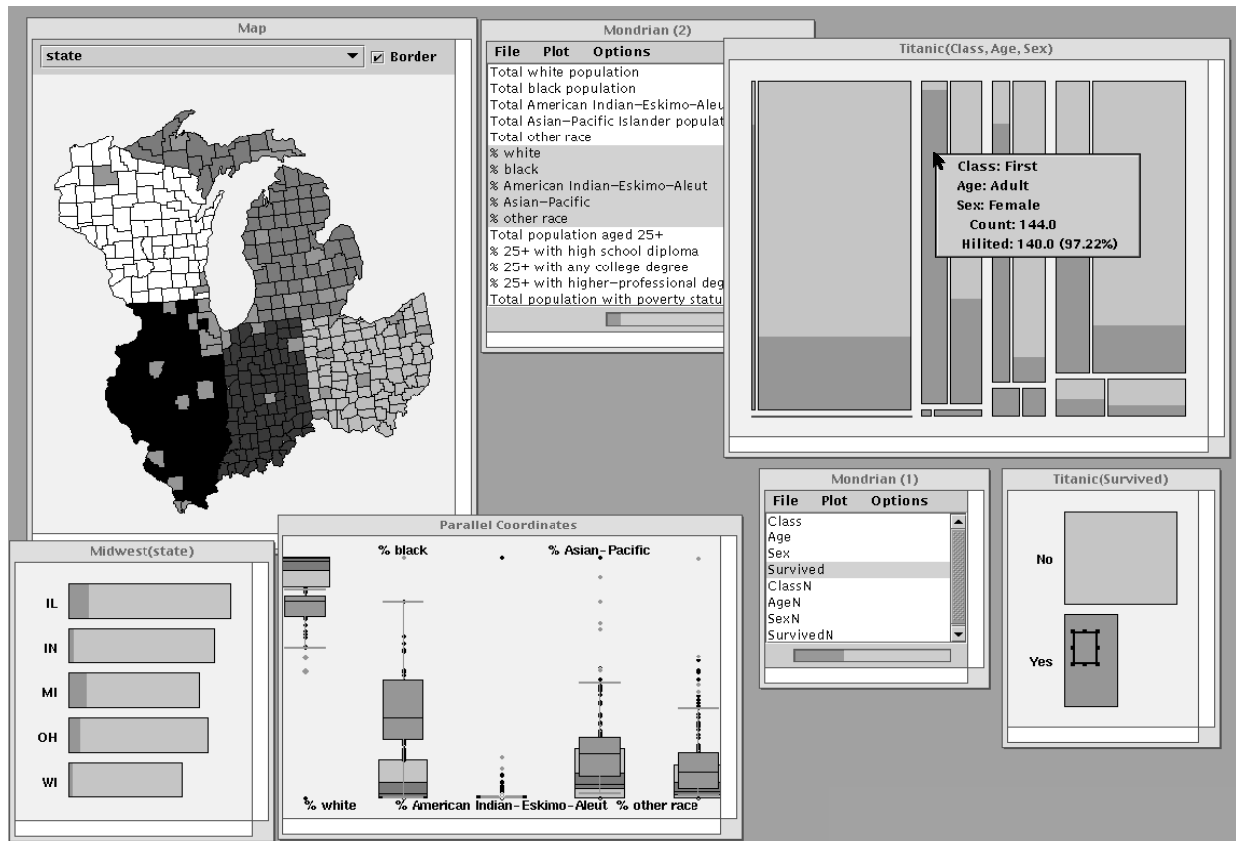


Figure 7: A sample screen of a Mondrian session. The plots on the left shows a dataset on five US mid-west states. The Titanic dataset is displayed in the mosaic plot and the barchart on the right.

Another promising package is the port of the LAPACK and the BLAS library to JAVA. A beta version can be downloaded at <http://www.cs.utk.edu/f2j/download.html>. The final version will be distributed at Netlib <http://www.netlib.org/>. JAVA BLAS is translated from the BLAS Fortran reference source archived at the Netlib numerical software repository. BLAS is composed of three levels of linear algebra operations:

- Level 1 BLAS consists of vector-vector operations such as dot product, etc.
- Level 2 BLAS consists of vector-matrix operations.
- Level 3 BLAS consists of matrix-matrix operations.

