The R and Omegahat Projects in Statistical Computing

Brian D. Ripley

RSS Statistical Computing Section, 21 November 2001

ripley@stats.ox.ac.uk
http://www.stats.ox.ac.uk/∼ripley

Outline

- Statistical Computing
  - History
  - S & R
  - The Omegahat Project
- Applications and Interfaces
  - Web Servers
  - Embedding — Medieval Chant, Dynamic Graphics
  - Databases
- The Future?

A useful reference is the DSC-2001 proceedings at
http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/

Statistical Computing and S

Scene-setting: Statistical Computing

1980

Mainly Fortran programming, or PL/I (SAS).
Batch computing (SAS, BMDP, SPSS, Genstat) with restricted range of platforms.

Some small interactive systems (GLIM 3.77, Minitab).

Very poor interactive graphics (2400 baud to a Tektronix storage tube if you were lucky). Flatbed and drum plotters, microfilm for publication-quality output off-line.

Mainly home-brew solutions in research. (GLIM macros?)

1990

PCs become widespread, but FPUs still uncommon. Sun etc workstations available for researchers, and for teaching in a few places.

Graphics could be pretty good (postscript printers, ca 1000×1000 pixel screens), but often was not, and mono text terminals were still widespread.

C was beginning to be used, as more portable than Fortran. (Few PC Fortran compilers then and now.)

Still SAS, SPSS etc as batch programs.

S beginning to be make an impact on research and teaching.

2001

Little spread in machine speed (min 800MHz, max 2GHz), fast FPUs are universal.

Colour everywhere, usually 24-bit colour.

The video-games generation is now at university.

Few people would dream of writing a complete program for a research idea: prototype and distribute in a higher-level language such as S or Matlab or Gauss or Ox or ....

Fortran is still used in scientific computing, but C or C++ is preferred, and Java has its advocates. SAS lives on as pseudo-batch program.

Lots of specialized tools are widespread, such as Perl, Python, Web browsers.

XML (eXtensible Markup Language) is the flavour of the year.

Scene-setting: The ‘S’ Language

Largely the work of one person, Dr John M. Chambers of Bell Laboratories (formerly AT&T, now Lucent Technologies).

Awarded the prestigious 1998 *Association for Computing Machinery* Award for Software Systems for, in the words of the citation,

> *the S system, which has forever altered how people analyze, visualize, and manipulate data.*

For the last decade it has been the major vehicle for the delivery of new statistical methodology to end users.

S has a long history: the GR-Z graphics system goes back to 1976. JMC is now a Bell Labs Fellow, and is working on *Omegahat*, so that can be considered the successor to S.

S History

The names have changed ('New S' and 'QPE' came and went) but the flavours of S are now known mainly by the colours of the covers of the books co-authored by Chambers.

S1	1984	*brown*	macro-based extension language
S2	1988	*blue*	user-written extensions as first-class objects
S3	1991	*white*	classes, some statistical functionality
S4	1998	*green*	more rigorous class system

All were Unix programs written in C and Fortran.

S-PLUS® was first produced in 1988 by a start-up in Seattle called *Statistical Sciences* which in 1993 acquired exclusive marketing rights to S and merged with *MathSoft*. In 2001 they demerged and became *Insightful*.

S is not thought of by its developers as a statistical system, rather as *an interactive environment for data analysis and graphics*, a system within which to do statistics.

S-PLUS has been available for a limited range of Unix platforms and DOS and then Windows. It was not available for Linux until 1998, and never for Macintoshes. Current versions are based on S4.

S-PLUS is very widely used for teaching statistics at graduate level. Some of the early enthusiasts were earth scientists, and it has been used for service teaching. It has had less impact for mainstream undergraduate teaching, despite radical approaches like Nolan & Speed (2000) *Stat Labs: Mathematical Statistics through Applications*.

S-PLUS is now pretty successful in several commercial sectors (finance, pharmaceuticals, manufacturing). A recent push is to Web-based delivery (StatServer® and S-PLUS Analytic Server) with planned moves towards data mining.

What is R?

R History

R is a system originally written by Ross Ihaka and Robert Gentleman of the University of Auckland (so the naming is clear) in about 1994. To the user it looks like a dialect of the S language, but the internal implementation is based on ideas from Scheme (a member of the LISP family). It is 'not unlike' S3, and becoming more like S4.

Probably this started as a research project, but versions were used at Auckland for elementary classes, on Macintoshes with 2Mb of memory.

By 1997 other people had become involved, and a *core team* had been set up with write access to the source code. (No one kept records of who joined when.) There was a Windows version, and Linux users pushed development forward, there being no S-PLUS version available for Linux.

I became involved in 1998, and a member of the core team in Jan 1999.

The first non-beta version of R, 1.0.0, was released on 29 Feb 2000. There are several releases a year (probably six in 2001).

Where is R now?

It is a system available as source code (at `www.r-project.org`) that compiles on almost all current Unix and Linux systems, and has binary versions for the major Linux distributions (Red Hat, SuSE, Debian, Mandrake), FreeBSD, MacOS X and 32-bit Windows and classic Macintosh (which also runs on MacOS X).

It is distributed under GPL2 (the GNU Public Licence).

The core system is fairly small but can be extended by *packages*, 10 of which ship with R and about 150 are available (13 'recommended') from CRAN (`cran.r-project.org` and mirrors). Collectively these cover a wide range of statistical functionality, mainstream and oddball.

