DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria
http://www.ci.tuwien.ac.at/Conferences/DSC-2001
K. Hornik & F. Leisch (eds.) *ISSN 1609-395X*

Applications of R Clients and Servers

B. D. Ripley and R. M. Ripley

Abstract

Using R as a component in a distributed system is a fast way to obtain the best features of several tools. This is illustrated by

(a) using a Visual Basic client with R and database servers under Windows to enter and analyse medieval chant,

(b) using R as client to a DBMS holding large-scale insurance data, or embedding R within such a DBMS, and

(c) using R as client of an Oracle database holding gene expression data.

1 Medieval Chant

Musicologists undertaking detailed analyses of manuscripts of Western Christian liturgical chant dating back to the ninth century CE would welcome computer assistance.

The early manuscripts employ several different notations, using *neumes* rather than notes. There are about twenty-five neumes, plus markings. Figure 1 shows two verses of a chant, and something of the typical manuscript quality.

There a few thousand known chants with further variations between manuscripts. Ideally one would use optical character recognition to read them in, but exploring the feasibility of that is a 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.

There were some critical design issues:

- The system has to be usable on fairly minimal Windows PCs by users whose experience stretches to Word and Internet Explorer.
- There is a need to build a database of chants, and eventually to store information on the chants with them.

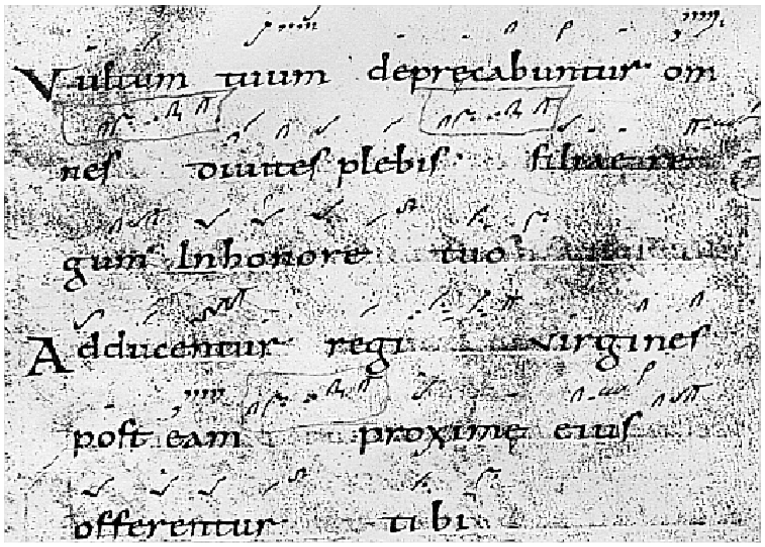


Figure 1: Two verses of a chant, after high-resolution scanning and some enhancement by image processing. The original had been annotated.

- The chants need to be displayed in a notation that is familiar to the musicologists. Figure 2 shows the current solution, which involved designing a TrueType font. This implies that neumes can be referenced by a font encoding as 7-bit characters.
- The matching algorithms to be used are fairly complicated and subject to tweaking, and will result in a similarity matrix S. Often the parameters need to be adjusted to give similarities that correlate with interesting features defined by the musicologists.
- Given S, we wanted to use fairly standard multivariate techniques to compare chants (or verses or phrases of chants).

Our solution has been to use a Visual Basic front-end driving a database interface and also a connection to an R server via DCOM.[1]

1.1 Sequence Matching

The techniques used are a modification of those from computational biology [4]. We use a dynamic programming algorithm that is essentially recursive. The textbook implementation is not well matched to the S language, shown here for an earlier version of the algorithm:

```
local <- function(seq1, seq2, sx=1, ss)
{
```

