sabreR: Grid-Enabling the Analysis of Multi-Process Random Effect Response Data in R

Daniel Grose[1], Rob Crouchley[1], Ties van Ark[1], Rob Allan[2], John Kewley[2], Adam Braimah[2] and Mark Hayes[3]

[1] University of Lancaster
[2] CCLRC Daresbury Laboratory
[3] University of Cambridge

Email address of corresponding author: `d.grose@lancs.ac.uk`

Abstract. The SABRE (Software for the Analysis of Binary Recurrent Events) program has been extended to run on parallel architectures. A web service interface has been provided using the GROWL (Grid Resources on Workstation Library) toolkit to securely access SABRE functionality deployed on the grid. This interface has been incorporated into an R package (sabreR) so that users can configure and analyse SABRE models and results from within R. The package allows for multiple SABRE models employing both serial and parallel SABRE implementations to be used simultaneously.

Sabre

Overview

Sabre (1, 2) (Software for the Analysis of Binary Recurrent Events), is a program specifically designed for the analysis of binary, ordinal, count recurrent events. Such data are common in many surveys either with recurrent information collected over time or with a clustered sampling scheme. As such, Sabre is particularly appropriate for the analysis of work and life histories, and has been used intensively on many longitudinal datasets.

Sabre can be considerably faster than conventional statistical modelling software. For example, with a data set of 11341 observations of 26 variates Sabre ran 435 times faster than an equivalent analysis made within Stata (3) using gllamm (4). Sabre's favourable performance is a result of several factors, for example analytical rather than numeric calculation of the derivatives.

Some of the random effects statistical models estimated by social scientists are computationally demanding on large data sets. In addition for substantive reasons, social scientists need the desirable features of random covariate parameters (i.e. acknowledging more stochastic complexity) and multiprocess capability (i.e. acknowledging the interdependencies between different aspects of behaviour). Even with the reduced analysis time obtained by using Sabre, these models may take many days to estimate. To mitigate these prolonged analysis times, Sabre version 4.0[1] has been developed with support for parallel computers. This provides an almost linear speed-up in terms of number of processes. Executing Sabre on a parallel system where many hundreds of processes may be available makes the analysis of complex models a possibility.

Performance and Timings

Tables I II and III contain results showing the analysis time of various data and models for Stata, gllamm and Sabre employing 1, 2, 4, 8 and in one case 16 processors. All of the computations were performed on Lancaster University's High Performance Computing Facility which, at the time the analyses were undertaken, consists of an array of 64 bit Sun-Blade workstations[2]. Each workstation has 1 gigabyte of memory and communications between each system run at 100 megabits per second.

Data	*Obs*	*Vars*	*Kb*	*Stata*	*gllamm*	*Sabre(1)*	*Sabre(2)*	*Sabre(4)*	*Sabre(8)*
hsb (16)	7185	15	1172	**01"**	20' 51"	06"	04"	03"	02"
thaieduc1 (15) thaieduc2 (15)	8582	4	378	11"	4' 52"	01"	01"	01"	01"
teacher1 (14) (13) teacher2 (14) (13)	661	3	22	n/a*	1' 14"	00"	00"	01"	01"
racd(dvisits) (13)	5190	21	1090	52"	18' 24"	03"	02"	01"	02"
racd(prescribe) (13)	5190	21	1090	42"	15' 11"	03"	02"	01"	02"
visit-prescribe (13)	10380	26	2717	n/a**	45hr 15'	2' 21"	1' 11"	36"	20"

Table I: Timing Comparisons for Cross Sectional Data (2).

