XLISP-Stat tools for building Generalised Estimating Equation models

Thomas Lumley
Department of Biostatistics
University of Washington
Box 357232
Seattle WA 98195–7232

June 12, 1996

Abstract

This paper describes a set of LispStat tools for building Generalised Estimating Equation models to analyse longitudinal or clustered measurements. The user interface is based on the built-in regression and generalised linear model prototypes, with the addition of object-based error functions, correlation structures and model formula tools. Residual and deletion diagnostic plots are available on the cluster and observation level and use the dynamic graphics capability of LispStat.

1 Introduction

Generalised Estimating Equations models, proposed by Liang and Zeger in 1986, are probably the simplest method for analysing data collected in groups where observations within a group may be correlated but observations in separate groups are independent. A complete description of the method is given in their two 1986 papers. The basic principle of the method is a generalisation of the fact that weighted least squares analyses give unbiased parameter estimates no matter what weights are used. Generalised linear models, such as logistic regression, have similar robustness properties, giving asymptotically correct parameter estimates even when the data are correlated. This means that it is possible to estimate regression parameters using any convenient or plausible assumptions about the true correlation between observations and get the right answer even when the assumptions are not correct. It is only necessary to use a "model-robust" or "agnostic" estimate of the standard errors. It would be unreasonable to expect this freedom of choice to be without cost and it turns out that there is a moderate gain in efficiency resulting from choosing a working correlation structure close to the true one.

Useful references include the two original papers (Zeger & Liang 1986, Liang & Zeger 1986) and two recent books: Diggle, Liang & Zeger (1994) and Fahrmeir & Tutz (1994). As far as I know the most elementary treatment anywhere in the literature is still Zeger & Liang (1986).

When the observations are all independent (ie only one observation per group) the GEE method reduces to a GLM with the Royall/Huber/White "agnostic" ("model-free", "model-robust", "heteroscedascity-free", "infinitesimal jackknife"...) sandwich estimate of standard errors. This provides the easiest way of constructing these extremely useful estimators in XLISP-Stat. In this situation the parameter estimates are the same as the usual GLM ones and the naive standard errors are the usual GLM standard errors. This makes it possible to use the `gee-proto` to construct generalised linear models. This is only worthwhile when you want to use a non-standard link function: the `gee-model` wrapper function handles arbitrary link and variance functions, unlike the wrapper functions for the `glim-proto`

2 Components of a GEE model

This similarity to the well-known generalised linear models makes GEEs very easy to use and to program. The XLISP-Stat code accompanying this document adapts the ideas of Tierney's (1989) regression model and generalised linear model prototypes and defines a general `gee-proto` prototype. To create `gee-model` one supplies the data and a choice of link function, variance function and correlation structure. The first two components are familiar from generalised linear models; the third is specific to GEEs. The independence correlation structure or *independence working model* is the simplest one, corresponding to estimating the regression parameters as if the data were independent. The exchangeable working model estimates a single correlation parameter and so behaves as if all the within-group correlations are identical. This is similar to a Normal-theory model with a single random effect for the intercept and is a good way to analyse cluster-sampled or cluster-randomised studies.

The saturated working model is only appropriate when the number of observations in each group is the same and there is a well-defined order within each group. It estimates the whole within-group correlation matrix and so allows the correlation between, say, the first and second observations to be different from that between any other pair. In large data sets this is the most efficient method and for Normal data it is equivalent to maximum likelihood (if the right correlation estimator is used, a point discussed in more detail below).

Other possible working correlation structures come from analogies to time series include 1-dependence, when the correlation is assumed to be zero except for adjacent observations, the more general m-dependence when correlations are estimated out to a separation of m observations, and continuous and discrete-time autoregressive structures.