R *vs* S-PLUS

The two systems co-exist, uneasily at times.

- S-PLUS is commercial. R is freely distributable.
- R is much smaller and runs on less powerful machines. On Windows it fits on four floppies and (I'm told) runs on an 8Mb Windows 95 machine.
- S-PLUS is monolithic: R has a small core plus many extensions.
- S-PLUS on Windows has a 'menus and dialog boxes interface'. R does not, and although there are means to program one, in C, Tcl/Tk or Java, they are laborious. [Demos by PD]
- Their performance is about equal, but R is much more tolerant of badly-written code that can make S-PLUS crawl.
- There is not much to choose in quality these days. I suspect R has more bugs, but they will be fixed faster by subject-matter experts.

- Both have 2D graphics of very high quality.
- R is currently missing the rich facilities of S for multi-panel plots (Trellis graphics) but alpha versions are available.
- But both are poor on 3D graphics and dynamic and interactive graphics. S-PLUS on Windows is better than on Unix, and an add-on for R on Windows is under development. [`Rgl` by Duncan Murdoch: Demo later]

It seems clear that the research emphasis in statistical computing has shifted from S to R: John Chambers is now a member of the R core team. The future looks like collaboration rather than competition.

What is Omegahat?

The Omegahat Language

Instead of S version 5, John Chambers and his colleague Duncan Temple Lang in 1998 started another track, a language called Omega. This arose out of discussion with (some of) the R developers and Luke Tierney on web-based software and distributed computing.

The *hat* was a contribution of Bill Venables to give a statistical flavour.

The Omegahat language (`www.omegahat.org`) is an interactive environment, effectively an interactive front-end to Java. It seems to have a handful of users.

The Omegahat project has also a range of Java packages implementing methods of interest in statistical applications.

The Omegahat Project

The aims of the project are wider, to provide a vehicle for collaboration in statistical computing, particularly to provide re-usable components.

As such it includes the whole R core team, the S and XLISP-STAT developers and several other people with cognate interests.

There is also a range of inter-system interfaces, for example between R or S and Java (of course), Perl, Python, Xalan, Netscape, Gnumeric and PostgreSQL. Also, a CORBA interface for R and for S.

The project is also an umbrella, at present for one sub-project on interfaces to DBMSs.

Omegahat uses a new-BSD licence that is more permissive than GPL2.

Applications and Interfaces

What is R being used for?

With a freely-distributable product, it is hard to know! However, users tend to ask for help, and a few contribute.

One of my main motivations for being involved is a (perhaps *the*) major use, to provide a first-class statistical system to students and researchers in the third world.

There are now many examples of R being used for large-scale data analysis. It was used for election forecasting in Austria and in the UK (by David Firth). My group use it to analyse 100Mb brain images.

There are several applications in gene-expression arrays, at least two of which are commercial systems built on R, and one, `sma`, is available from CRAN.

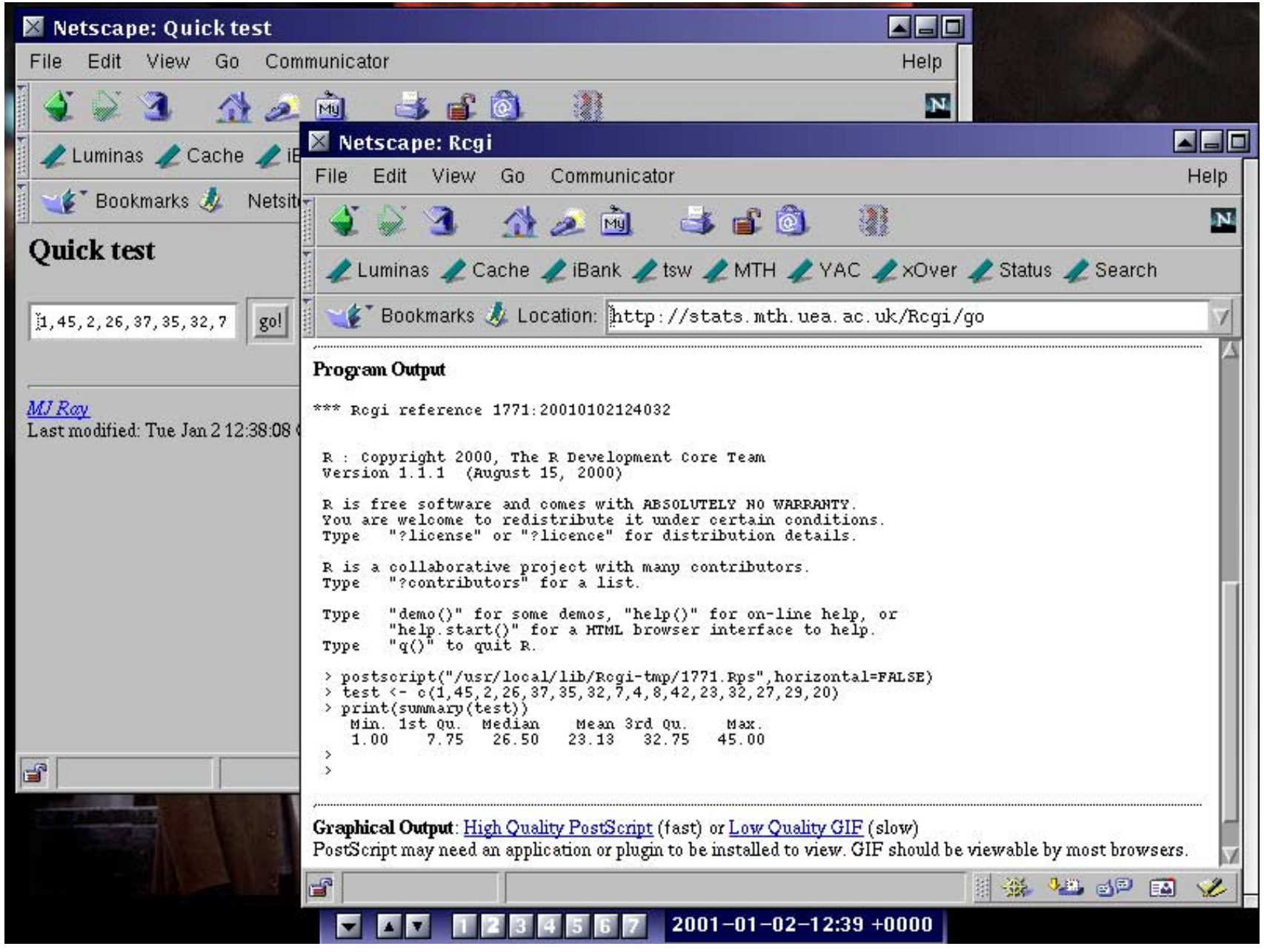
It is clear that researchers in many commercial companies are building systems around R.