Data	*Obs*	*Vars*	*Kb*	*Stata*	*gllamm*	*Sabre(1)*	*Sabre(2)*	*Sabre(4)*	*Sabre(8)*
pefr (17)	34	4	2	**00"**	29"	00"	00"	01"	01"
nls(wage) (18)	18995	20	3859	**03"**	2hr 12'	27"	15"	08"	05"
growth (19)	153	8	14	**00"**	1' 00"	00"	00"	01"	01"
nls(union) (18)	18995	20	3859	2' 02"	30' 04"	05"	03"	02"	02"
schiz (20) (21) (22) (23)	1603	8	140	n/a*	2' 24"	00"	00"	01"	01"
dvisits (23) (24)	2227	10	242	39"	9' 07'	02"	02"	01"	01"
filled (25)	390432	94	367556	59hr 52"	> 3mnth	34' 38"	18' 51"	11' 03"	7' 01"
lapsed (25)	390432	94	367556	67hr 31"	> 3mnth	29' 414"	16' 20"	9' 45"	6' 21"
filled-lapsed (25)	780864	261	2134413	n/a**	> 3years	54hr 29'	32hr 5'	18' 49"	11' 58"
union-wage (18)	37990	25	9683	n/a**	n/a***	18hr 21'	9' 13"	4' 41"	2' 26"

Table II: Timing Comparisons for Longitudinal Data (2).

* Stata 9 cannot estimate random effects ordered response models using quadrature.
** Stata 9 cannot estimate bivariate random effects models using quadrature.
*** Unexpected failure.

Boldface entries in the *Stata* column of tables I and II indicate that the Stata `xtreg` command was employed to estimate a random effects linear model using Maximum Likelihood Estimation (MLE) which, for normal distributed random effects, has a closed form for the likelihood integration. The remainder of the results employ quadrature: the number of quadrature points used varying in each example according to the needs of accuracy[3].

Clearly Sabre out-performs gllamm on all data sets and Stata only out performs Sabre when MLE is employed. Sabre out performs both gllamm and Stata when quadrature is employed. A

[1] For the remainder of this work, Sabre is used to refer to Sabre version 4.0 unless otherwise stated.
[2] The Lancaster University High Performance Computing Facility is being replaced with higher specification systems in April 2006.
[3] The adaptive quadrature algorithm in glamm was not employed.

useful illustration of relative performance for such a case is provided by the lapsed and filled-lapsed data sets in table II. These data are from a study providing the first estimates of the determinants of employer search in the United Kingdom using duration modelling techniques and involve modelling a job vacancy duration until it is either successfully filled or withdrawn from the market. For full details of this analysis see (25).

Data	*Obs*	*Vars*	*Size*	*Stata*	*gllamm*	*Sabre(1)*	*Sabre(2)*	*Sabre(4)*	*Sabre(8)*	*Sabre(16)*
aus	3665704	53	2Gb	10183'	> 6months	62'	32'	16'	9'	5'

Table III: Timing Comparisons for Large Data Set Demonstrating Approximately Linear Speed-Up (2).

Table III concerns the analysis of a large data set (3665704 observations with 53 variables). The results highlight the approximately linear improvement in performance of Sabre as the number of processors employed increases. That Sabre running on 16 processors is >2000 times faster than Stata provides persuasive evidence of the benefits of employing parallel Sabre for this kind of analysis.

For more information regarding the above comparisons and detailed theory of the models employed by Sabre see (2) and (1)

Integrating Sabre functionality into statistical environments

Serial Sabre provides a text based Graphical User Interface (GUI) which allows the user to configure and run Sabre models and examine the results obtained from an analysis. Both the serial and parallel versions allow a Sabre script (the text of a GUI session) to be processed from the command line and produce results to an output file. However, it is desirable, for a number of reasons, to provide access to Sabre functionality from within a statistical environment such as R (5) or Stata. Doing so allows the user to prepare the model data, configure models and analyse results using the native data structures and extensive functionality of the packages and eliminates the need for a user to learn the syntax of the Sabre environment.