3 How to use it

There is a wrapper function `gee-model` which takes the following arguments

`:x`	sequence, list of sequences matrix, or model-formula	predictor variables
`:y`	sequence	response variable
`:g`	sequence	grouping variable
`:times`	sequence	for longitudinal data: observation number or time
`:link`	link object	link function
`:error`	error (variance) object	variance function
`:correlation`	correlation object	working correlation
`:offset`	sequence	offset (default 0)
`:prior-weights`	sequence	prior weights (default 1s)
`:intercept`	logical	include an intercept? (default `t`)
`:verbose`	logical	be chatty? (default `t`)
`:print`	logical	display the results? (default `t`)
`:allow-missing`	logical	allow missing observations (only affects some correlations)
`:init-beta`	sequence	initial guess for β
`:count-limit`	integer	maximum number of iterations (default 30)
`:predictor-names`	list of strings	names

You must specify the first three arguments and at least one of `:link` and `:error`. If only `:error` is specified the canonical link is used, if only `:link` is specified then the appropriate error function is chosen. If `:correlation` is not specified then the independence working model is used. If `:verbose` is true then messages are printed out about these choices.

The independence and exchangeable correlation structures ignore the observation `:times` argument, for other structures it is always required (in contrast to other packages). This is a feature, not a bug. The GEE functions written for other general-purpose packages mostly assume the data are sorted so that each group is contiguous and in the right order. In my experience data do not naturally occur that way: if the times column is in the data set it is easy to use it, if it isn't it is hard to be confident that you have the ordering correct (and virtually impossible with missing observations). The performance implications of this are discussed below.

Note that for binomial data $y = r/n$ you should use the observed proportion not the observed count as the response and then specify the denominators using `:prior-weights`. Correlated binomial data (other than binary) are comparatively rare; the individual binary observations are usually available.

The default pairs of variance and link functions are:

binomial-error	logit-link
	cloglog-link
	probit-link
poisson-error	log-link
normal-error	identity-link
gaussian-error	
gamma-error	inverse-link

where `gaussian-error` is identical to `normal-error`. In addition there are power link and variance functions given by `(power-link k)` for $\mu = \eta^k$ (with `inverse-link` and `sqrt-link` as special cases) and `(power-error k)` for $V(\mu) = \mu^k$. These do not have default counterparts. The use of power link and variance functions in modelling data with non-Normal, skew error distribution was discussed by Nelder (1994).

Any link function can be used with any variance function but the results may be silly. The link functions are taken from the `glim-proto` system but have had extra methods `:valideta` and `:validmu` added to allow domain checking.

The available working correlation structures are currently `independence-corr`, `exchangeable-corr`, `m-dependence` and `stat-m-dependence`, `saturated-ml-corr` and
`saturated-corr`, `fixed-corr` and `AR-1-corr`. The difference between the two saturated correlation structures, both of which estimate an unstructured correlation matrix, is discussed in more detail below. The parameter `:allow-missing` controls the assumptions made when the observation times are not the same for every group. Under the exchangeable and independence models it has no effect; under the other models unbalanced data are not allowed when `:allow-missing` is `nil`. When `:allow-missing` is `t` unbalanced data are allowed and the correlation parameters are estimated from the available data. This is valid if the data are missing completely at random but not otherwise (unless the working correlation model is true).

The basic m-dependence structure is non-stationary m-dependence. It takes an argument (m) and so two-dependence, for example, would be specified as `(m-dependence 2)`. To specify stationary 2-dependence use `(stat-m-dependence 2)` or `(m-dependence 2 :stationary t)`. A known correlation matrix can be specified using the fixed correlation structure: `(fixed-corr R)` uses R as the correlation matrix. This is not very useful except as a quick and dirty way of using a complicated correlation structures (eg nested clustering). It is included for completeness and because it was easy. Note that R need not be a correlation matrix, it just has to be positive definite. This could be useful if the scale parameter varied across times, for example. For the fixed correlation structure the scale parameter estimate is fixed at 1 for simplicity. This has no effect on the parameter estimates or the robust standard errors but does affect the naive standard errors.

There is no need for the observation times to be consecutive integers. In fact they need not be integers at all, or even numbers. The requirements are that `equalp` is a valid test for equality and that `rank` will rank them. This means in practice that they must be all numbers or all strings and that if they are string

the correct order is alphabetic. Some correlation structures in future may use the numerical values of the times in which case they will have to be numbers. Group identifiers must be strings or numbers too; they are also compared with `equalp`.

