Data objects and model formulæ for the future: some proposals

J.K. Lindsey

Biostatistics, Limburgs Universitair Centrum, Diepenbeek

Email: jlindsey@luc.ac.be

Abstract

Rectangular data matrices and linear formula specification used in current statistical software packages are two of the major impediments to developing and implementing advanced statistical models.

A set of three data objects is proposed to replace the rectangular data matrix. Although developed for repeated measurements data, these have much wider application, such as to independent data and to time series. The three objects contain, respectively, the response variable, with all of the information required to specify its probability distribution, the inter-unit covariates, and the intra-unit covariates, necessary for any set of models of interest.

As an extension to the Wilkinson and Rogers (1973) notation widely used to specify the linear part of regression models, two general procedures for describing nonlinear models are proposed. The first simply uses the builtin function construction facilities of R or S, whereas the second is more closely related to the Wilkinson and Rogers notation, but allows unknown parameters to have individual names. Then, such a formula can be automatically transformed into a suitable function, with the software distinguishing between knowns and unknowns.

Both the data objects and the model formulation have been implemented in R. A wide collection of model-fitting functions for nonlinear regression, including repeated measurements, based on them is available.

KEYWORDS: Clustering, data objects, durations, model specification, nonlinear model, repeated measurements, time series, time-varying covariates.

1 Introduction

In this paper, I discuss some limitations of currently available software with respect to data handling and model specification and make suggestions for improvements. I have implemented all of my proposals in a library of functions for R (Ihaka and Gentleman, 1996), demonstrating that they are all feasible and useful.

One of my concerns is with how currently available techniques constrain statistical users in a mind-frame such that they must think in fixed and old-fashioned ways not suitable for modelling the data now being collected in science and industry. Current computing power has revolutionized data collection; it should also fundamentally change the ways in which such data are analyzed. Some may claim that modern nonparametric and Bayesian procedures are doing this. However, the first are essentially either descriptive methods (such as kernel density estimation and other smoothers) or designed for testing hypotheses (such as Cox proportional hazards), not for understanding the mechanisms by which the data were generated. The second is a 'bigger is better' philosophy that has not (yet?) fundamentally changed the way data are actually treated because of the additional assumptions and computer time required.

Two basic theses underly my discussion in this paper:

1. the classical rectangular data matrix is inadequate to handle many modern data types;
2. the now standard Wilkinson and Rogers (1973) formulæ for linear models need to be extended, in user-friendly ways, to the specification of nonlinear models.

All current major statistical software systems oblige users to coerce their data into a rectangular form, generally with individuals as rows and variables as columns. Even planned experiments are not always balanced in this way. Longitudinal repeated measurements studies may have highly unequal numbers of observations per subject. Missing values and dropouts may occur.

When only a desk calculator was available in data analysis, estimation was usually only feasible for simple linear models. Everyone agreed that they were often poor approximations to the phenomena under study, but nothing else was possible. Current commercial software generally retains this 'desk calculator' philosophy, simply allowing users to handle larger data sets in less time. The one major exception, generalized linear models, has had a surprisingly narrow impact on current practice.

In a certain sense, modern statistics is fundamentally anti-scientific, attempting to impose its 'generally applicable' methods in all circumstances instead of trying to understand each specific scientific problem and to adapt specific procedures to it. The classical linear model is the archetypical case: it is widely believed that most problems can be transformed in some way so that least-squares multiple regression can provide a solution. Empirical models are preferred to mechanistic ones, the extreme example being nonparametric statistics. Those areas of statistics that have escaped from this rule (for example, statistical mechanics, population and molecular genetics, or pharmacokinetics) have almost exclusively been developed by nonstatisticians.

In what follows, I shall concentrate on the handling of repeated measurements types of data. This has the advantage that it covers both the classical independent observations and time series as special cases, as well as providing simple extensions to spatial data. However, this also means that I make no pretension of universality. In any case, it is doubtful that one type of data structure could ever be designed to handle all conceivable cases.