Most common statistical packages offer facilities and tools that allow a programmer to add functionality to their environments. To use the features of existing software in such a way, it is usually necessary to have available the source code or a pre-compiled library containing the required functionality. However, neither the serial or parallel versions of the Sabre code were developed with reference to integration into third party applications and the Sabre methods are not available directly from a library. In addition, the benefits of parallel Sabre are only available when it is executed on a remote High Performance Computer (HPC). This makes integrating Sabre into a statistical package challenging. However, such issues are not unique to Sabre, and particularly in the context of grid computing, legacy application functionality is often required to be accessed from within other software components. For this reason, GROWL (Grid Resources on a Workstation Library) (6, 7, 8) was developed.

Grid Resources on a Workstation Library

Overview

Amongst other components, GROWL provides a client server system in which the server can host arbitrary services that provide a SOAP (9) interface. Client access to these services is over a secure (PKI/SSL) connection to a single port on the host system. Clients are authenticated to the server using their distinguished name extracted from a certificate provided by a trusted certificate authority, such as the National Grid Service. Client access to particular services can then be granted/denied based upon client status.

Since the server is provided as a stand-alone web service, client access to a specific service becomes virtually transparent. Furthermore, significant advantages are gained in having the server as a stand-alone service. Firstly, the server has persistence (traditional web services typically do not) providing meaningful state for managing clients and services. Secondly, it eliminates any administrative dependency on services such as HTTP(S) allowing many of the difficulties associated with institutional firewalls to be overcome.

In the context of the grid, the architecture allows a developer to create client-side interfaces to grid facilities hosted as web services. This is significant since the need for grid middleware knowledge is restricted to developers providing the grid service. In addition, administration/security issues are devolved from the grid developer to the server administrator. This "architecture enforced demarcation" categorises the development associated with providing grid services into classes very much in keeping with those identified in Foster and Kesselman (10).

GROWL Server Architecture

The GROWL client/server has a three tier architecture (figure 1) comprising a client, server and services tier.

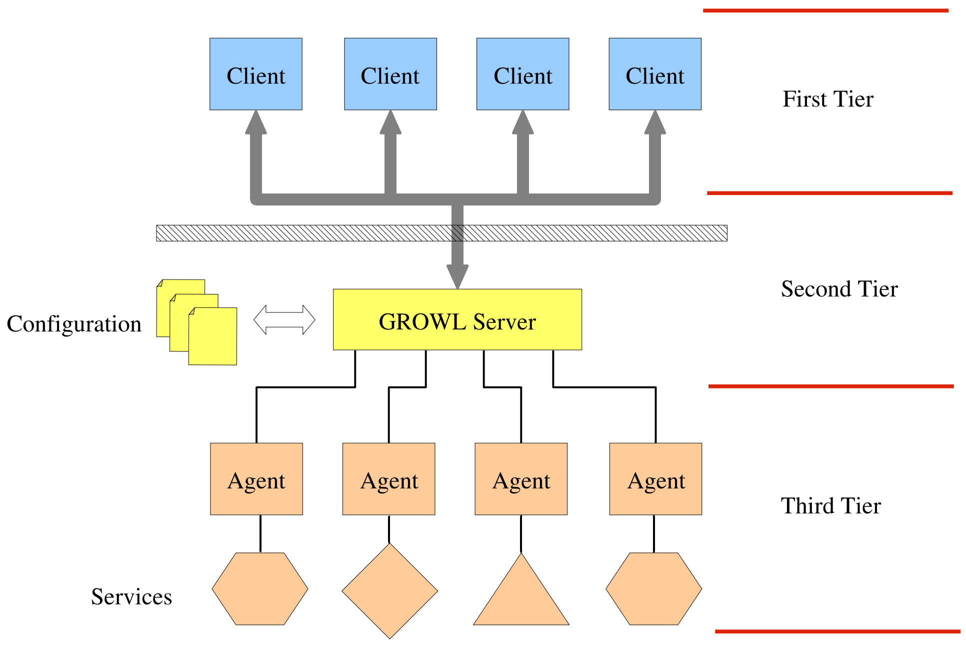

Figure 1. GROWL Client - Server Architecture

The first tier consists of the client interfaces, specialised to specific client application requirements, and integrated into the client server architecture using modules created from service interface definitions published using the Web Services Definition Language (WSDL). The WSDL are generated and published by the developers of the services represented in the third tier of the architecture.

