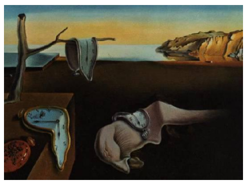

Journal of Statistical Software

August 2005, Volume 14, Issue 8. *http://www.jstatsoft.org/*

An Algorithm for Clustered Data Generalized Additive Modelling with S-PLUS

Lin Yee Hin
Private Medical Practitioner

Vincent Carey
Harvard Medical School

Abstract

We present a set of functions in S-PLUS to implement the clustered data generalized additive marginal modelling (CDGAM) strategy proposed by Berhane and Tibshirani (1998). A variety of working correlation structures are supported, and the regression basis may include components from the family of smoothing splines.

Keywords: generalized estimating equations, clustered data analysis.

1. Introduction – Notations and theoretical background

The CDGAM algorithm for the semi-parametric setting presented in this paper implements fitting methods for a class of generalized additive models for clustered data with observations $(y_{it}, \mathbf{x}_{it})$. Here $t = 1, \cdots, m_i$ indexes observation times within the ith cluster, $i = 1, \cdots, n$; y_{it} is the response with expected value μ_{it} and $\mathbf{x}_{it}$ is a $(p+q) \times 1$ vector of covariates. There are p parameters to be estimated under a standard generalized linear modeling framework, and q smooth functional parameters to be estimated non-parametrically. The marginal mean of the response is related to the parameters and covariates by

$$g(\mu_{it}) = \eta_{it} = \beta_0 + \beta_1 X_{it1} + \cdots + \beta_p X_{itp} + f_1(X_{it(p+1)}) + \cdots + f_q(X_{it(p+q)}) \tag{1}$$

The marginal variance of the response, $\mathsf{VAR}(y_{it})$, depends functionally on the marginal mean through the function $v(\mu)_{it}$. Hence

$$\eta_{it} = \eta_{\text{total},it} = \eta_{\text{parametric},it} + \eta_{\text{nonparametric},it} \tag{2}$$

We refer to the 1 to $(p+1)$ components of (1) as the parametric component and components $(p+2)$ to $(p+q)$ as the nonparametric component of the model.

Estimation proceeds by forming the adjusted dependent variable McCullagh and Nelder (1989)

$$\mathbf{z} = \eta + \mathbf{D}^{-1}(\mathbf{y} - \mu) \tag{3}$$

where $\mathbf{D} = \mathbf{D}_1 \oplus \mathbf{D}_2 \oplus \cdots \oplus \mathbf{D}_n$ and using iterative reweighted least squares with weights $\mathbf{W} = \mathbf{W}_1 \oplus \mathbf{W}_2 \oplus \cdots \oplus \mathbf{W}_n$ and $\mathbf{W}_i = \mathbf{D}_i \mathbf{V}_i^{-1} \mathbf{D}_i$. $\mathbf{D}_i$ is a $m_i \times m_i$ diagonal matrix with diagonal elements being $\partial \mu_{it} / \partial \eta_{it}$. $\mathbf{V}_i$ is defined as $\mathbf{V}_i = (\mathbf{A}_i^{1/2} \mathbf{R}_i(\alpha) \mathbf{A}_i^{1/2})/\phi$, with $\mathbf{A}_i$ being the $m_i \times m_i$ diagonal matrix v_{it} as the diagonal elements. The correlation structure for each cluster is denoted by $\mathbf{R}_i(\alpha)$, which is an $m_i \times m_i$ square matrix for cluster i, as described by Berhane and Tibshirani (1998), and that $\mathbf{R}(\alpha) = \mathbf{R}_1(\alpha) \oplus \mathbf{R}_2(\alpha) \oplus \cdots \oplus \mathbf{R}_n(\alpha)$. Estimation of ϕ, the dispersion parameter, and $\mathbf{R}_i(\alpha)$ are performed as described in Section 3.3 of Liang and Zeger (1986), while the only difference being that the total degrees of freedom (df_{total}) taken into consideration is represented here as the sum of total degrees of freedom due to the parametric terms ($df_{\text{parametric}}$) and the total effective degrees of freedom due to the nonparametric terms ($df_{\text{nonparametric}}$) expressed as below