Table 1: An example of repeated response data for three selected individuals. First line: times, measured from randomization (with negative times being pre-randomization baseline values). Second line: response measurement. (Lindsey, 1999, p. 400.)

-27	-13	28	56	84	168	259	331	427	504	672	771	834	945	1008	1092
561	334	157	374	191	465	125	212	232	177	98	207	127	202	143	174
1289	1306	1351													
216	245	237													

-14	-6	58	253	358	508	574	672	855	924
429	587	446	269	131	50	145	634	273	144

-14	-7	0	56	84	168	336	420	504	672	756	840	924	1000	1135	1260
231	312	123	127	297	337	225	312	178	111	97	133	239	151	115	297
1280	1337														
141	113														

2 Data objects

2.1 Types of variables

The classical rectangular data structure does not distinguish between response and explanatory variables. This is justifiable in certain situations, as for strictly multivariate data or for graphical models. However, in most contexts, these two classes of variables are fundamentally different; this should be reflected in the data objects containing them.

2.1.1 Response variables

The response variable is that for which a (conditional) probability distribution is assumed in some given set of models. This can entail a considerable amount of supplementary information. Consider two examples. The usual longitudinal repeated measurements have times associated with them, as in Table 1. These data clearly do not have a rectangular structure and forcing it upon them would be difficult and uninformative. In contrast to much of classical time series, here there are unequal numbers of observations per individual and the time points are both unequally-spaced and different for each individual.

Times between events, as in Table 2, provide a second example. If, in the previous example, the times might possibly have been fixed in advance, here it is impossible. It is the random times that are of interest. An additional complication is that observation of an individual may not terminate at an event, so that last recorded time may be censored.

Thus, at each observation point, we must collect more than just a univariate response value.

Table 2: An example of repeated response data for six selected individuals: times between repeated events, with the final value indicating whether the last time is censored or not. (Lindsey, 1999, p. 436.)

```
 5 13  0
12  4  2  0
23  0
 3  3  2  4 14  4  0
 3 13  7  1
 3  6 12  2  0
```

For repeated measurements, at least one of the following must also be available:

1. Times (if longitudinal).
2. Location (if spatial).
3. Nesting indicators (if clustered).
4. Censor indicators (if durations).
5. Binomial denominators (if binary).
6. Unit of measurement (if continuous).
7. Jacobian of any transformation (if continuous).
8. Weights.

Many of these will also be required even for the simpler independent observations.

All of this information is necessary in order to construct the statistical model based on a probability distribution, even if no covariates are present. It should all be stored together, along with the corresponding response values, in one data object.

Nevertheless, the first three of these types of information have a somewhat ambiguous status. They are required with the responses in order to define the dependencies among them. However, they may also be necessary, in some contexts, as explanatory variables: times or locations for trends, nesting for fixed effects. I shall return to this point below.

2.1.2 Covariates

For independent observations and for time series, covariates generally have a simply structure, being in one-to-one correspondence with the response values. The same is not true for repeated measurements. Some distinguish among the individuals (inter-unit or time-constant), staying identical for all responses on each individual whereas others (intra-unit or time-varying) may change along with the responses.

Table 3: An example of repeated time-varying covariates for the three individuals in Table 1. First line: times, measured from randomization, but at different moments than the response. Second line: dose. (Lindsey, 1999, p. 407.)

0	28	58	85	113	159	203	333	375	585	591	1306
1.0	1.167	1.333	1.5	1.667	1.833	2.0	1.833	2.0	0.0	2.0	0.0

0	218	312	352	403	406	973
1.0	1.2	1.4	1.6	1.8	2.0	0.0

0	29	57	87	119	164	203	241	287	818	835	1280
1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	0.0	2.4	0.0

Inter-unit covariates In the repeated measurements context, the rectangular data matrix paradigm forces the user to replicate any inter-unit covariates as many times as there are repeated responses. Not only is this wasteful of storage space (and copying time in certain statistical software), but often statistical calculations can be made more efficiently if each such covariate has only one value per subject.