The second tier consists of the GROWL server. The main functions of the server are threefold:

1. Authentication of clients.
2. Hosting services by acting as a proxy for the service interface.
3. Mapping of client requests to specific service instances.

The third tier consists of the services themselves. A service is defined by its interface which is published using the WSDL. Importantly, an individual service may be implemented in a number of different ways, the particular implementation varying according to the requirements of the system(s) hosting and the client(s) accessing it. It may, of course, also vary in time as the requirements of a service implementation change. Services are created by service developers and the interface definitions for a service can be created automatically from existing code using GROWL utilities.

The key advantages of this three tier client/server architecture are:

- Clients, server and services may be upgraded or replaced independently.
- A single interface may correspond to many service implementations.
- All services are accessed via a single port.
- Services have persistence.
- Developers of client applications can program against an interface in a language and platform independent manner and need no understanding of the service logic.
- Developers of services need not be aware of the client application logic and do not require an understanding of web services.

Migrating Legacy Systems into the GROWL Client-Server Model

There are five actors associated with developing, deploying, and securing services hosted by the GROWL client/server. They are:

- The client.
- The client application developer(s).
- The service developer(s).
- The server administrator.
- The certificate authorities used to enable authentication and realise secure communication.

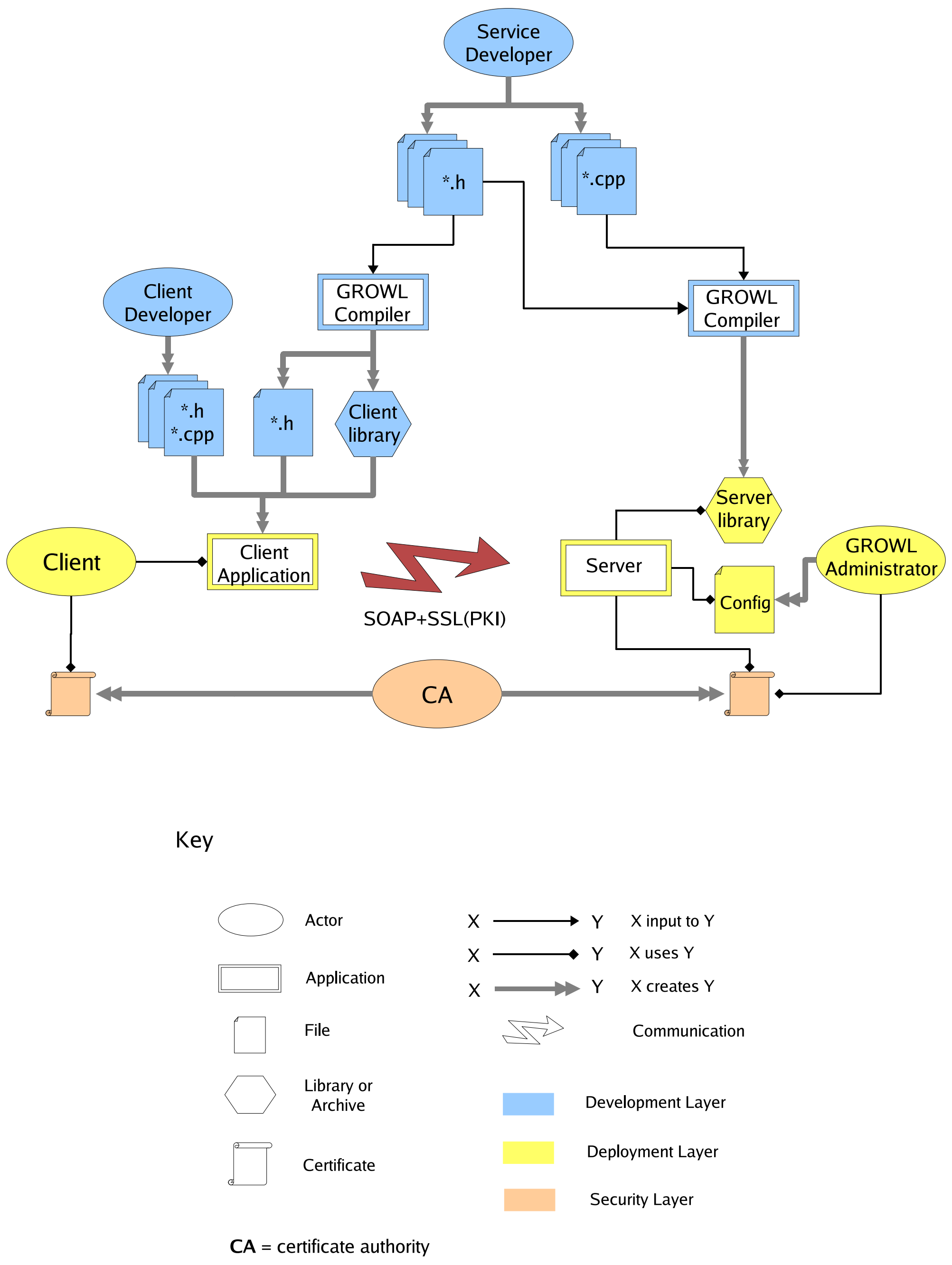

Figure 2. Development, deployment and security of components. Demarcation of actors.

Of particular interest is the demarcation of these roles in the process of hosting and accessing services using the GROWL client/server architecture. The involvement of each actor in the process of developing, deploying and making secure a service is demonstrated in figure 2.

The development of the a service and client applications that employ it are undertaken by the server and client developers respectively. The server developer typically has an existing library/application which provides the service logic. Using the definition of the service interface and implementation in conjunction with the GROWL developer compiler a server library is generated. A client developer can take the same service interface definition and uses the growl compiler to generate a client side library and header files. These are then used to develop a client application using the same interface as employed if the service were hosted on the client system.