Web-based Statistical Teaching

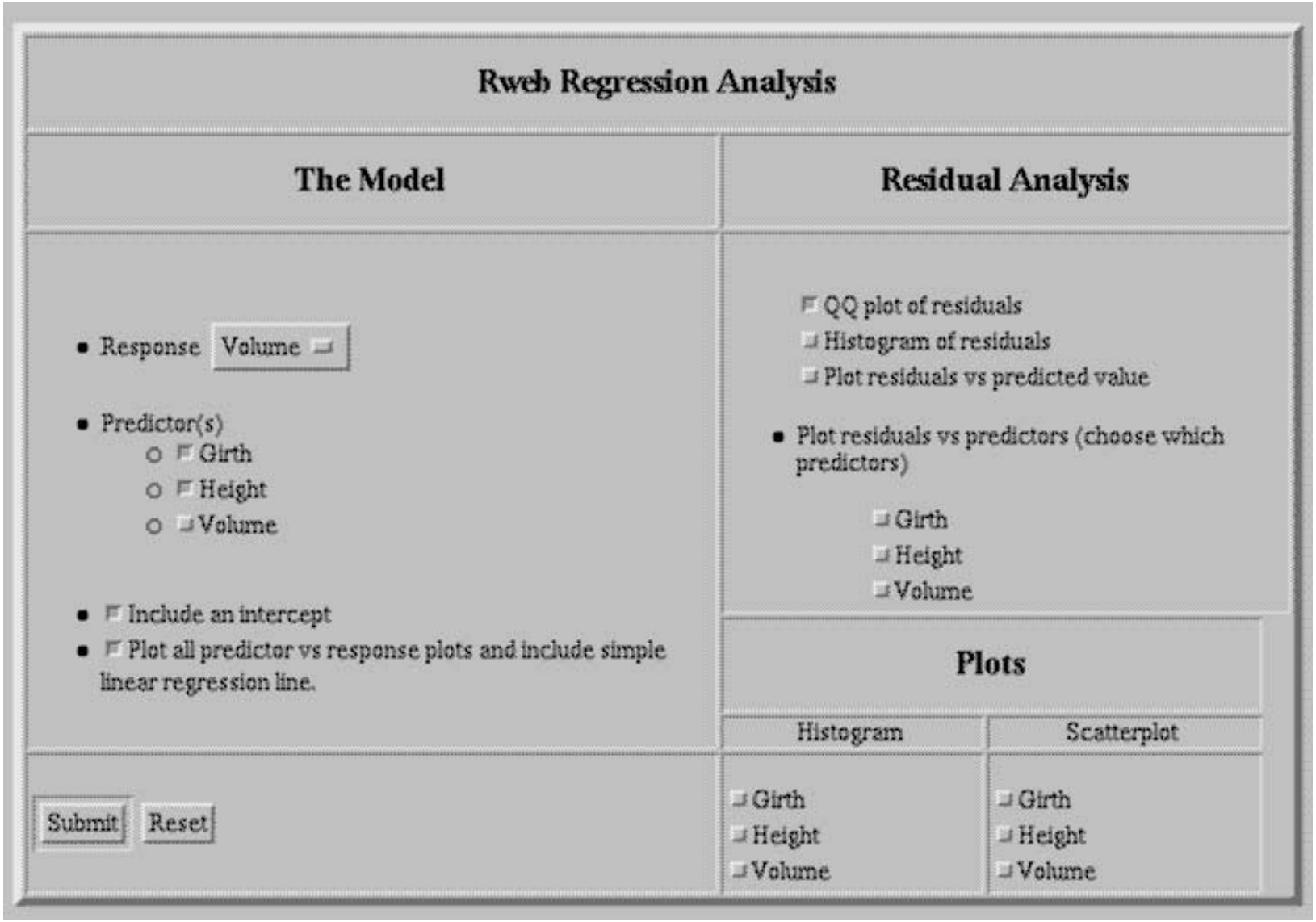
There are two harnesses, `Rcgi` (Mike Ray, U. East Anglia) and `Rweb` (Jeff Banfield, Montana State), to running R sessions from Web browsers. Both provide a simplified teaching interface.