Some examples, using a data set distributed with the original SAS GEE macro by Karim and Zeger. The data give the number of hospital visits of 73 children over four time periods together with the age and sex of the child and the smoking status of the mother.

```
> (def gee0a (gee-model :x (list sex smoke age time2 time3 time4)
:y num-visits :g id :times times :link log-link :error poisson-error
 :correlation exchangeable-corr :predictor-names (list "sex" "smoker"
"age" "2nd vs 1st" "3rd vs 1st" "4th vs 1st")))
Iteration 1: quasideviance = 517.114
Iteration 2: quasideviance = 478.436
Iteration 3: quasideviance = 475.935
Iteration 4: quasideviance = 475.970
Iteration 5: quasideviance = 475.982
Iteration 6: quasideviance = 475.982

GEE Estimates:

                         Coefficient     Std Error           Naive Std Error
Constant                  0.245980      (0.383535)          (0.363382)
sex                      -0.194216      (0.208003)          (0.222975)
smoker                    0.168846      (0.244118)          (0.242226)
age                       2.963862E-3  (6.653705E-3)       (6.265295E-3)
2nd vs 1st               -0.435318      (0.195649)          (0.188774)
3rd vs 1st               -0.307485      (0.202848)          (0.181464)
4th vs 1st                -1.12847      (0.221787)          (0.243342)

Scale Estimate:            1.83171
Independence model deviance:   475.982
Number of cases:               292
Link:                     #<Glim Link Object: LOG-LINK>
Variance function:        Poisson error
Exchangeable Working Model:  correlation =  0.2257
GEE0A
>
```

Most of this is fairly clear. The "Independence model deviance" is the result of applying the deviance function for the independence working model to the fitted model. These will at least give some idea of the relative differences between models. The scale factor is the variance of the Pearson residuals, the reciprocal of the parameter ϕ in Zeger & Liang (1986) and the analogue of the residual mean square in linear regression. The "naive standard error" is

the standard error that would be estimated from a parametric analysis if the working correlation structure were true.

Another example:

```
> (def gee2a (gee-model :x (list sex smoke age time2 time3 time4)
 :y num-visits :g id :times times :link log-link :error poisson-error
 :correlation (stat-m-dependence 2) :predictor-names (list "sex"
"smoker" "age" "2nd vs 1st" "3rd vs 1st" "4th vs 1st")))
Iteration 1: quasideviance = 517.114
Iteration 2: quasideviance = 478.517
Iteration 3: quasideviance = 475.988
Iteration 4: quasideviance = 475.944
Iteration 5: quasideviance = 475.941
Iteration 6: quasideviance = 475.941

GEE Estimates:

                        Coefficient     Std Error           Naive Std Error
Constant                 0.246413      (0.425101)          (0.349471)
sex                     -0.176885      (0.218639)          (0.213840)
smoker                   0.153786      (0.261751)          (0.230775)
age                      2.934218E-3  (7.046725E-3)       (5.989854E-3)
2nd vs 1st              -0.435318      (0.195649)          (0.190156)
3rd vs 1st              -0.307485      (0.202848)          (0.185687)
4th vs 1st               -1.12847      (0.221787)          (0.270775)

Scale Estimate:            1.82809
Independence model deviance:   475.941
Number of cases:               292
Link:                     #<Glim Link Object: LOG-LINK>
Variance function:        Poisson error
Stationary  2 - dependence Working Model:
 correlation matrix =
#2a(
    (  1.0      0.21      0.19      0.00    )
    ( 0.21       1.0      0.21      0.19    )
    ( 0.19      0.21       1.0      0.21    )
    ( 0.00      0.19      0.21       1.0    )
   )
GEE2A
>
```

For the stationary 2-dependence working model we see that the estimated correlations are printed. Adding `:verbose nil` to the arguments will suppress everything before the line `GEE Estimates`. Adding `:print nil` will suppress

everything from that point on (only really useful if you are doing further programming using the resulting object).

The function returns a `gee-proto` object which responds to a lot of messages, including many of the messages implemented for generalised linear models. Some description of most of the methods is available using the `:help` method (eg `(send mygeeobject :help :scale)` describes the `:scale` method. To find all the methods which have help omit the topic specifier: `(send gee-proto :help)`.