The server administrator is responsible for deploying the server, adding and removing services and administrating client access. Each service corresponds to a service library provided by the service developer and is mapped to a service name from within an XML based configurations file maintained by the administrator. A list of authorised clients for each service along with information regarding each active client session (generated by the server) is also present within this file.

Client identification and client/server authentication is via the use of certificates. Certificates are issued by a certificate authority in the normal manner and employed within the client/server architecture for secure SSL/PKI communication.

Sabre R interface

Overview

The Sabre extension to R was developed using GROWL components and R scripts combined into an R package (sabreR). In addition, parallel Sabre is hosted on the grid using a GROWL server. The main advantages of adopting this approach are

- The Sabre wrapper interface developed using GROWL is identical for both serial and parallel Sabre
- The GROWL server provides secure/authenticated access to parallel Sabre on the grid by employing the wrapper interface
- The GROWL server exposes Sabre functionality as a web service, thus eliminating many of the problems commonly associated with institutional firewalls and account management.
- The user does not require an account on the system hosting parallel Sabre
- A user can start a grid hosted Sabre session and then terminate the R session without cancelling the Sabre analysis. They can then recover the session for later use, even on a different client system.

Features

Being able to access Sabre functionality from within R has a number of advantages. In particular, the R user can undertake an Sabre analysis using native R data structures and commands with which they are familiar. This allows the preparation and analysis of Sabre input and output to be integrated into existing methods and work flows that might be already undertaken within the R environment. In addition to this, use of the GROWL API and server allows a user to have multiple concurrent Sabre models within a single R session. Because the GROWL facilities are multi-threaded, control is returned to the user after each Sabre command, even if the command is still being processed. These features allow a user to easily study the effects of modifications in model parameters and/or data by comparing estimates from multiple models within the same R session. The latter two features were not in the original project specification but are a natural outcome of employing GROWL.