`Rweb` provides 'a set of point and click modules that are useful for introductory statistics courses and require no knowledge of the R language'.

Rcgi Example



Rweb Example Module



Embedding

Embedding can be taken much, much, further.

It is most advanced on Windows, where Thomas Baier's DCOM interface allows R to be called from Excel, Visual Basic, ..., but there is also a Unix/Linux version of R as a shared library.

These enable R to do what it does best, statistical analysis and presentation graphics.

The Omegahat project has R embedded in Netscape, Gnumeric (a spreadsheet) and PostgreSQL (a DBMS) on Unix/Linux, although I have never seen a successful demonstration.

Analysing Chant —
An Example of R Embedding

Medieval Chant

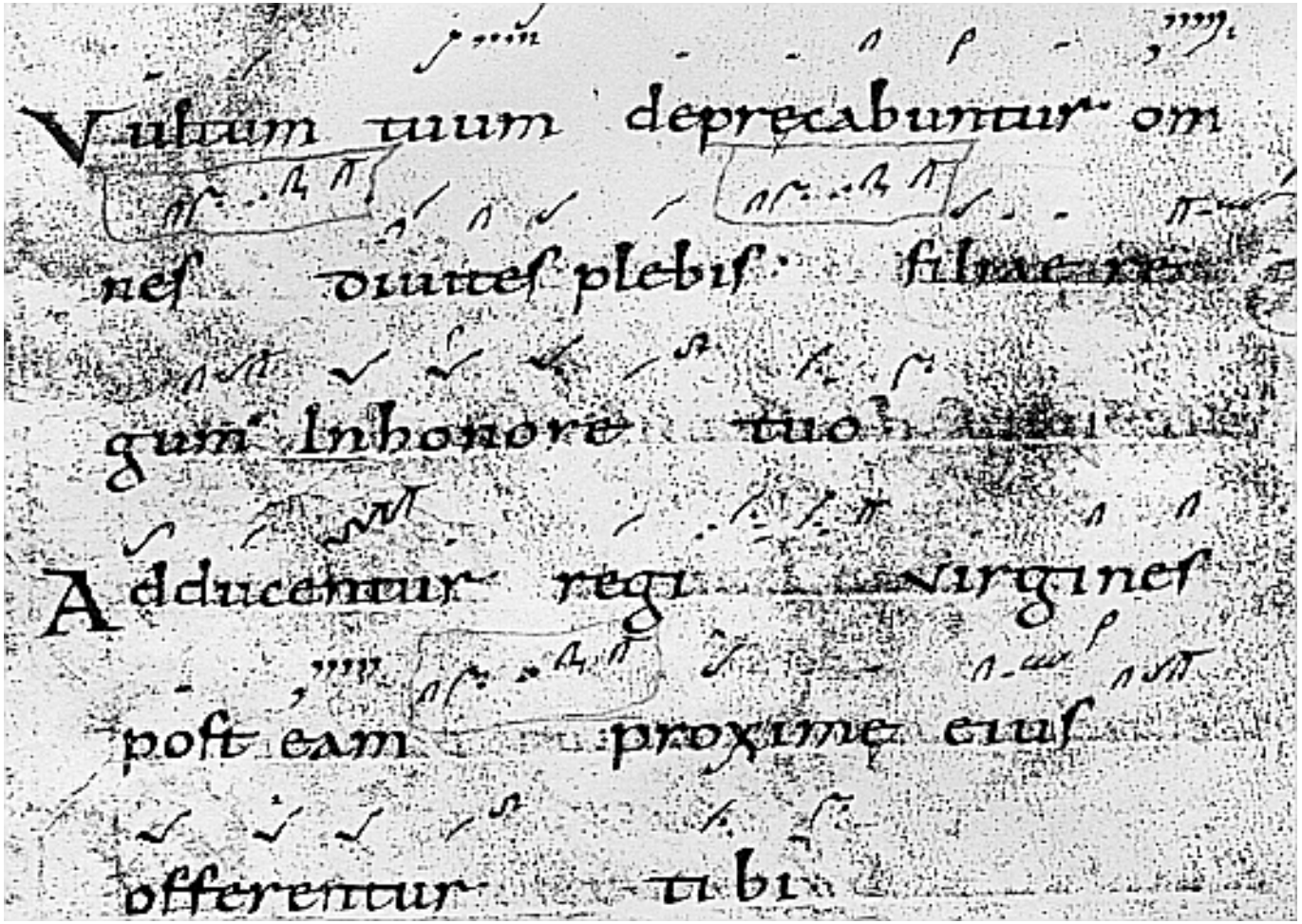
Musicologists undertaking detailed analyses of manuscripts of Western Christian liturgical chant dating back to the ninth century CE would welcome computer assistance. (Emma Hornby & John Caldwell, Faculty of Music in Oxford, statistics by Ruth Ripley.)

The early manuscripts employ several different notations, using *neumes* rather than notes. There are about twenty-five neumes, plus markings.

There a few thousand known chants with further variations between manuscripts. Ideally one would use optical character recognition to read them in: exploring the feasibility of that was a (successful) project for a Master's student this summer.

At present chants are entered by a point-and-click data entry system written in Visual Basic.