There is also a method for producing confidence intervals for the coefficients, transformed to any appropriate scale. The message `:conf-interval` gives the estimates and a 95% confidence interval. There are three optional keyword arguments: `:coverage` specifies the desired coverage, `:names` is `nil` to prevent predictor names being returned and `:transform` specifies a function to apply to the confidence interval. To produce 90% confidence intervals for the odds ratio in a logistic regression model you would use the message `:conf-interval :coverage 0.9 :transform #'exp`.

There are further examples in the test file `geetest.lsp`. This includes most of the test examples for an S-PLUS `gee()` function by Carey and Macdermott that is available from `statlib` (`ftp:lib.stat.cmu.edu`). Good agreement with the S-PLUS results is obtained. The code has also had limited testing against the statistical package SPIDA (Statistical Laboratory, Macquarie University, Sydney, Australia) which has a built-in GEE procedure.

4 Model formulae

It is possible to use a model formula object to include factor (class) and interaction terms more easily in a model. The support is still a bit primitive: it's less simple than either SAS or S-PLUS and less complete, allowing interactions but not nesting. The function `as-formula` creates formula objects which contain a design matrix `:design-matrix`, a list of column labels `:name-list`, a list of block (eg factor or interaction) names `:block-names` and a list of lists of column numbers specifying each block `:block-indices`. This should have been an extension of Tierney's design matrix tools written for `glim-proto` but I didn't realise they were there so I wrote new functions for creating indicators and interactions. The form of a call to `as-formula` is

```
(as-formula '((factor a) (factor b) (term x) (interaction (list a b))
 (interaction (list a x) :is-factor (list t nil))))
```

where `A` and `B` are factor variables and `X` is a metric (continuous) variable. This model would be written `~A+B+X+A:B+A:X` in S-PLUS if `A` and `B` had already been defined as factors. The keyword argument `:is-factor` to `interaction` indicates whether of the terms in the interaction should be treated as factors. It is optional; the default is to treat them all as factors. The reason for this default is partly that it is more common and partly that making a mistake is more likely to be obvious. The argument to `(as-formula)` must be quoted, as

in the example. This appears to be unavoidable as otherwise the arguments will get evaluated. The Right Thing would be to write a parser that could accept strings as arguments.

The `gee-model` command and `gee-proto` object have some built-in recognition of formula objects, such as the ability to use a formula object as the `:x` argument to `gee-model`. The file `modelformula.lsp` which defines the objects also contains methods for `regression-model-proto` and some more primitive functions which may be useful for creating and manipulating design matrices. In particular the functions `interaction`, `factor` and `term` can be used on their own to create contrast matrices and variable names.

The main difference caused by using a model formula argument to `gee-model` is that the default output changes to report the Wald chisquare and p-value for each block. This is produced by the message `:display-with-formula :block-only nil`. To get only the test statistics, use the message `:display-with-formula :block-only t`. To get the old display use `:display`. The default value of the `block-only` keyword is controlled by a global variable `*gee-display-block-only*` which is initially `nil`.

The first of the two examples given above can be replaced by the following code

```
> (def modelx (as-formula '((factor gender) (factor smoker) (term age)
(factor times))))
>  (def gee1 (gee-model :x modelx :y num-visits :g id :times times
:link log-link :error poisson-error :correlation exchangeable-corr))
Iteration 1: quasideviance = 517.114
Iteration 2: quasideviance = 478.436
Iteration 3: quasideviance = 475.935
Iteration 4: quasideviance = 475.970
Iteration 5: quasideviance = 475.982
Iteration 6: quasideviance = 475.982

GEE Estimates:

Block               Wald Chisq     p-value
Intercept             0.41133      0.5213
      Variable            Estimate       Std.Err.       p-value
     Intercept             0.24598      (0.383535)     0.5213
GENDER                0.87183      0.3504
      Variable            Estimate       Std.Err.       p-value
     (GENDER M)           -0.19422      (0.208003)     0.3504
SMOKER                0.47839      0.4892
      Variable            Estimate       Std.Err.       p-value
     (SMOKER Y)            0.16885      (0.244118)     0.4892
AGE                   0.19842      0.6560
      Variable            Estimate       Std.Err.       p-value
     AGE                   2.96386E-3  (6.653705E-3)  0.6560
```

```
TIMES                  26.659      0.0000
      Variable            Estimate       Std.Err.       p-value
     (TIMES 2)            -0.43532      (0.195649)     0.0261
     (TIMES 3)            -0.30748      (0.202848)     0.1296
     (TIMES 4)             -1.1285      (0.221787)     0.0000

Scale Estimate:            1.83171
Independence model deviance:   475.982
Number of cases:               292
Link:                     #<Glim Link Object: LOG-LINK>
Variance function:        Poisson error
Exchangeable Working Model:  correlation =  0.2257
GEE1
>
```