$$df_{\text{total}} = df_{\text{parametric}} + df_{\text{nonparametric}} \tag{4}$$

where

$$df_{\text{parametric}} = p + 1 \tag{5}$$

accounting for the intercept term, and

$$df_{\text{nonparametric}} = df(f_1) + \cdots + df(f_q) \tag{6}$$

Effective degrees of freedom for each nonparametric term is estimated using the approach described by Berhane and Tibshirani (1998), where

$$df(f_j) = 2tr\mathbf{S}_j - tr(\mathbf{S}_j^T \mathbf{W} \mathbf{S}_j \mathbf{W}^{-1}) \tag{7}$$

for the jth predictor, for $j = 1, \cdots, q$, where $\mathbf{S}_j$ is the smoother matrix and $\mathbf{W}$ the weight matrix at convergence.

The procedure for updating $\mathbf{S}_j$ and $\mathbf{W}$ is described by Green and Silverman (1994), where the presence of ties and unsorted nature among the data points in the covariate is tackled by making use of the notion of $\sum m_i \times q$ incidence matrix, $\mathbf{N}$, with $\sum m_i$ being the total number of data points in the covariate undergoing smoothing, q the number of unique values of the covariate. The smoother matrix, $\mathbf{S}_j$, for the jth covariate is hence defined as

$$\mathbf{S}_j = \mathbf{N}_j (\mathbf{N}_j^T \mathbf{W} \mathbf{N}_j + \lambda_j \mathbf{K}_j)^{-1} \mathbf{N}_j^T \mathbf{W} \tag{8}$$

where $\mathbf{N}_j$ refers to the incidence matrix, λ_j refer to the smoothing parameter, and $\mathbf{K}_j$ the basis matrix for the jth predictor. Calculation of λ_j by cross-validation is described by Hastie and Tibshirani (1990), while details on the construction of $\mathbf{K}_j$ is found in Green and Silverman (1994).

The local scoring algorithm for maximizing the penalized quasi-likelihood follows in vein with that described in Berhane and Tibshirani (1998), and Green and Silverman (1994). Methods on covariance estimation for nonparametric terms are detailed in Berhane and Tibshirani (1998) where the empirical covariance for the jth covariate is approximated as

$$\mathsf{COV}_{\text{emp}}(\mathbf{f}_j) = \mathbf{S}_j \mathbf{W}^{-1} \mathbf{U} \mathbf{U}^T \mathbf{W}^{-1} \mathbf{S}_j^T \tag{9}$$

where $\mathbf{U} = \mathbf{D}\mathbf{V}^{-1}(\mathbf{y} - \mu)$, together with $\mathbf{S}_j$ and $\mathbf{W}$ evaluated at convergence. The calculation of empirical chi-squared statistics is also described in Berhane and Tibshirani (1998).

2. Implementation of CDGAM

The code discussed in this paper has been developed under S-PLUS 2000 Professional Release 1 for Windows.

In addition to some auxiliary scripts, the library contains the following main functions:

`cdgam`	function to fit the CDGAM
`cdgam.par`	script called from `cdgam` to fit the parametric part of the model
`cdgam.nonpar`	script called from `cdgam` to fit the nonparametric part of the model
`summary.cdgam`	function to display the summary results of model fitting performed by `cdgam`
`plot.cdgam`	function to plot the estimated functional form against the respective covariates

2.1. General schematics

The present **cdgam** implementation involves three major steps:

1. estimation of starting values for iterations in `cdgam` by fitting a generalized additive model under independent correlation structure as described by Hastie and Tibshirani (1990),

2. estimation of the correlation matrix for each cluster and fitting the parametric portion of `cdgam` using the framework of GEE,

3. estimation of the non-parametric portion of the `cdgam`.