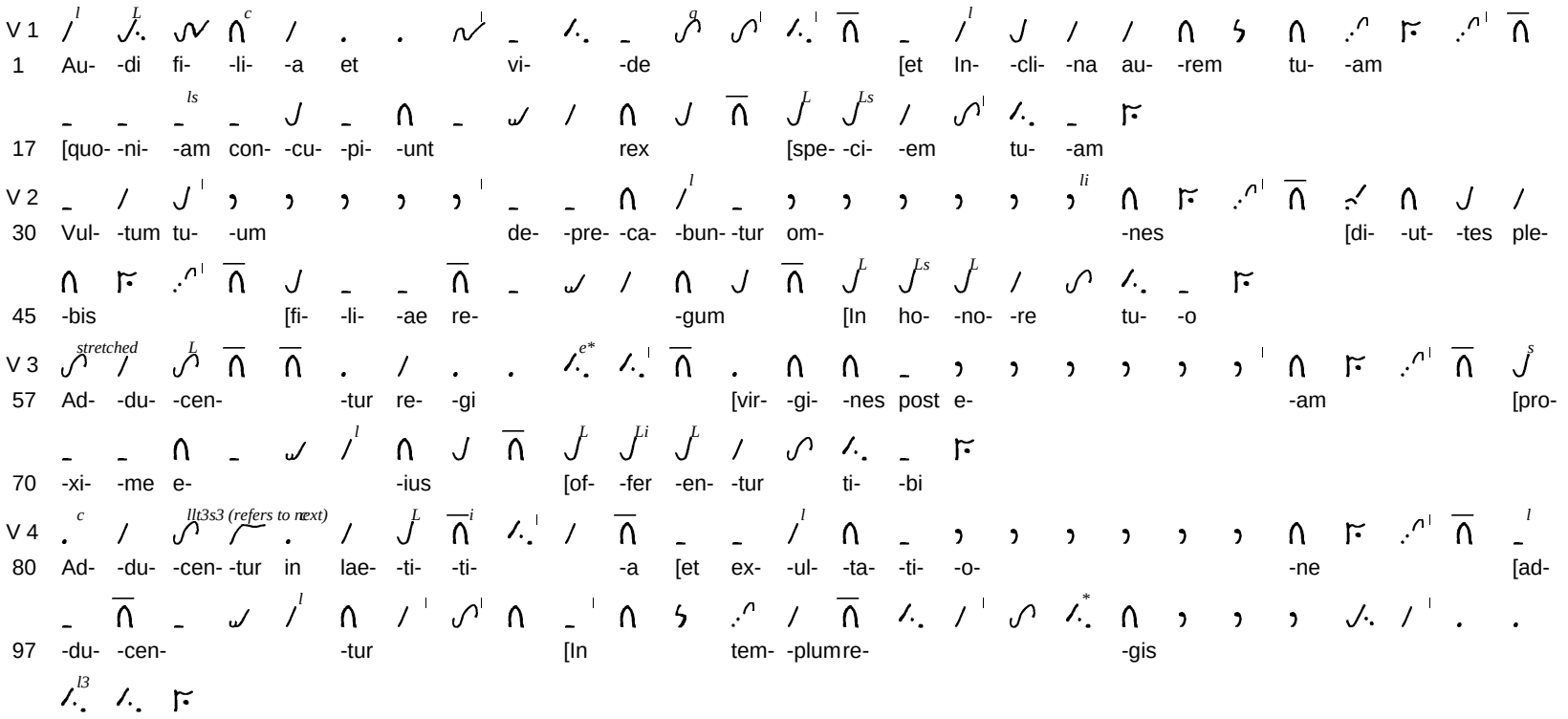
Part of a Chant — Original



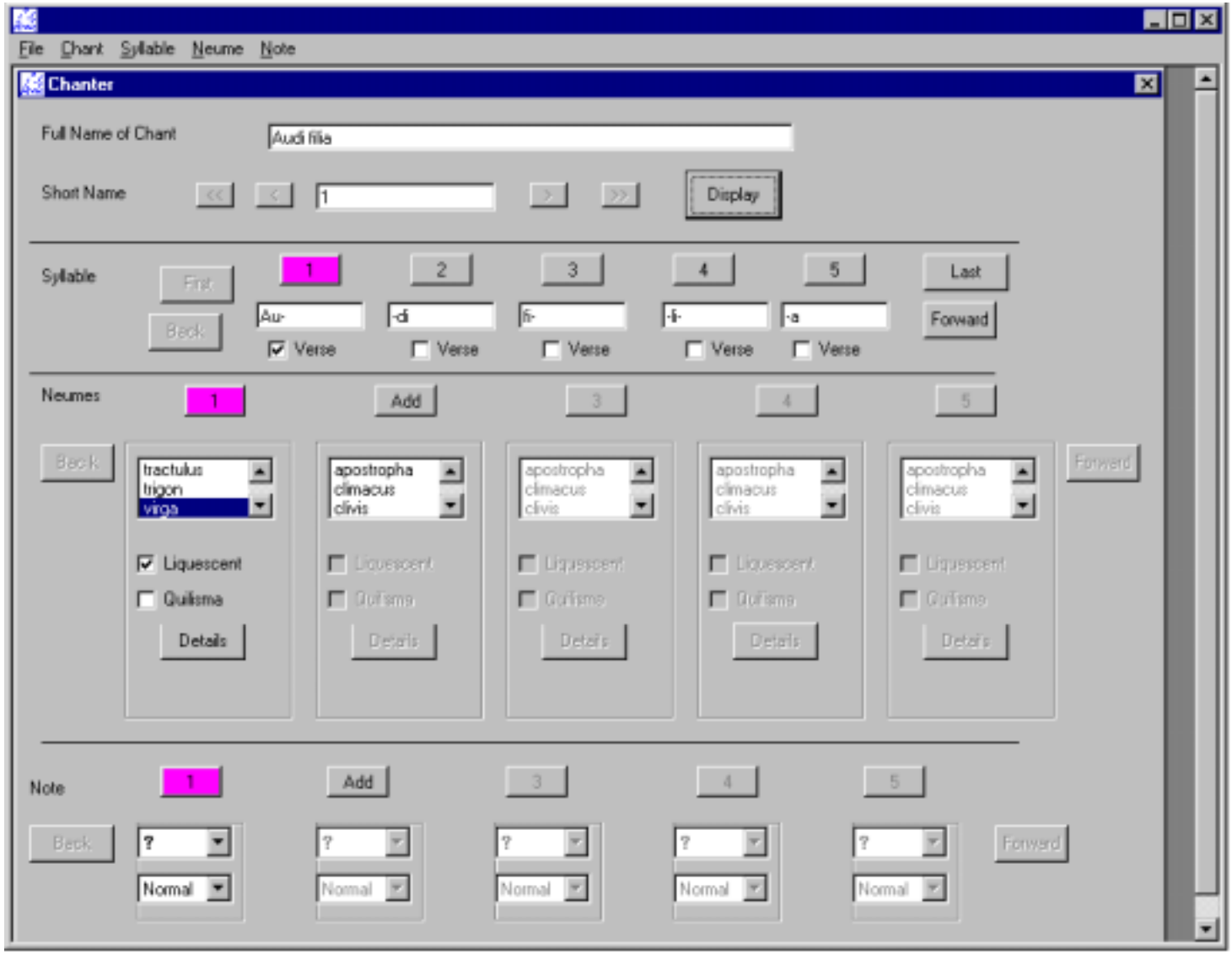
Two (annotated) verses of a chant, scanned and enhanced.