where `GENDER` and `SMOKER` are character versions of the variables `SEX` and `SMOKE` used in the earlier model. The predictor names are automatically generated from the variable names and values. This is particularly useful when the variable names and values are easily interpretable as in this example.

5 Diagnostics

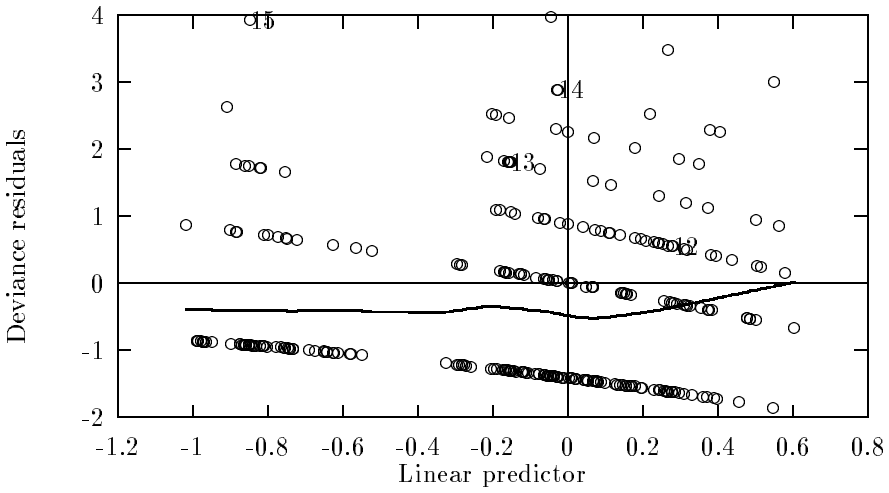
Deletion diagnostics are available as well as a variety of flavours of residual. Current graphical methods are

`:plot-raw-residuals`	$y - \hat{y}$
`:plot-pearson-residuals`	same as the independence case
`:plot-deviance-residuals`	same as the independence case
`:plot-standardised-residuals`	Pearson residuals standardised by working correlation
`:plot-leverage`	Leverage statistics
`:plot-dbeta`	$\Delta\beta$s
`:plot-cooks-distance`	Cook's distance (influence)

The standardised residuals are the Pearson residuals multiplied by the Cholesky square-root of the inverse of the correlation matrix. They are approximately independent with equal variance if the working model is roughly true.

The last three methods are generalisations of the leverage (diagonal of hat matrix), DBETA (change in coefficient when point is deleted) and Cook's distance (influence) statistics for linear and generalised linear models. The formulae are from Preisser & Qaqish (1996 preprint) and are natural generalisations of the independent data versions (though the derivations and calculations are rather more complicated). The methods take a keyword argument `:unit` of 0 for individual observation deletion and 1 for group deletion. Constants `+group+` and `+cluster+`, and `+not-group+` and `+observation+` have been defined to make these easier to use (eg `:unit +group+`). The default is diagnostics for individual observations.

Figure 1: Deviance residuals against linear predictor with group highlighting. Plot created by `(send gee1 :plot-deviance-residuals)` and selecting point 15 with the mouse.



The residual plots and `plot-dbeta` take a keyword argument `:variable`. The argument for `:variable` is a list of indices specifying which delta-betas to plot or which variables to plot the residuals against. For `:plot-dbetas` variable 0 is the intercept, for `:plot-foo-residuals` it is the linear predictor. If `variable` is not specified the delta-betas plot uses all variables; the residual plots just use the linear predictor. In addition, `:plot-foo-residuals` has a `:smooth` keyword which indicates whether a lowess smoother should be added to the plot. The default is `t` for both `show-labels` and `smooth`. Any other arguments given to these functions will be passed to a call to `plot-points`. An example is given in Figure 1, based on the example given above for the model formula system.