In Step 1, a generalized additive model is being fitted under the independence ($\mathbf{R}(\alpha)$) framework, but not using the GEE sandwich method, in order to obtain fitted values for the parametric and nonparametric covariates to be used in Step 2. The script in S for performing Step 1 is `gam()`, an algorithm in S-PLUS that fits the generalized additive model when the data points are not correlated. See Chambers and Hastie (1993) for operation details.

Calculations for Step 2 and Step 3 are coordinated by a script called `cdgam()`. Step 2 is executed by `cdgam.par()`. Step 3 is executed by `cdgam.nonpar()`. `cdgam()` separates the covariates into two groups. Covariates requiring parametric estimation are passed, within `cdgam()`, to `cdgam.par()` where conventional **GEE** fitting is performed until local convergence is reached. Then, the results from `cdgam.par()` is passed into `cdgam.nonpar()` to perform the nonparametric estimation until local convergence criteria is reached. Finally, global convergence of the parametric and nonparametric covariates is checked.

The presence of local and global covergence arise from the nature of the iteration architecture. The outer loop is a local scoring procedure. In the inner loop, the Fisher scoring iteration is performed in `cdgam.par()` while the Gauss-Seidel iteration is performed in `cdgam.nonpar()`. Local convergence criteria refers to the convergence criteria used within either `cdgam.par()` or `cdgam.nonpar()`, while global convergence criteria refers to the criteria used within `cdgam()`.

Local convergence within `cdgam.par()` is checked by measuring the values of absolute difference between each of the estimated coefficients from the present Fisher scoring interation

(β_{new}) and the immediately preceeding iteration (β_{old}), defined as $\Delta(\beta_{\text{new}}, \beta_{\text{old}})$. Parametric local convergence is considered reached when all values are smaller than a predetermined value, called the local tolerance. In `cdgam.par()`, the local tolerance is set at 5×10^{-4}, and it can be adjusted by the user within `cdgam.par()`. Hence the parametric local convergence for `cdgam.par()` is expressed as

$$\Delta(\beta_{\text{new}}, \beta_{\text{old}}) = \max(\|\beta_{\text{new}} - \beta_{\text{old}}\|) \tag{10}$$

Local convergence within `cdgam.nonpar()` is checked by measuring the fraction of absolute change in the values of the estimated nonparametric functions between those estimated from one Gauss-Seidel iteration (f_{new}) and the immediately preceeding iteration (f_{old}), defined as $\Delta(f_{\text{new}}, f_{\text{old}})$. Nonparametric local convergence is considered reached when all values are smaller than a predetermined value, called the local tolerance. In `cdgam.nonpar()`, the local tolerance is set at 0.02, and it can be adjusted by the user within this script. Hence the nonparametric local convergence is:

$$\Delta(f_{\text{new}}, f_{\text{old}}) = \max\left\{\frac{(\|f_{\text{new}} - f_{\text{old}}\|}{\|f_{old\|}}\right\} \tag{11}$$

where, in the spirit of Equation (1), we can define $f_{\text{new}} = f_{1,new} + \cdots + f_{q,new}$ and $f_{\text{old}} = f_{1,old} + \cdots + f_{q,old}$ respectively.

Global covergence within `cdgam()` is checked by measuring the fraction of absolute change in the values of the estimated nonparametric functions between those estimated from one local-scoring iteration (F_{new}) and the immediately preceeding local-scoring iteration (F_{old}), defined as $\Delta(F_{\text{new}}, F_{\text{old}})$. Nonparametric global convergence is considered reached when all values are smaller than a predetermined value, called the global tolerance. Similar to the nonparametric local convergence criteria, the global tolerance in `cdgam()` is set at 5×10^{-5}, and it can be adjusted by the user within `cdgam()`. It follows that

$$\Delta(F_{\text{new}}, F_{\text{old}}) = \max\left\{\frac{(\|F_{\text{new}} - F_{\text{old}}\|}{\|F_{old\|}}\right\} \tag{12}$$

where, we define $F_{\text{new}} = F_{1,new} + \cdots + F_{q,new}$ and $F_{\text{old}} = F_{1,old} + \cdots + F_{q,old}$ respectively.