A Chant — as Entered

Chant fullname: Audi filia



Data Entry

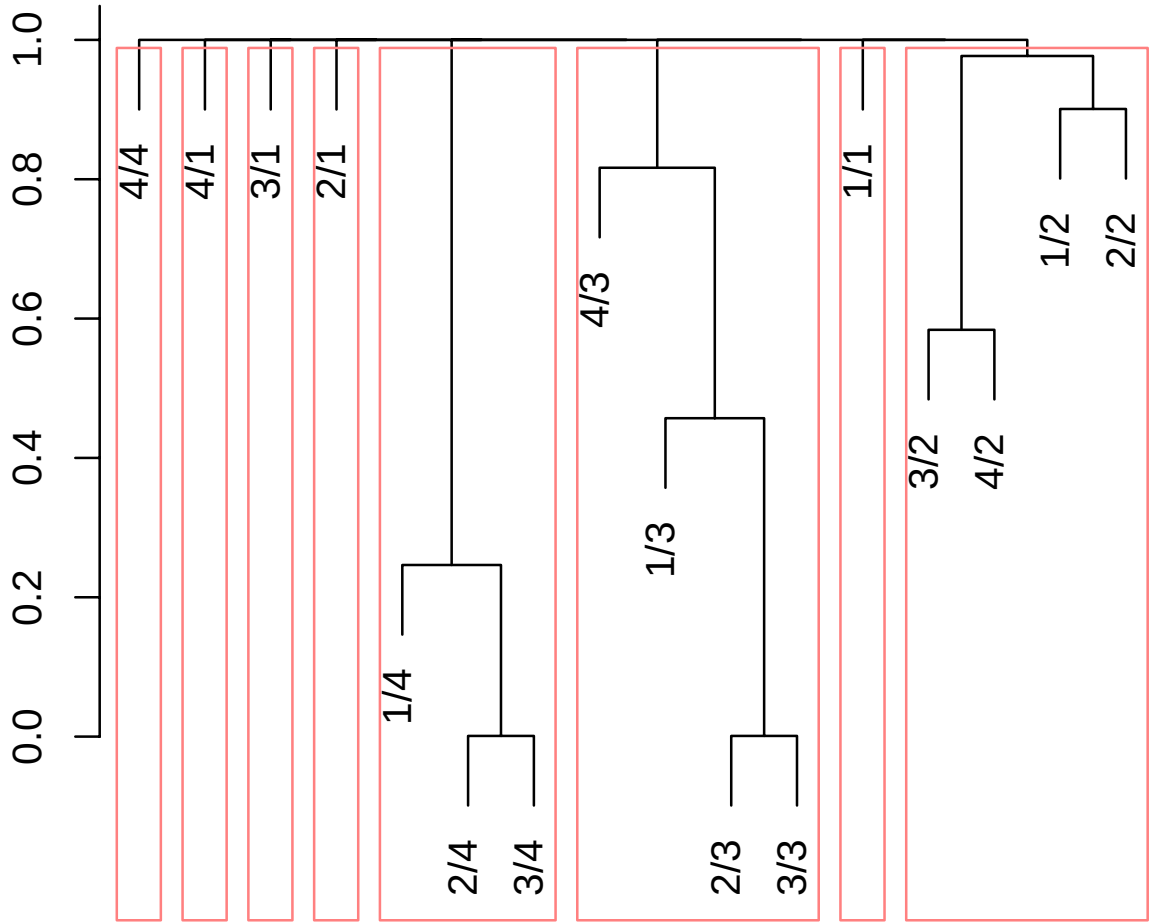

File Chant Syllable Neume Note
Chanter
Full Name of Chant
Audi filia
Short Name
<<
<
1
>
>>
Display
Syllable
First
Back
1
2
3
4
5
Last
Forward
Au-
-di
fi-
-li-
-a
Verse
Verse
Verse
Verse
Verse
Neumes
1
Add
3
4
5
Back
tractulus
trigon
virga
apostropha
climacus
clivis
Liquescent
Quilisma
Details
Forward
Note
1
Add
3
4
5
Back
?
Normal
Forward

Medieval Chant: Design Issues

- The system has to be usable on fairly minimal Windows PCs by users whose experience stretches to Word and Internet Explorer.
- Need to build a database of chants.
- Non-trivial display issues: involved designing a TrueType font.
- The matching algorithms to be used are fairly complicated and subject to tweaking, and will result in a similarity matrix S.
- Given S, use standard multivariate techniques to compare chants (or verses or phrases of chants).