Intra-unit covariates Consider again the example responses above in Table 1; the time-varying covariate, dose, is also available. The values are presented in Table 3. Here, not only is the covariate recorded at unequally-spaced times, different for each individual, but the times are different than those for the responses. Thus, some kind of matching may be required in order to know, in the model, what covariate value is in effect at the time a given response value is measured.

Thus, an intra-unit covariate may require manipulation before storage. It requires one value per response value, in contrast to an inter-unit covariate that needs only one value per individual.

3 Data manipulation

Following upon this typology of variables, we can ask a number of questions:

- How should we read in such data?
- How should we store them?
- What should we do with missing values?

I shall now describe my attempted solution to some of these problems.

3.1 Recording and reading

It should be clear that each type of variable should be recorded in a separate file (maximum 3 files). Such data can create problems as much of the recording technology (spreadsheets) conforms to the rectangular format. Foolproof ways of making the links among values in the different files must be available, as in database technology.

In simple cases, the data can first be read into the statistical software in rectangular form (for example, the dataframe in R or S). If the data are not in such balanced rectangular form, they will generally have to be read as a list (`read.list` or `read.surv`, respectively, for the two examples given above) with one element (vector or matrix) per individual. In either case, they will then have to be converted into the data objects to be described next.

3.2 Storage

In an object-oriented language such as R or S, data are stored in *objects*. In those languages, the principal data object is the rectangular dataframe. From the above arguments and examples, this is not always appropriate for modern data handling.

The one major innovation of the dataframe structure was the ability to store quantitative and qualitative (factor) variables together. However, this is accompanied by subsequent important inefficiencies as the factor variables do have to be transformed into the appropriate set of indicator variables before a model can be fitted. This contrasts with a program such as GLIM where such a matrix containing indicator variables is never actually constructed in the model-fitting process. Such a procedure, although very desirable, would require major modifications (such as bit-coding) to R or S and hence has not been pursued here.

Objects contain *slots* in which various items of different types can be stored and have *methods* by which they can be accessed without the user knowing the internal structure. The objects have *classes* so that the language can recognize which methods are appropriate for which objects. Here, in the implementation in R, the objects will simply be constructed internally as lists (as is a dataframe), permitting storage of varying types of information. The methods to access them will be functions specific to the object.

However, lists cannot easily be directly transferred to and efficiently accessed in a lower level language such as C or Fortran where the more complex model construction needs to be done. Hence, for modelling efficiency, data for all individuals, in an object's slots, will stored together as vectors or matrices so that these can be directly accessed in the lower level language.

Another major inefficiency of Lisp-like languages such as R and S is that copies of objects are generally made when they are passed between functions. In contrast to Fortran or C, objects cannot be accessed through pointers. In R, this problem can be minimized by using R's scoping rules and function closure. However, this implies that many of the specific procedures developed here for R will not work in S.

In the R implementation, three classes of objects, corresponding to types of variables, are available, called `response`, `tccov`, and `tvcov`.

The `response` class contains all of the relevant supplied information discussed above as separate vectors; for a given problem, irrelevant slots are `NULL`. Because of its importance in repeated measurements for locating the appropriate observations, the first level of nesting, indicating the number of observations per individual, is kept as a vector in a separate slot. The unit of measurement and Jacobian are combined in one slot: the unit of measurement is the precision of the instrument used (when different from unity) whereas the Jacobian of the transformation is a set of numerical values. For example, if the response is stored in its slot as say $\log(y)$, then the Jacobian is stored as $1/y$ (times the unit of measurement).