Interface

The sabreR commands are designed to be similar, in terms of case, naming convention and argument handling, to the native R commands and to those found in R packages. In addition, all data used in conjunction with a Sabre model is organised using native R data structures. The following demonstrates a typical sabreR session

```
> library(sabreR)                                  # load the sabreR library
> sabre0<-sabre.session();                         # create a new sabre model
> trade.union<-read.table(‘‘./TradeUnion.table’’)  # read the data into a data frame
> names(trade.union)                               # show the variates
 [1] "CASE" "YEAR" "AGE"  "EVNO" "SUPR" "HRS"  "NOEM" "SEX1" "TU"   "PROM"
[11] "SC80"
> sabre.data(sabre0,trade.union)
> sabre.display.variates()

    Name            Levels    Type
    --------------------------------
    cons                1      X
    case                1      X
    year                1      X
    age                 1      X
    evno                1      X
    supr                1      X
    hrs                 1      X
    noem                1      X
    sex1                1      X
    tu                  1      YVAR
    prom                1      X
    sc80                1      X
    fnoem               5      X
    fsc80               6      X

> plot(trade.union)                                # plot the data
> sabre.y.variate(sabre0,’’tu’’)
> sabre.factor(sabre0,’’noem’’,’’fnoem’’)
> sabre.factor(sabre0,’’sc80’’,’’fsc80’’)
> sabre.display.model()

    X-vars            Y-var
    ------------------------------
    year              tu
    age
    fnoem
    fsc80

    Univariate model
    Standard logit

    Number of observations             =    1633
    X-var df           =    12
    Log likelihood =     -1073.0110     on    1621 residual degrees of freedom

> sabre.lfit(sabre0,''year'',''age'',''fnoem'',''fsc80'') # linear fit

    Iteration       Log. lik.       Difference
    __________________________________________
        1         -1131.9093
        2         -1073.2798        58.63
        3         -1073.0111        .2687
        4         -1073.0110       0.1204E-03
        5         -1073.0110       0.4700E-09

> sabre.display.estimates(sabre0)

   Parameter              Estimate         Std. Err.
    ___________________________________________________
    year                  -0.16136E-01      0.54224E-02
    age                    0.32899E-01      0.69741E-02
    fnoem       ( 1)       -1.3945           .80939
    fnoem       ( 2)       -.78157           .47819
    fnoem       ( 3)      -0.35445E-01       .47920
    fnoem       ( 4)        .14679           .46976
    fnoem       ( 5)       0.48744E-01       .46787
    fsc80       ( 1)        .00000          ALIASED [I]
    fsc80       ( 2)        .39780           .29480
    fsc80       ( 3)       -.17355           .30840
    fsc80       ( 4)        .60508           .28237
    fsc80       ( 5)        .49547           .29331
    fsc80       ( 6)        .51569           .32419

> sabre.case(sabre0,''case'')
> sabre.fit(sabre0,''year'',''age'',''fnoem'',''fsc80'')
> # NB returns control to user immediately even though analysis still running
> sabre.display.estimates(sabre0)
        *** Sabre analysis still in progress ***
> #     ... some time later ...
> sabre.display.iterations(sabre0)

   Initial Homogeneous Fit:

    Iteration       Log. lik.       Difference
    __________________________________________
        1         -1131.9093
        2         -1073.2798        58.63
        3         -1073.0111        .2687
        4         -1073.0110       0.1204E-03
        5         -1073.0110       0.4700E-09

    Iteration       Log. lik.         Step      End-points     Orthogonality
                                     length    0          1      criterion
    ________________________________________________________________________
        1         -917.86146        1.0000    fixed  fixed       6.5174
        2         -878.33512        1.0000    fixed  fixed       19.983
        3         -868.98256        1.0000    fixed  fixed       4.6722
        4         -867.52529        1.0000    fixed  fixed       5.2621
        5         -867.20337        1.0000    fixed  fixed       11.115

> # ... user can be doing other things within R whilst sabre analysis continues
> sabre.display.estimates(sabre0)
```

```
    Initial Homogeneous Fit:

    Iteration       Log. lik.       Difference
    __________________________________________
        1          -1131.9093
        2          -1073.2798        58.63
        3          -1073.0111        .2687
        4          -1073.0110       0.1204E-03
        5          -1073.0110       0.4700E-09

    Iteration       Log. lik.         Step      End-points     Orthogonality
                                     length    0          1      criterion
    ________________________________________________________________________
        1          -917.86146        1.0000    fixed  fixed       6.5174
        2          -878.33512        1.0000    fixed  fixed       19.983
        3          -868.98256        1.0000    fixed  fixed       4.6722
        4          -867.52529        1.0000    fixed  fixed       5.2621
        5          -867.20337        1.0000    fixed  fixed       11.115
        6          -867.06558        1.0000    fixed  fixed       5.9164
        7          -866.93034        1.0000    fixed  fixed       6.8780
        8          -866.93024        1.0000    fixed  fixed       11.479
        9          -866.93024        1.0000    fixed  fixed
```