[1]Distributed Component Object Model

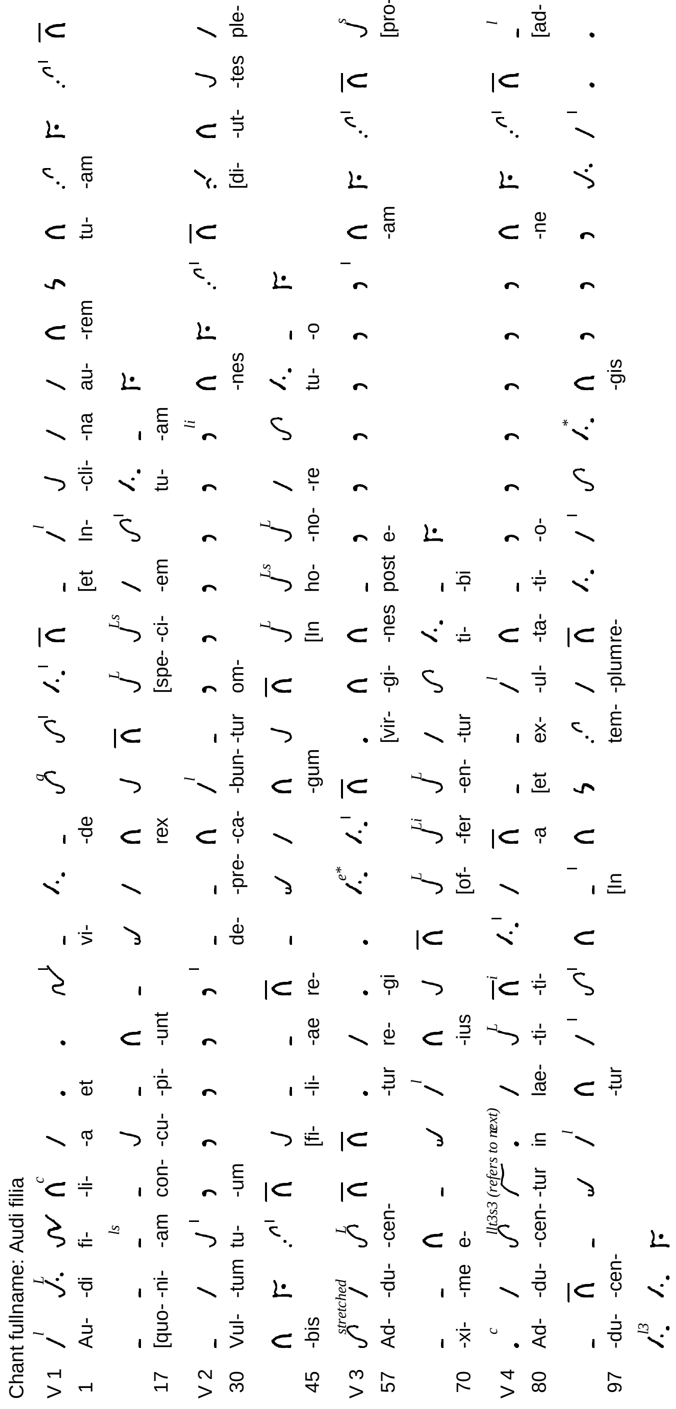

Chant fullname: Audi filia
V 1
1 Au- -di fi- -li- -a et vi- -de [et In- -cli- -na au- -rem tu- -am
17 [quo- -ni- -am con- -cu- -pi- -unt rex [spe- -ci- -em tu- -am
V 2
30 Vul- -tum tu- -um de- -pre- -ca- -bun- -tur om- -nes [di- -ut- -tes ple-
45 -bis [fi- -li- -ae re- -gum [In ho- -no- -re tu- -o
V 3
57 Ad- -du- -cen- -tur re- -gi [vir- -gi- -nes post e- -am [pro-
70 -xi- -me e- -ius [of- -fer- -en- -tur ti- -bi
V 4
80 Ad- -du- -cen- -tur in lae- -ti- -ti- -a [et ex- -ul- -ta- -ti- -o- -ne [ad-
97 -du- -cen- -tur [In tem- -plum re- -gis
stretched
ll3s3 (refers to next)

Figure 2: The chant for which verses 2 and 3 are shown in figure 1, as entered.

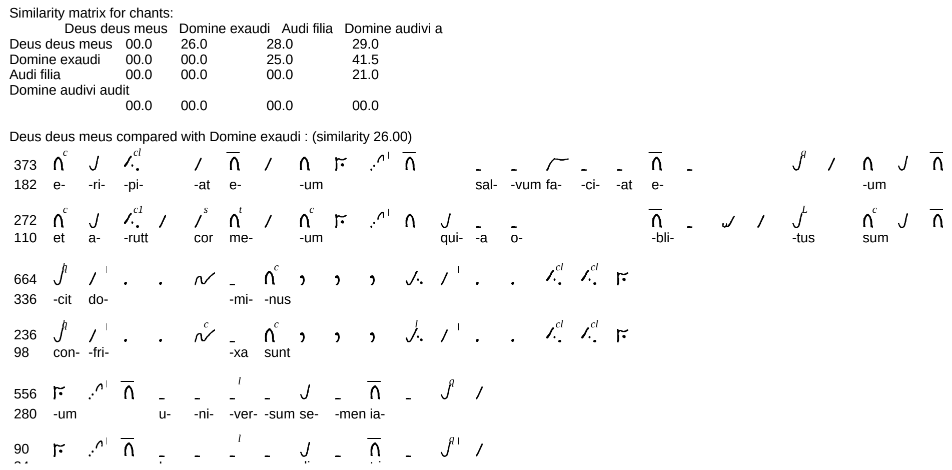
```
    n <- length(seq1); m <- length(seq2)
    S <- matrix(0, n+1, m+1)
    for (i in 1:n) for (j in 1:m) {
        if (seq1[i] == seq2[j]) sxy <- ss[seq1[i]]
        else sxy <- -2
        poss <- c(S[i, j+1] - sx, S[i, j] + sxy, S[i+1, j] - sx)
        allposs <- seq(along = poss)[poss == max(poss)]
        S[i+1, j+1] <- max(0, poss)
    }
    list(S=S, seq1 = seq1, seq2 = seq2)
}
```