At present, in the R implementation, if factor variables are present among either the intra-unit or inter-unit covariates, the user may choose to store them in the corresponding slot in the object as a dataframe instead of as an ordinary matrix. Model-fitting functions need to know how to handle this. This approach leads to serious inefficiencies as a copy of all of the data must be made when the indicator variables are constructed in setting up a model; in contrast, when the covariates are stored as a matrix, with the indicator variables already calculated for the factor variables, they can be directly read in the object without making a copy. The tradeoff is between ease of referencing variables by name (indicator variables must all be specified by name) and speed of model fitting.

When the observation times for the response and a time-varying covariate differ, the most recent value of the latter can be brought forward to the response time (using the function, `gettvc`). However, care must be taken with ties in the times when the two are recorded. If the covariate and the response are measured at the same time, does the effect begin instantaneously? If the covariate is, say blood pressure, the current value should be used, but if it is the new level of dose of medication, the change will not have had time to take effect, and the previous dose level should be used.

The special covariates, times and nesting indicators contained in the `response` object, can be accessed in model formulæ by keywords: `times`, `individuals` (for the first level of nesting), and `nesting` (for clusters within individuals, for example in certain cross-over trials).

The handling of missing values is a particularly thorny issue. The missing value process will rarely be independent of the process of interest. The only appropriate procedure would seem to be to have a separate slot containing information as to why each particular value is missing so that a model for missingness could be constructed. This would be required in all three classes of objects. Such a structure has not yet been implemented, but such information could, at present, in many cases, simply be stored as extra covariates.

Finally, all of the information from a set of one to three objects of classes `response`, `tccov`, and `tvcov` must be combined to produce an object of the new class called `repeated` (using `rmna`). In this way a given combination of variables in a model is combined, with missing values (NAs) removed. This object provides all of the information that will be required to fit some set of models of interest that will be directly comparable because they are based on the same set of data (for example, with the same missing values removed).

3.3 Methods

Some of the functions and methods required for this approach have already been discussed above. It is now time to look at the required methods in some more detail.

The first basic set of procedures must be able to transform the matrices and/or lists read into the software to create the required objects just described. In my R implementation, these methods are `restovec` to create `response` objects, `tcctomat` to create `tccov` objects, and `tvctomat` for `tvcov` objects. (The names arise for historical reasons, indicating the original underlying data forms contained in the objects.) These can generally automatically transform vectors, matrices, or lists of data to the appropriate object. For example, if a list of matrices (one for each individual, containing responses as the first column and possibly times as the second column) is supplied to `restovec`, it can automatically detect which other columns contain binomial denominators, censoring, nesting, and/or units of measurement.

Once the objects have been created, methods are available to print summary information for each class, not the whole data array, and to plot longitudinal responses and time-varying covariates. The latter allows

- choice of subsets.
- individual points or profiles.
- if nesting, times starting over at zero in each cluster (for example, in a cross-over design).

Methods must also be available to find all information about each individual for a given model.

- As described above, time-varying covariates may need to be carried forward to response times (`gettvc`).
- Interactions among time-varying covariates or with time-constant covariates may be required (`tvctomat`).
- The user may want to transform the response, the times, or certain covariates (`transform`). If the response is transformed, the Jacobian is also automatically updated.
- If the model-fitting procedure does not perform the task itself in constructing the likelihood, time-constant covariates must be matched to individual responses (`covind`).

As discussed above, missing values can only be handled after all information has been joined for a given model. Methods must be available so that, when applicable, this can be coordinated for each given combination of variables (`rmna`).

The only method currently available for handling missing values in my implementation is the elimination of these recordings (`rmna`), with the accompanying (generally incorrect) assumption of randomness. This can only be done for a given combination of all three types of variables, as otherwise the individual values could no longer be matched up. It has the consequence that the number of recorded observations can change with the covariates present in the model and that entire individuals can disappear when the model is changed, for example if an inter-individual covariate value is missing and this covariate is added to the model. However, it would be technically, if not conceptually, easy to develop other methods for handling missing values to be used in place of the `rmna` method.