Solution has been to use a Visual Basic front-end driving a database interface and also a connection to an R server via DCOM.

Sample Results: Analysis



Dendrogram of phrases within the four verses of a chant, with groups highlighted.

Embedding into R

Dynamic Visualization

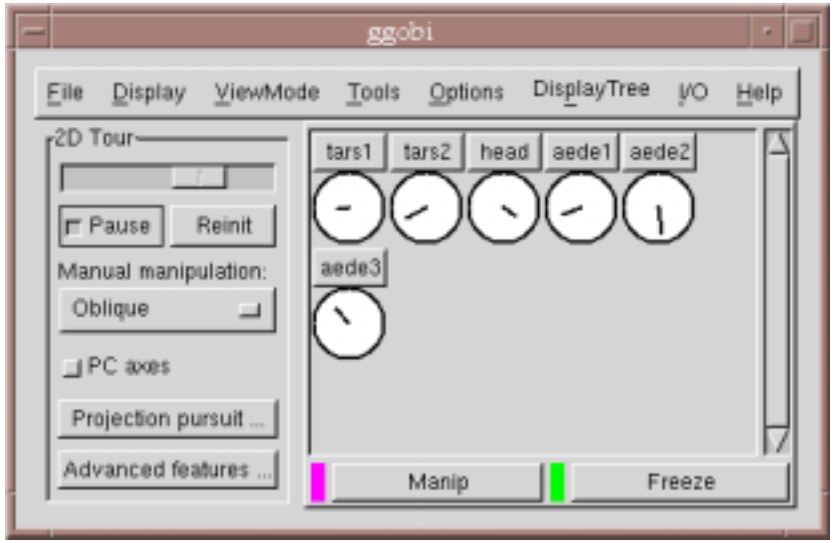
We have mentioned that R is lacking dynamic graphics. The obvious way to fill the gap is to borrow from another project, and Open Source licensing is intended to make such collaborations easy.

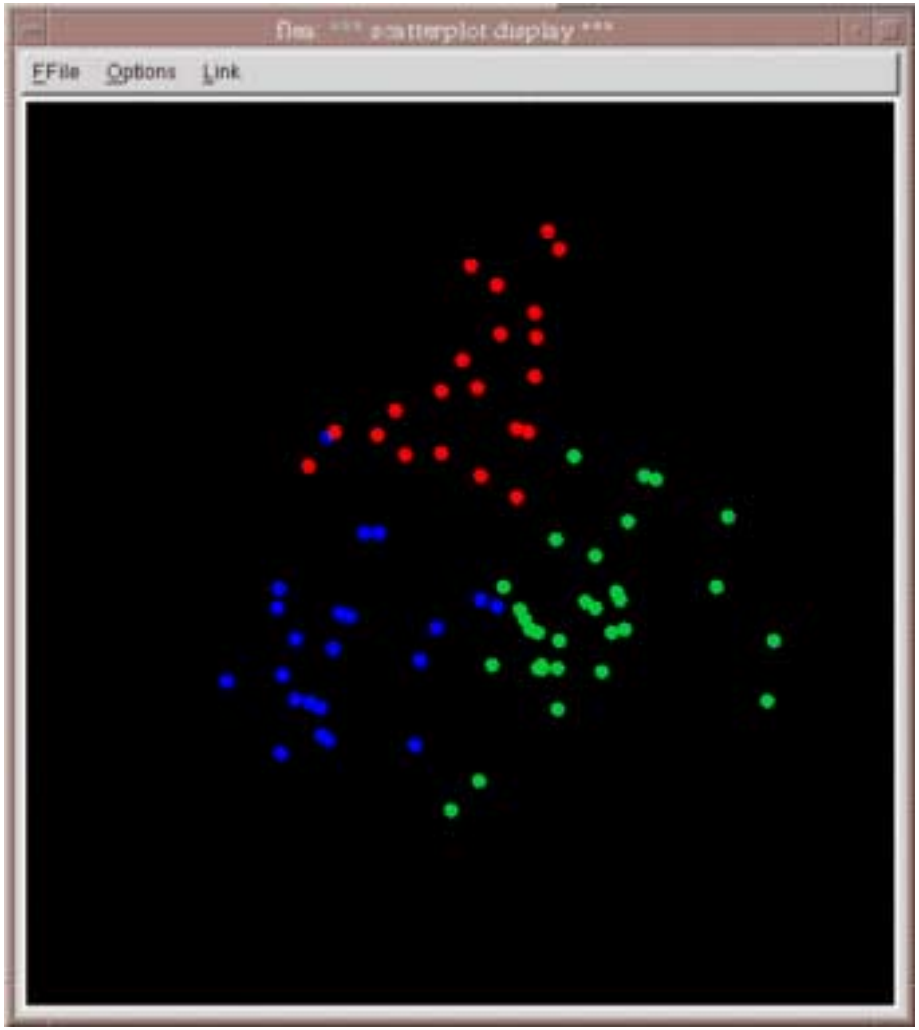
The XGobi system was developed at Bellcore/AT&T in the late 80's and early 1990's and provides an advanced dynamical graphics system. It runs under X Windows under Unix (and under Windows with an Xserver). Its developers were intending to move to the Gtk+ toolkit (developed for GIMP), and wanted a scripting language.