Notice how in the example the first argument to all of the Sabre commands is a Sabre session. This is how sabreR distinguishes between multiple Sabre models within a single R session. Furthermore, notice that the first call to sabre.display.estimates resulted in a warning that the analysis was not yet complete. This demonstrates the multi-threaded nature of the sabreR package. Finally, use of the sabre.display.iterations allows a user to keep track of each Sabre analysis that is currently running.

How would the above session differ if a parallel Sabre analysis was being executed on a grid resource ? The following demonstrates how little additional effort is required:

```
> nwg<-grid.resource(‘‘~/smith.pem’’,’’~/smith.pem’’,
+ ’’growl.lancs.ac.uk:50000’’,’’~/smith.passwd’’)
> sabre0<-sabre.session(nwg)  # this time create a sabre session with a grid resource
> # ..... continue as before
```

In this example, a grid resource is acquired by passing the grid.resource function the location of the users certificates, a file containing the users password and the name of the system hosting the GROWL server. Sabre sessions are created as in the previous example except that the grid resource is passed to the sabre.session function. Any ensuing Sabre commands are identical those as used when using a local (serial) session of Sabre.

If a user leaves the R session while a grid Sabre session is active, they can return to it later. The sabreR package offers two simple functions for retrieving grid Sabre sessions. These are outlined in the following example.

```
> library(sabreR)
> nwg<-grid.resource(‘‘~/smith.pem’’,’’~/smith.pem’’,
+ ’’growl.lancs.ac.uk:50000’’,’’~/smith.passwd’’)
sabre.current.sessions(nwg)

            started       last command
1  02/01/2006 13:01   02/01/2006 14:27
```

```
2  02/01/2006 13:07   02/01/2006 13:54
3  02/01/2006 13:08   02/01/2006 13:57
4  02/01/2006 13:11   02/01/2006 14:03

> sabre0<-sabre.recover.session(nwg,3) # recover session 3
```

Conclusions

Support for parallel processors introduced in version 4.0 of Sabre dramatically reduces the run times of statistical analysis of multi-process random effect response data. An almost linear improvement in analysis speed with number of processors employed makes it particularly suitable for deployment on a computational grid. However, there are a number of significant benefits to be had by integrating Sabre functionality into existing statistical environments such as R. However, doing so in a manner that makes use of the underlying grid resources as transparent as possible poses a number of technical challenges. These challenges are far from being unique to Sabre and consequently the GROWL library has been developed to provides the programmer with an API that generalises the solutions to these challenges. A three tier client/server architecture is provided by GROWL and this has been employed to host a generic Sabre interface defined in WSDL and access to this service has been integrated within R. Work is currently being undertaken to provide access to the Sabre services from within Stata.

Acknowledgements

- David Stott. Lancaster University Centre for e-Science. Development of Sabre version 3.0 (1996) and Sabre version 3.1 (1997) adding support for Poisson distribution and Markov model. Development of Sabre version 4.0 (2005) adding bivariate and trivariate modelling.
- John Pritchard. Lancaster University Centre for e-Science. Development of serial and parallel implementation of Sabre 4.0 (2005).
- Jon Barry, Brian Francis and Richard Davies. Lancaster University Centre for Applied Statistics. Development of Sabre version 2.0 (1989).