4 Formulæ

In the mid 1970s, statistics was at the forefront of computations using electronic computers with the introduction of the GLIM interactive system for generalized linear models. Later, S (Becker and Chambers, 1984) extended the same basic paradigm to a wider class of statistical operations, the major innovation for modelling being to allow the user to extend the language, something that was rather difficult with GLIM macros. This lead was, however, rather quickly lost as more powerful packages were developed in other areas, such as Matlab for linear algebra and Maple and Mathematica for symbolic algebraic manipulation. For example, the latter packages contain powerful facilities for distinguishing among known and unknown variables, for translating functions directly into C or Fortran, and for exporting formulæ into $\TeX$. These, and other, useful possibilities that would be invaluable aids for specifying models, are not available in standard packages designed specifically for statisticians.

4.1 Models

4.1.1 Calculating the likelihood

Once data are available in an appropriate form, as described above, and some model has been chosen, the major role of the statistical software is to fit the model by likelihood methods and then to provide any required information about the results. Various criteria can be specified for such a process. Among others, these include

1. correct model formulation;
2. speed: fitting in real time;
3. ease of user specification;
4. default information displayed should not be erroneous or misleading in any context.

Readers will have varying orders of preference for these criteria and will certainly add others.

4.1.2 Specifying the model

Any statistical model generally has two basically distinct parts, the probability distribution and the regression model(s). These are reflected in the data objects described above.

GLIM clearly separated specification of the distribution (`$yvariate` and `$error`) from the linear structure in the model (`$fit`) using the Wilkinson and Rogers (1973) notation. Between these is the link function (`$link`). S (Chambers and Hastie, 1992) obscured this clarity in model construction by combining the first and third GLIM instructions as, in a simple case, `y~x1+x2`. This structure appears to imply that Y is distributed with mean depending linearly on x_1 and x_2. However, this can only be true when the link function is the identity and hence is generally misleading. Both GLIM and S have the additional defect of maintaining the user in a mentality whereby only the mean parameter can possibly depend on covariates.

The probability distribution is generally chosen by the user from a list of possibilities. In the future, one may expect that software will be able to optimize over some set of prespecified model functions in the same way as over a set of parameter values; from a likelihood point of view, the two are logically equivalent. Stepwise and all subsets regression illustrate the misuse of such procedures. What is required is some penalty for the number of models tried, just as the AIC penalizes for the number of parameters in a model.

Regression models describe the ways in which the various parameters (location, dispersion, shape) of this distribution depend on covariates. For linear (parts of) regression models for the mean, the standard way is now by a Wilkinson and Rogers formula set up by the user.

4.1.3 Probability distribution

For the probability distribution, the first criterion is the need for a wide choice. The five generalized linear models usually provided (even the Weibull distribution is excluded!) by statistical software are entirely insufficient. This current restriction is essentially a technically one: the linear parameters of all generalized linear models can easily be estimated by the iterated weighted least squares algorithm.

For repeated measurements, the stochastic dependence relationships among the responses of an individual must be specified. These include the longitudinal dependencies and clustering effects. Except in special cases, the former will require recursive updating. The latter are generally handled by random effects, but this requires either some form of efficient multidimensional integration or recursive updating. For speed, recursive likelihoods (such as Kalman filtering) must be calculated in a lower level language such as C, dynamically loaded into R or S. Only vectorized operations are sufficiently fast for likelihood construction directly in R or S and such recursion cannot be vectorized.

4.1.4 Covariate dependence

Nonlinear function construction In a language such as R or S, one natural way to specify complex regression models, not handled by the Wilkinson and Rogers notation, is to use the builtin function construction abilities of these languages. Let us first look at this approach.