JAVA BLAS levels 1, 2, and 3 have passed the Netlib test suite. JAVA LAPACK has completed the testing stage. The javadoc reference for BLAS and LAPACK can be found at <http://www.cs.utk.edu/f2j/docs/html/packages.html>.

The JAVA Generic Library (JGL) contains various al-

gorithms and datastructures, which are not part of the JAVA util package. The JGL library can be found at <http://www.objectspace.com/jgl/>.

Whereas the accuracy of JAVA BLAS and LAPACK is guaranteed it has to be proven for the other packages. The efficiency of the packages might vary, too.

8 Where to go?

Using the categorization of section 7 I like to highlight where JAVA development will move to in the near future:

- **Applets** will increasingly be used inside intranets to provide easy access to statistical monitoring processes. These tasks, formerly covered by little helper applications designed to run on certain computers, are then accessed through browsers, running on various platforms.
- In the near future **Applications** will be written entirely, i.e. 100% in JAVA. These applications will

then be totally independent from any platform. There is no reason why JAVA applications should be less stable or have less performance than applications we use today, written in C and C++. JAVA applications open the door to a platform independent development of highly graphical oriented software.

- As we saw in section 7.3 **JAVA packages** are the building blocks for using JAVA in a specific domain. The various packages listed in section 7.3 already provide a good basis to build software on, without the need of reimplementing common methods. But given JAVA packages/libraries are no longer restricted to pure numerical tasks. The unified graphics system allows to build graphical libraries as well. Providing essential visualization tools in a unified package would allow users to easily add various plots to their statistical analyses written in JAVA. Section 5 talked about the big effort of developing interactive statistical graphics software. JAVA-packages can already include the core functionality like linking and highlighting, which would allow to set up linked plots with very little programming effort, compared to most other GUIs.

9 The next Generation?

I do not dare to make a forecast of how statistical computing environments will look like in the future. But what I can do is to vision what statistical computing can look like in the future if we make use of JAVA.

In a first step it might be worth to think about the future of statistical computing in the light of teaching. It is essential to get students and young researchers interested and involved into statistical computing issues. Looking at the field of data mining, engineers and computer scientists start to take over what used to be a classical area for statisticians. Statisticians more and more fail to keep up with this new development, either because of too little experiences with real data problems, or their skills in computer science are too small.

Many computer science departments start to teach programming using JAVA as the underlying language. Thus students of statistics and mathematics will be used to JAVA and we should not discourage them by sticking to older environments we might be more used to.

Let us review the previous sections of this paper in order to see what the impact of JAVA can be:

- The particular strength of JAVA is certainly the possibility to write libraries/packages, which are no longer bound to a certain platform and can now include high interaction interfaces.
- Given these packages, JAVA can be used as an universal tool to develop and exchange statistical methods.
- With JAVA graphical interfaces are much easier to implement and port. This should encourage statisticians to revive the impetus found in the papers collected in (Cleveland & McGill, 1988).

To support this development we need well defined package APIs, which are specially designed for statistical computing tasks. Based on this design — which probably should be done at a central place — these packages can then be implemented by various developers and users.

Acknowledgement

I like to thank all my colleagues at the statistics group at AT&T Labs Research for their contributions and discussions about their experiences with JAVA. Special thanks goes to Richard Becker for a review of the statistical computing environments used in times when I still was in kindergarten.

References

- Becker, R. A. (1989), "Statistical Computing Environments: Past, Present and Future" *150 Years ASA — Sesquicentennial Invited Paper Sessions*.
- Becker, R. A. (1994), "A Brief History of S," *Computational Statistics*, Eds. Dirschedl & Ostermann, Physica, Heidelberg.
- Chambers, J. M. & Hastie T. J. (Eds) (1992) *Statistical Models in S* Wadsworth&Brooks, Pacific Grove, CA.
- Cleveland, W. S. & McGill M. E. (Eds) (1988) *Dynamic Graphics for Statistics* Wadsworth&Brooks, Pacific Grove, CA.
- Flanagan, D. (1997), *JAVA in a Nutshell, 2nd Edition* O'Reilly, Cambridge.
- Härdle, W., Klink, S., Turlach, B.A. (1995), *XploRe: An Interactive Statistical Computing Environment* Springer, New York.
- Theus, M. (1998), "Visualizing Loglinear Models" to appear in *Journal of Computational & Graphical Statistics*.
- Venables, W. N. , Ripley, B. D. (1994), *Modern applied statistics in S-Plus* Springer, New York.