The `cdgam()` script makes use of the results of calculation performed by `gam(···,x=T)` in order to retrieve three entities: (1) data matrix for the nonparametric terms, (2) effective degree of freedom for each nonparametric term, and (3) estimated values for the nonparametric terms.

The data matrix for the nonparametric terms is used for calculation of the incidence matrices for the jth nonparametric term, $\mathbf{N}_j$. This, together with the estimation of smoothing parameter λ_j and basis matrix $\mathbf{K}_j$, are used to construct the smoother matrix $\mathbf{S}_j$ for the jth nonparametric covariate later on. For fast calculation, the calculation of λ_j is made based-on the S-PLUS built-in function called `smooth.spline()`. For a domain of unique x values $x_1, x_2, \cdots, x_t$ on some interval $[x_1, x_t]$, satisfying $x_1 < x_2 < \cdots < x_t$ over which smoothing is carried out, where $x_1 < x_2 < \cdots < x_t$, the `spar` value of the object fitted by `smooth.spline()` returns a value, denoted as ξ, that is connected to λ_j as below:

$$\lambda_j = \xi(x_t - x_1)^3 \tag{13}$$

The effective degrees of freedom and the estimated values for the nonparametric terms are then passed, within `cdgam()`, into `cdgam.par()` where the correlation structure $\mathbf{R}(\alpha)$,and the dispersion scale parameter ϕ are estimated. Then, using these two entities, the Fisher scoring iterations for estimating the parametric covariates is performed. At local convergence, tests for significance as described in Liang and Zeger (1986) is carried out.

The results of estimated values for the parametric covariates, $\mathbf{R}(\alpha)$, and ϕ at convergence in `cdgam.par()` are then used by `cdgam.nonpar()` to perform Gauss-Seidel iterations in order to fit the nonparametric terms. At local convergence, the chi-squared test of significance as described in Berhane and Tibshirani (1998) is carried out. Then, the updated effective degree of freedom and estimated values for the nonparametric terms are passed into `cdgam.par()` again for the second local scoring iteration until global convergence is reached.

2.2. Handling of intracluster correlation structure

The calculation of within cluster correlation matrices $\mathbf{R}_i(\alpha)$ is performed in the script called `cdgam.par()` where the Pearson's residual calculated using the most update $\eta_{\text{parametric},it}$, $\eta_{\text{nonparametric},it}$, $df_{\text{parametric}}$ and $df_{\text{nonparametric}}$ for each cluster. The options of correlation structures currently supported are:

1. exchangeable correlation, otherwise known as uniform correlation model, where there is a positive correlation coefficient, α, between any two measurements within the same cluster and that α is the same across all clusters;
2. stratified exchangeable correlation, where there is a positive correlation coefficient, α_i, between any two measurements within the same cluster, and variation of α_i across clusters is allowed;
3. first order autoregressive model for evenly spaced time scale; and
4. first order autoregressive model for unevenly spaced time scale. The methods applied in this algorithm follows directly from that described in Liang and Zeger (1986).

Since $\mathbf{R}(\alpha) = \mathbf{R}_1(\alpha) \oplus \mathbf{R}_2(\alpha) \oplus \cdots \oplus \mathbf{R}_n(\alpha)$ is a blocked-diagonal matrix, calculation of $\mathbf{R}^{-1}(\alpha)$ required in the calculation of weights $\mathbf{W}$ makes use of the identity $\mathbf{R}^{-1}(\alpha) = \mathbf{R}_1^{-1}(\alpha) \oplus \mathbf{R}_2^{-1}(\alpha) \oplus \cdots \oplus \mathbf{R}_n^{-1}(\alpha)$ in order to save computing memory and time. Therefore, two subroutines options need to be specified in `cdgam()`, where the one assigned to the option called `alpfun` specifies one of the four options of correlation structures described above, and one to the option called wcorigen calculates $\mathbf{R}_i^{-1}(\alpha)$ for clusters $i = 1, \cdots n$.