As an example, consider an open first-order one-compartment model widely used in pharmacokinetics. The location parameter over time varies as

$$\mu_t = \frac{Vk_a}{k_a - k_e}\left(\mathrm{e}^{-k_e t} - \mathrm{e}^{-k_a t}\right)$$

where t is time, V is volume, k_a is the absorption rate, and k_e is the elimination rate. The latter three are unknown parameters. In R or S, this can be specified as a function of the parameter vector:

```
mu <- function(p){
    p[1]*p[2]/(p[2]-p[3])*(exp(-p[3]*times)-exp(-p[2]*times))}
```

Note that the function does not have, as argument, the times, so that they are not copied when the function is evaluated, but must be found somewhere in the environment, preferably in a specified data object.

In addition, the dispersion parameter depends on time through the location parameter, often assumed to be

$$\sigma^2 = \mu_t^\delta$$

The corresponding function for the log dispersion might be

```
disp <- function(p, mu) p[1]*log(mu)
```

Another possibility is that the regression functions for two parameters of the probability distribution may have parameters in common without one being a strict function of the other, as here.

The parameters in such a regression model, k_a, k_e, V, may also depend, in various ways, on other covariates. I next consider this.

Linear (in parameters) part As already mentioned, the linear part of a model is now generally specified by the Wilkinson and Rogers notation for formulæ. Any extensions should retain this as a subset. On the other hand, link functions are only useful for a (transformed) parameter depending on covariates through a strictly linear model.

To pursue our example, if the dependence of a parameter on the covariates contains a strictly linear part, this might be specified as

```
mu <- function(p, linear){
  tmp <- exp(linear)
  p[4]*tmp/(tmp-p[5])*(exp(-p[5]*t)-exp(-tmp*t))}
```

Here, the absorption parameter depends on the covariates through a linear (in the parameters) part, after application of a log link to ensure that its value is always positive. Then, some model-fitting function might have the following general form:

```
    modelfn(..., mu=mu, linear=~height+gfr, ...)
```

Here, the expression following the tilde is in standard Wilkinson and Rogers notation and may refer to factor variables. It never contains anything before the tilde and, hence, can be used to specify how any parameter of a probability distribution depends on covariates.

This approach has been implemented in my suite of nonlinear regression and repeated measurements R libraries that I shall describe briefly below. The function, `fnenvir`, can modify an R function such as the above so that it reads the covariates from the data objects described above.

4.1.5 General nonlinear specification

The above approach to specifying nonlinear models, through R or S functions, is powerful and useful, but certainly not always as intuitively clear and user-friendly as it might be. However, for very complex models where a series of functions and subfunctions may be required, it may be the only feasible and efficient method at present possible.

In a linear regression model, the positions of the parameter coefficients in the formula do not need to appear explicitly. They can be implicitly assumed, as in the Wilkinson and Rogers notation. This is no longer possible in any extension to nonlinear regression models. To implement nonlinear specification, the ability to distinguish variables and parameters (existing vectors and unknowns) in formulæ (like Maple, Mathematica) would be a step forward. Then, both variables and parameters could retain their individual names. Thus, for the above example, the user should be able to specify, as an argument to a model-fitting function,

```
    mu = ~volume*absorption/(absorption-elimination)*
      (exp(-elimination*times)-exp(-absorption*times))
```

and the model-fitting function can detect automatically which are known covariates (preferably stored in the data objects described above) and which are unknown parameters.

Here, the tilde begins the formula to indicate that it is a formula; again, it is not preceded by the response variable. Thus, it does not have the restrictive S signification that the response is 'distributed as' the model in the formula. Thus, again, such formulæ can be used to describe how any parameter in a probability distribution depends on covariates.

Note that such a formulation may be inefficient in complex situations, as with the linear part for the dependence of the absorption parameter on covariates above. Here, this would have to be given twice whereas it was only given once above in the R function. Matters quickly become worse in still more complex cases so that the direct use of R functions is then still necessary.