References

"An OGSA Component-Based Approach to Middleware for Statistical Modelling". ESRC Report reference number RES-149-25-0010.

http://sabre.lancs.ac.uk/timings.html

Stata Statistical Software. http://www.stata.com

Rabe-Heketh S., Skrondal A., Pickles A. "GLLAMM Manual". U.C.Berkeley, Division of Biostatistics Working Paper Series. Working Paper 160. 2004. http://www.gllamm.org/docum.html

The R Project. http://www.r-project.org

Hayes M., Morris L., Crouchley R., Grose D., van Ark T., Allan R., Kewley J. "GROWL: A Lightweight Grid Services Toolkit and Applications". 2005

Crouchley R., van Ark T., Pritchard J., Kewley J., Allan R., Hayes M., Morris L. “Putting Social Science Applications on the Grid”. 2005.

Grid Resources on a Workstation Library (GROWL). http://www.growl.org.uk

Van Engelen R., Gallivan K. “The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks”. 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), May 21-24, Berlin, Germany. 2002.

Foster I., Kesselman C. (editors) “The Grid: Blueprint For A New Computing Infrastructure”. Morgan Kaufman Inc. 1998.

Sabre in R. http://www.ncess.ac.uk/research/pdp/index.shtml#ogsa

Cameron, A.C., Trivedi, P.K., Milne, F., Piggott, J. “A microeconometric model of the demand for Health Care and Health Insurance in Australia”. Review of Economic Studies, 55, 85-106. 1988

Cameron, A.C., Trivedi, P.K. “Regression Analysis of Count Data”. Econometric Society Monograph No.30, Cambridge University Press. 1998.

Rowan, B., Raudenbush, S., Cheong, Y. (1993). “Teaching as a non-routine task: implications for the organizational design of schools” Educational Administration Quarterly, 29(4), 479-500. 1993.

Raudenbush, S.W., Bhumirat, C., 1992. “The distribution of resources for primary education and its consequences for educational achievement in Thailand”. International Journal of Educational Research, 17, 143-164. 1992.

Raudenbush, S.W., Bryk, A.S. “Heirarchical Linear Models”. Thousand Oaks, CA. Sage. 2002.

Bland, J. M., Altman, D. G. “Statistical methods for assessing agreement between two methods of clinical measurement”. Lancet, 1, 307-310. 1986.

Stata Longitudinal/Panel Data. Stata Reference Manual, Release 9. Stata Press, StataCorp LP, College Station, Texas. 2005.

Snijders, T. A. B., Bosker, R. J. “Multilevel Analysis”. London: Sage. 1999.

Gibbons, R. D., Hedeker, C., Waterneaux, C., Davis, J.M. “Random regression models: A comprehensive approach to the analysis of longitudinal psychiatric data”. Psychopharmacology Bulletin, 24, 438-443. 1998.

Gibbons, R. D., Hedeker, C. “Application of random effects probit regression models”. Journal of Consulting and Clinical Psyschology. 1994.

Hedeker, D., Gibbons, R.D. “Applied longitudinal data anlaysis”. Chichester, UK, Wiley. 1996.

Rabe-Hesketh, S., Skrondal, A. ”Multilevel and Longitudinal Modelling using Stata”. Stata Press, Stata Corp, College Station, Texas. 2005.

Winkelmann, R. “Healthcare reform and thye number of doctor visits: an econometric analysis”. Journal of Applied Econometrics, 19, 455-472. 2004.

Andrews M. J., Bradley S. Stott D., Upward R. “Succesfule Emploer Search ? An Empirical Analysis of Vacancy Duration using Micro Data” Online paper hosted at http://www.lancs.ac.uk/staff/ecasb/papers/vacdur_economica.pdf.