Comparing two chants with 381 and 197 neumes took an hour and 45Mb of memory in S-PLUS 2000 (and had not completed in 12 hours on a machine with 'only' 64Mb), and 4 minutes in R. A C version took less than a second. (Our all-time record speed-up, [3, pp. 176–7].)

This calculation *can* be vectorized as a diagonal scan, but this was still too slow at 12 seconds in R.

Currently the matching is done in Visual Basic code: if necessary a DLL of compiled C code can be linked to Visual Basic, but compiled Basic code is far faster than R.

1.2 Results



```
Similarity matrix for chants:
          Deus deus meus  Domine exaudi  Audi filia  Domine audivi a
Deus deus meus    00.0      26.0          28.0          29.0
Domine exaudi     00.0      00.0          25.0          41.5
Audi filia        00.0      00.0          00.0          21.0
Domine audivi audit
                  00.0      00.0          00.0          00.0

Deus deus meus compared with Domine exaudi : (similarity 26.00)
```

373
182 e- -ri- -pi- -at e- -um sal- -vum fa- -ci- -at e- -um
272
110 et a- -rutt cor me- -um qui- -a o- -bli- -tus sum
664
336 -cit do- -mi- -nus
236
98 con- -fri- -xa sunt
556
280 -um u- -ni- -ver- -sum se- -men ia-
90

Figure 3: Similarity matrix between four chants, and alignment of phrases between the first pair of chants.

Some example results are shown in figures 3 and 4. At present only a handful of chants have been entered, and the analysis is by clustering phrases (see figure 4) and verses. The

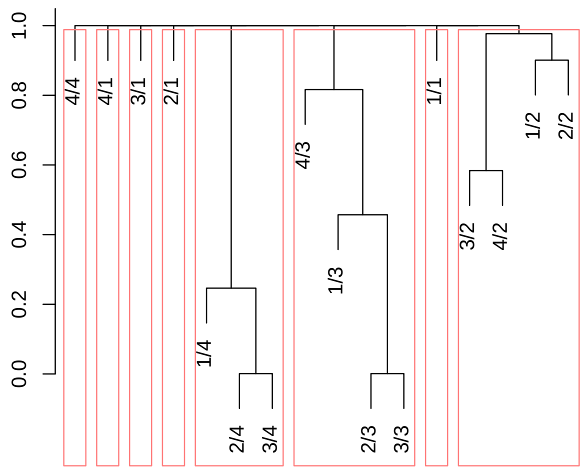


Figure 4: Dendrogram of phrases within the four verses of the chant shown in figure 2, with groups highlighted.

musicologists also want to be able to search for (near-)matches to ‘interesting’ groups of neumes.

When more data becomes available, we will use the full panoply of multivariate visualization methods, for example multidimensional scaling.

1.3 Architecture

The entered chants are kept in a database, primarily accessed from Visual Basic, but which could also be accessed from R via the `RODBC` package. Microsoft has done a good job with ‘joined up thinking’ over databases. Using ADO[2], a database can be anything from a text file to a remote Oracle database, building on ODBC[3] and OLE DB[4]. In this application early development used text files, and now a Jet database (Jet is the re-distributable backend of Access).

R is invoked via Thomas Baier’s DCOM interface[5]. The main usage is to send matrices (e.g. the similarity matrix) to R and retrieve results as needed, as Visual Basic arrays. All the analysis code can be loaded via `.Rprofile` or in a saved `.RData`: this is easier than using the DCOM interface to load a package, for example. We found it very useful to have a command to save the current R workspace, a ‘snapshot’ which can be reloaded into a normal R session to investigate further. Also, sending Visual Basic objects to R to inspect and hence help with debugging was very useful, as R has much better facilities for exploring matrices than Basic provides.