Various criteria are required for handling such expressions in model-fitting functions:

- detecting existing covariate vectors;
- ignoring conflicts of unknown parameters with existing (non-variable) object names, such as

functions;

- transforming the formula into a function of *one* vector of unknown parameters;
- substituting the resulting function into a probability distribution function to construct a likelihood *function* that can be rapidly evaluated by a nonlinear optimizer;
- evaluating it either in the appropriate environment or with respect to the data object(s) supplied to the model-fitting function, without copying the data.

All of the above criteria have been fulfilled in my R implementation: the function, `finterp`, constructs the appropriate R function from such a model formulation with known covariates and unknown parameters, using function closure to retain the environment in which it was defined. As for Wilkinson and Rogers formulæ, which `finterp` also recognizes and can also transform into R functions, the formulæ begin with a tilde so that they have class, `formula` and are `language` objects.

Although its primary use is to define regression models, this formulation can also be used to construct a complete likelihood function as the following example for Poisson nonlinear regression shows.

```
# the regression function
regfn <- finterp(~a+exp(b0+b1*x1+b2*x2))
# the terms of the negative log likelihood function
poisfn <- finterp(~-y*theta+exp(theta)+lgamma(y+1), vector=F)
# the null likelihood
poislikefn <- function(p) sum(poisfn(theta=p))
# the regression likelihood
poisreglikefn <- function(p) sum(poisfn(theta=regfn(p)))
```

The latter two assignments yield R functions that can be fed directly into the nonlinear optimizer. Here, the variables, `y`, `x1`, and `x2`, are searched for in the global environment and `a`, `b0`, `b1`, and `b2` are recognized to be unknown parameters (if they do not exist), being collected together to form one vector argument to the function, `regfn`. If the appropriate data object were specified as the environment in an additional argument to `finterp`, the variables would, instead, be sought in that object (and only there).

5 Discussion

Both the procedures for creating data objects and for handling model formulæ have been implemented in my R library called `rmutil`. They are used in a wide variety of model-fitting functions in my libraries, `gnlm` for generalized nonlinear regression models, `growth` for multivariate normal and elliptical distribution models for repeated measurements, `repeated` for non-normal repeated measurements models, and `event` for event histories. Many of the model-fitting functions in these libraries provide the choice among approximately 25 different probability distributions.

The latest source code for all of these libraries is available at

`www.luc.ac.be/~jlindsey/rcode.html`

All of the examples of the analysis of repeated measurements data in Lindsey (1999) were analyzed using this system. The data and R code are available at

`www.luc.ac.be/~jlindsey/books.html`

Some of the weaknesses of the implementations have been mentioned above. As well, certain data types definitely cannot be handled by the proposed data objects, such as overlapping clusters. Others not yet available could easily be implemented, for example, multivariate responses by storing them in the slot as a matrix instead of a vector. Still others are inefficiently implemented, such as spatial coordinates which might better be stored as some sort of tree system rather than simply as a pair of vectors.

Factor variables cannot be handled in the nonlinear formulæ because they refer to a vector of parameters. It would also be nice to be able to nest the specification of models (a formula within a formula), for example, a linear part specified by Wilkinson and Rogers notation within a nonlinear regression.

Acknowledgments I thank Robert Gentleman and Ross Ihaka and the core group for developing the R software in which the functions described in this paper were written.

References

[1] Becker, R.A. and Chambers, J.M. (1984) *S: An Interactive Environment for Data Analysis and Graphics.* Monterey: Wadsworth.

[2] Chambers, J.M. and Hastie, T.J. (1992) *Statistical Models in S.* Monterey: Wadsworth.

[3] Ihaka, R. and Gentleman, R. (1996) R: a language for data analysis and graphics. *Journal of Computational Graphics and Statistics*, **5**, 299–314.

[4] Lindsey, J.K. (1999, 2nd ed.) *Models for Repeated Measurements.* Oxford: Oxford University Press.

[5] Wilkinson, G.N. and Rogers, C.E. (1973) Symbolic description of factorial models for analysis of variance. *Applied Statistics* **22**, 392–399.