In addition, there is an input option, called `cor.met`, in `cdgam()` that needs to be specified for calculation of certain types of correlation structure specification. For uniform correlation structure, it need not be specified. For stratified uniform correlation structure, the variable assigned to `cor.met` is the pointer variable identifying the cluster origin of the data points. For first order autoregressive model for evenly spaced time scale, assignment to cor.met constitutes a matrix with two columns, the first is the pointer variable identifying the cluster origin of the data points, while the second is the recording of the times at which the corresponding data points were observed.

This arrangement leaves room for extensions by users where, by simply writing the scripts for `alpfun` and `wcorigen`, one can deploy the present **cdgam** to cater for other types of intracluster correlation structures.

2.3. Choice of smoothing parameter

Leave-one-out generalized cross-validation is computationally intensive in the setting of generalized semiparametric modelling, requiring re-computation of the entire iterative fit for each value of λ on a grid in order to carry out the necessary minimization over λ (Green and Silverman 1994), and its performance in practice is sometimes questionable (Hastie and Tibshirani 1990).

The use of empirical-bias bandwith selector (EBBS) (Ruppert 1997) is an alternative approach when applied to clustered data using profile likelihood-kernel regression GEEs. However, Lin and Carroll (2001b) and Lin and Carroll (2001a) noted that the profile likelihood-kernel regression is not semiparametric efficient when correlation at the observation-level is taken into account, and, in order to achieve consistency, arbitrary undersmoothing or assuming correlation structure at observation-level to be independent becomes necessary.

In the light of this situation, the present approach is to obtain λ using `smooth.spline()`, a smoothing function generic to S-PLUS, based on cross-validation as a starting point. Then, by graphical inspection, and model refit by adjusting the value of λ, a reasonable degree of smoothing is achieved (See demonstration in later section dealing with Infectious Disease Data). Changing the value of λ in multiples of 10 in the initial phase of modelling process, and reduce magnitude of change in λ later on for fine tuning once a reasonable fit is obtained is a useful strategy to speed up the modelling process.

2.4. Choice of smoothing technique

Lin and Carroll (2001b) reported that conventional kernel method, when used in semiparametric form of PA-GEE, does not produce $n^{1/2}$ consistent estiamtes of coefficients for the parametric covariates. Subsequent work reported in Lin, Wang, Welsh, and Carroll (2004) justified the use of smoothing splines for clustered data in this setting because splines are non-local, and are able to account for intra-cluster correlation, as opposed to conventional kernel methods. Therefore, we have chosen to employ smoothing splines for nonparametric covariates handling as described in Berhane and Tibshirani (1998).

2.5. Choice of platform

The initial conception of this project evolves from codes written in S-PLUS. In S-PLUS, the calculation of Equation (3) involves extending the existing canonical link for the exponential family to provide $\partial\mu/\partial\eta$. For convenience, we modified `glm.links` in the S-PLUS into `yags.links` used in **cdgam**. However, there is no exact equivalence of `glm.links` in the R environment. Hence, a major portion of the program needs to be re-written to enable migration from S-PLUS to R, which is currently under way. Once the R version of **cdgam** becomes available, it will be submitted to CRAN for public access.

2.6. Limitations

1. For the case of

$$\eta_{it} = \beta_0 + f_1(X_{it(p+1)}) + \cdots + f_p(X_{it(p+q)}) \tag{14}$$

   the formula in `cdgam()` is set to be `formula = y ~ 1`, where β_0 is taken as the centering value for the model fit.

However, `cdgam()` does not cater for

$$\eta_{it} = \beta_0 + \beta_1 X_{it(p+1)} + \cdots + \beta_q X_{it(p+q)} \tag{15}$$

because it reduces to the special case described by Liang and Zeger (1986), and there are already available libraries such as **gee** and (yags) that can handle this situation.