Out of this came GGogi (`www.ggobi.org`, still alpha/beta) which can run standalone (under Unix or Windows) or be embedded in R (or S-PLUS on Unix) which provides a scripting language.

This is currently incomplete, but one can currently select points and feed the results back to R, and eventually 'interesting' projections will be able to be passed back.

Ggobi ‘Grand Tour’ Example





Database Interfaces

Databases

or more properly, DBMSs (DataBase Management Systems). Several types

- Text and `csv` files.
- Spreadsheets (programs and formats), like *Excel*.
- Flat file databases like *DBase*.
- Hierarchical databases.
- 'Lean' relational databases like *Access*, *MySQL*, *MiniSQL* (aka *mSQL*) and *SQLite*.
- Heavyweight relational databases like *Oracle*, *DB/2*, *Informix*, *Sybase*, *SQL Server*, *PostgreSQL*.

Sometimes the user has a choice of database, but often not.

Standard DBMS Interfaces

Most DBMSs come with a *monitor*, a text-based client, and some have GUI clients (notably Access, which looks very like a set of spreadsheets).

Almost all have a C or C++ API. And they are almost all different. (I am told *Sybase* and *SQL Server* share a common interface (TDS).)

The actual commands are normally sent in a dialect of SQL (Structured Query Language). This has ISO standards, rarely complied with.

Having multiple standards with multiple levels encourages non-compliance.

Statistics and DBMSs

Why would you want to be involved with DBMSs?

- That's where the data are kept!
- DBMSs may be good ways to store analyses.
- Some DBMSs are designed for fast queries on large databases.

So there are two separate issues:

- Efficiently extracting data from a database (or storing results back in a database).
- Using the DBMS to do some of the calculations.

R / S-PLUS Interfaces

There are five DBMS-specific interface packages on CRAN,

`RPgSQL`	for *PostgreSQL*	by Tim Keitt
`RMySQL`	for *MySQL*	by David James & Saikat DebRoy
`RmSQL`	for *MiniSQL*	by Torsten Hothorn
`ROracle`	for *Oracle*	by David James
`RSQLite`	for *SQLite*	by David James

using the C interfaces. All run on Linux, and I have run `RMySQL` on Windows (with some difficulty). Those by David James have similar user interfaces.

S-PLUS 5.1/6.0 on Unix (and not Linux) have connections to import data from *Informix*, *Oracle* and *Sybase* databases.

`RMySQL` and `ROracle` have siblings for S-PLUS.

Cross-Database Solutions

ODBC (Open DataBase Connectivity) is an X/Open and ISO standard for a common interface to relational databases. It is common in the Windows world, and has been much enhanced by Microsoft and used as the basis of later developments (ADO, ...). On Windows, spreadsheets and even text files are covered. `unixODBC` has a text driver too.

ODBC drivers are available for all the common DBMSs (not all freely).

CRAN has a package `ROBDC` (by Michael Lapsley) which interfaces to ODBC driver managers on Windows and Linux/Unix.

Two other cross-database solutions worth noting are JDBC (Java ...) and Perl DBI. Either could be used via the Omegahat R/S interfaces to those languages.

RS-DBI and Rdbi

Both are common S-language interface with multiple (planned) backends interfacing to different DBMSs. Currently there is a backend to `Rdbi` for *PostgreSQL*, and to `RS-DBI` for *MySQL*, *Oracle* and *SQLite*.

More collaboration is needed!

Transparency

One idea is to be able to treat a database table as an R data frame, with indexing operations delegated to the DBMS. See the DSC-2001 proceedings for the current state and pros and cons.

The Future?

Where are R and Omegahat going?

This is hard to predict: volunteer projects go where the volunteers want to take them. But some predictions:

- The S language and (first) the R implementation will gain real object-oriented features.
- There will be a strong push towards establishing a synergy with DBMSs.
- There will be moves to ‘literate data analysis’ with XML as the glue.
- Threads, events, exceptions, . . . , will at last be tackled seriously.
- Cross-platform support will continue to be taken seriously.

Component-based Systems

- It is no longer credible to apply for large research grants to re-implement statistical methods.
- The commercial sector is finding it hard to cover the costs of re-implementing statistics (although people will pay for interfaces).
- Incremental development based on re-use and enhancement looks very promising.