The plots produced by these methods are linked, so that selections in one plot show up in the others. The residual plots also have an additional mouse mode "group highlighting" where selecting a point also selects all the other points in the same group. The linking is different from the usual LispStat mechanism in that all group-level plots are linked and all observation-level plots are linked but plots at different levels are not linked to each other. The gee object contains lists of observation-level and group-level plots that are separate from the linking list maintained by LispStat. To link another plot (eg a scatterplot of covariates) use the `:bind-plot-to-gee` message

```
(send mygee :bind-plot-to-gee a-plot +group+)
```

links `a-plot` to the group-level plots and similarly

```
(send mygee :bind-plot-to-gee a-plot +observation+)
```

links it to the observation-level diagnostic plots.

The same method names without the `plot-` prefix return the values of the diagnostics. The individual observation diagnostics are returned in a list or matrix in the same order as the data set; the group diagnostics have the group identifier attached and are in no reliable order. The plotting methods call two plotting functions which may be useful for other purposes: `index-plot` and `scatter-smooth`. The former takes a single argument and plots it against the integers; the second behaves like `plot-points` but also adds a lowess smooth to the plot.

6 Notes on Internal Workings

The main computations are carried out in the method `:compute-step-beta`. This does not use the iteratively reweighted least squares method of `glim-proto` which would require rewriting `regression-proto` to handle non-diagonal weight matrices (feasible) and would involve inverting $n \times n$ matrices (undesirable). As observations in different groups are independent the weight matrix is block diagonal and so is mostly easily dealt with in blocks correponding to the groups. This only requires inverting the correlation matrix for each group and the final quasi-information matrix, all of which are small. The algorithm is as described by Liang & Zeger (1986). Variable names in `compute-step-beta` are generally based on the notation in that paper. Some of the correlation estimates are slightly modified from the original paper to agree with other software.

The information required for the deletion diagnostics is computed by the same function as updates the regression parameter estimates. At the moment this information is not computed in the usual fitting process and another iteration is required the first time any deletion diagnostics are calculated. It would be possible to calculate this information while the model is fitted. This would cause extra garbage collection but would save time in plotting diagnostics. To do this, set the global variable `*gee-diagnostics-while-fitting*` to `t`. Variable names in the deletion diagnostic methods are based on the notation of Preisser & Qaqish, and so differ from those in the rest of the code.

As mentioned above, the gee functions do not assume that the data are presorted by group and observation time and instead require that observation times are supplied. This is because missing data handling is practically impossible without having a record of observation times (a recent version of the SAS macro introduces observation times to handle missing cases). It is also safer: other packages will give you the wrong answer with no warning if the data are not sorted by time within group. The way this is now implemented is not very efficient: asymptotically the calculation time is dominated by the time taken to find each group. More efficient code using hash tables would be possible. In practical examples, though, this is not a real issue. For very large data sets it is clearly worth assuming the data are sorted but it is also worth writing special-purpose compiled code.

As the observations within a group may vary in number and be in random order the correlation structures use an association list to match up the `:times` variable with the appropriate indices for selecting a submatrix of the correlation matrix. This produces very simple code for the method that returns the inverse correlation matrix and also simplifies the code for estimating correlations.

Despite various inefficiencies described above the code is fairly fast. A test data set with 72 groups of 5 observations and 5 variables using the autoregressive correlation structure (one of the slowest) took about 5 seconds per iteration on a (shared) Sparcstation 10 or about 6 seconds per iteration on a 75mHz Pentium PC running Linux. This data set had the same number of observations in each group; unbalanced data require an extra matrix inversion for each group that is smaller than the largest one and so would be slightly slower. The speed is only improved about 20% after compiling the code, presumably because it is mostly vectorised and so none of the inefficiency is easy to optimise away.