[2]ActiveX Data Objects
[3]Open DataBase Connectivity
[4]A set of database interfaces with COM components
[5]`http://cran.r-project.org/contrib/extra/dcom/`

2 R as a Database Client

The other pair of applications we will describe involve interactions with large databases, and come from projects in BDR's research group.

2.1 Insurance Rating

Fei Chen's doctorate is sponsored by English Matthews Brockman, a firm of insurance consultants, whose main business is setting premiums, e.g. for car insurance.

Large databases are available to some insurers, around a million car owners for the UK and around 30 million for the US. They have around 30 items of information on each. These are close to feasible limits (including how long people will spend filling in forms).

There are two main things to predict, the rate of claims (a Poisson regression?) and given there is a claim, the distribution of its value (gamma regression?). Although the database is large, actual information is sparse within it (few people claim per year, information rapidly becomes outdated).

Fitting models to datasets of this size will tax S implementations even on a machine with 1Gb RAM. One can often go a long way by sampling, but here the sparseness does not help. These are not really many near-duplicates, as many large databases have. That's one big issue in *data mining*: we only need large datasets if the structure is subtle. One wants 'local' models (in the sense used in local polynomial fitting in smoothing, [1, 2]).

This is work in progress, but we are contemplating fairly close links between an R client and a DBMS. The sort of ideas that one could use include:

- Use the DBMS for weighted sub-sampling.
- Ask the DBMS for an appropriate local subset of experiences. 'Locality' probably needs to be defined within the DBMS.
- Do much of the summarization in the DBMS. During the iterated weighted least squares fit of a generalized linear model, one could send the weights and get the DBMS to compute the sufficient statistics. (One might want an intermediary to ease the communication.)
- Invert the process, and use R-based functions to extend the functionality of the DBMS.

2.2 Gene Expression Arrays

As there are now thought to be 'only' around 30,000 genes in the human genome, a complete description of anyone's personal genome is thought to be feasible within a couple of years. At present gene expression chips[6] can look at up to 12,000 genes.

Data is available at various levels from the chip, from the raw fluorescence image (50Mb) through features (20Mb) and summaries at various levels (e.g. at gene level, 2Mb). A designed experiment will have a few tens of chips, and research projects can consist of many experiments.

[6]Affymetrix is the manufacturer of those whose data we will be using.

Data on that scale needs to be handled in databases. There are also large amounts of ancillary information needed to structure the data (10Mb per chip design). The pharmaceutical companies tend to use Oracle, and so we intend to too. (Oracle tables may be the interchange format.)

We are currently negotiating access to such data for Wiesner Vos's project. Given Wiesner's interests, the analysis will be in S on Windows or Linux, and delivery needs to be on Windows. Once we have the legalities out of the way, the issues will be to extract subsets efficiently to an R / S-PLUS 6.0 or 2000 client.

3 Conclusions

Data entry and extraction is an increasingly important part of statistical applications. We have had an interest in databases for a long time (RMR is a former mainframe programmer), but several projects seem urgently to need such tools. Computers get bigger and faster, but at least for now so do applications, at an even faster rate. As we have hinted, in many domains it may not be long before all feasible information is available, at some cost.

S implementations have long been regarded as very weak for large-scale problems: this is a criticism that has been levelled at S-PLUS for a decade, even though S version 4 has been claimed to help address this. Often the criticism is unfair: regressions on 10,000 subjects or analysing a clinical trial of 17,000 [3, §7.2] do not normally make much sense. Large datasets are rarely homogeneous, and if they are, a small subset suffices.

Borrowing strength (a Tukey-ism) seems *the* way forward. We have moved elements of the chant system between Basic, C and R, and between text files and relational/hierarchical databases as was most convenient during the course of the project.

Setting up these connections has been fun and frustrating, in equal measure. It is all too easy for statistical thinking to get swamped by programming tasks.

Acknowledgements

RMR's work on chant is in collaboration with Dr Emma Hornby and Prof. John Caldwell pf the Oxchant group[7] and was partially supported by the Music Faculty of the University of Oxford.

References

[1] Clive Loader. *Local Regression and Likelihood*. Springer, New York, 1999.

[2] Jeffery S. Simonoff. *Smoothing Methods in Statistics*. Springer, New York, 1996.

[3] W. N. Venables and B. D. Ripley. *S Programming*. Springer, New York, 2000.

[4] Michael S. Waterman. *Introduction to Computational Biology. Maps, Sequences and Genomes*. Chapman & Hall / CRC Press, Boca Raton, 1995.

[7] `http://www.music.ox.ac.uk/oxchant/`