One feature of GEEs and the generalised linear models from which they arose is the ability to combine arbitrary link and variance functions. Some care is needed to ensure that the numerical estimation doesn't crash when strange combinations of link and variance are chosen. The `:compute-step-beta` method contains a check that the new values of the regression coefficient produce valid values for the linear predictor and the fitted mean. If this is not the case then the step size is halved and the check repeated until the values are valid. A warning message is also printed. The choice of starting values for the regression parameters is also complicated when the link and variance can be chosen arbitrarily (this is a disadvantage of not using iterated reweighted least squares). The program deals with three cases. For all the commonest models $\beta = 0$ gives valid estimates. For models where this is not the case (eg gamma/reciprocal link, power variance/power link) the initial values depend on whether the model contains an intercept. If it does then all the other regression parameters are set to 0 and the intercept is chosen to give valid starting values. If there is no intercept then a valid set of starting values need not exist. The program tries $\beta_i = g(\bar{y})/p\bar{x}_i$ (where $g(\cdot)$ is the link function) and if this doesn't work it gives up and asks for starting values to be specified.

The saturated correlation model estimates all the pairwise correlation parameters separately. There is a unique sensible estimate of the covariance matrix (assuming a balanced design) but not of the correlation matrix. The problem is that the diagonal elements of the covariance matrix are assumed equal under the model but will not be estimated as equal. The method used in the S-PLUS library and attributed to Karim's SAS macro is to estimate the dispersion ϕ by the variance of the Pearson residuals, to estimate the correlations ρ_{ij} by $\rho_{ij} = \mathrm{cov}(r_i, r_j)/\phi$ for $i \neq j$ where r_i are the set of Pearson residuals at time i and to set $\rho_{ii} = 1$. I think this would be the ML estimate of the correlations for Normal data with constant variance. Liang & Zeger (1986) used a similar estimate but didn't set the diagonals to 1. This gives the maximum likelihood estimate of the covariance matrix if the dispersion is not assumed constant over time. A third possibility would be to use the Normal maxmimum likelihood estimate of the correlation matrix. It is not clear which is best, but most of

the programs have settled on the same choice, which makes software testing easier. There is one significant disadvantage of this choice: it is the only one where the "correlation matrix" need not be positive definite. This can happen when the sample size is not large relative to the number of correlations being estimated especially if the true time-specific variances are not all the same. It can only happen for three or more time points as for two time points the arithmetic-geometric mean inequality guarantees that the estimated correlation will be less than the usual estimate. The usual choice (diagonals forced to 1) is correlation structure `saturated-corr`, the covariance matrix divided by the scale parameter (the estimate given by Liang & Zeger and Fahrmeir & Tutz) is `saturated-ml-corr`

Much of the default behaviour is under the control of global variables. These are defined and documented at the end of the source file and control the printing of iteration information, the default working correlation, the default converge tolerance and maximum iteration count and the default display methods.The variable names are all of the form `*gee-...*` They are defined at the end of the file because the default correlation cannot be selected until after the correlation objects have been defined.

7 Acknowledgements

The author is supported by a Howard Hughes Medical Institute Predoctoral Fellowship in the Biological Sciences. Thanks to Anthony Rossini, Charlie Hall and John Preisser for help with diagnostics.

8 References

Diggle, P., Liang, K-Y., Zeger, S.L. (1994) *Analysis of Longitudinal Data* Oxford University Press, Oxford.

Fahrmeir, L., Tutz, G. (1994) *Multivariate Statistical Modelling based on Generalized Linear Models* Springer, New York.

Liang, K-Y., Zeger, S.L., (1986) "Longitudinal analysis using generalized linear models." *Biometrika* 73: 13–22.

Nelder, J.A. (1994) "An alternative view of the splicing data." *Applied Statistics.* 43: 469–476.

Preisser, J.S., Qaqish, B.F., (in press) "Deletion diagnostics for generalized estimating equations".

Tierney, L. (1989) *XLISP-Stat: A Statistical Environment based on the XLISP language.* Technical Report No. 528. University of Minnesota School of Statistics.

Zeger, S.L., Liang K-Y., (1986) "Longitudinal data analysis for discrete and continuous outcomes" *Biometrics* 42: 121–130.