2. Due to the transparent nature of the present implementation of **cdgam**, all codes are written in S, and admittedly, the computation is slow compared to compiled languages such as C and FORTRAN. This is made worse by the heavy demand of inverting large matrices. We trade speed for ease of maintainance, and leaves room for further refinement. It will also allow users to modify the codes according to their specific needs, including writing scripts to handle correlation structures not included here, and adjusting presentation of the output.

3. Example with simulated data

In this section, we illustrate the use of our routines on a simulated example (similar to the simulation example in Section 4.3 of Berhane and Tibshirani (1998).

We consider three predictors given by

$$f_1(x_1) = x_1^{1.5}, \qquad f_2(x_2) = \cos\left(\frac{2.5\pi x_2}{1+3x_2^2}\right), \qquad f_3(x_3) = x_3. \tag{16}$$

for a model given by

$$\text{logit}(y_{ij}) = f_1(x_1) + f_2(x_2) + f_3(x_3) \tag{17}$$

where x_1,x_2, and x_3 are generated from U(0,1) in the framework of Lee (1993), following a similar strategy described by Berhane and Tibshirani (1998). In that setting, the intra-cluster correlation is expressed in terms of ψ with $(0 < \psi < 1)$, where a low value of ψ signifies a high correlation, and vice versa. We generated 150 clusters, with each cluster containing 3 observations, with a high exchangeable intra-cluster correlation ($\psi = 0.3$). We have enclosed the dataset, denoted as `cdgam.data`. The dataset contains the followings variables:

`individual`	Cluster pointer where observations of the same cluster share the same number.
`x1`	Random number used to generate $f(x_1)$.
`fx1`	Values for $x_1^{1.5}$ as defined above
`x2`	Random number used to generate $f(x_2)$
`fx2`	Values for $\cos((2.5\pi x2)/(1+3x_2^2))$ as defined above.
`x3`	Random number used to generate $f(x_3)$.
`fx3`	Values for x_3 as defined above.
`y`	Binary variables generated as the response variable

We first fit the generalized additive model under independent correlation structure as described by Hastie and Tibshirani (1990).

```
> step1 <- gam(y ~ s(x1) + s(x2) + x3, family = binomial, x = T,
+ data = cdgam.data)
```

Then, we fit the model under exchangeable correlation structure with logistic link. Note that in `step1`, we need to specify the entire formula, as opposed to specifying only the parametric covariates in `step2`.

```
> step2 <- cdgam(formula = y ~ x3, id = individual, family = binomial,
+ corstr = "exchangeable", gamob = step1, data = cdgam.data)
```

The summary of the model fit result can be obtained by the following:

```
> summary(step2)

$call:
cdgam(formula = y ~ x3, id = individual, family = binomial,
corstr = "exchangeable", gamob = step1, data = cdgam.data)

$parametric.coefficients:
             Estimate Naive S.E.    Naive z Robust S.E.   Robust z
(Intercept) -0.142578  0.3946116 -0.3613122   0.3439056 -0.4145846
         x3  3.746447  0.7466218  5.0178646   0.6338137  5.9109591

$coef.smooth:
      Df exact Df approx empirical Chisq     P(Chi)
s(x1) 59.84610  63.38170        47.58698 0.85314782
s(x2) 65.55037  69.21069        86.35508 0.03634361

$scale:
[1] 1.390494

$alpha:
[1] 0.4480969

attr(, "class"):
[1] "summary.cdgam"
```

The following produces a plot of the estimated values of $f_2(x_2)$ against x_2 and adds the line representing plot of the actual values $f_2(x_2)$ against x_2 (See Figure 1):

```
> plot(step2, ci = T, resid = T, j = 2)
> attach(cdgam.data)
> lines(smooth.spline(x2, fx2), lty = 4, lwd = 4)
```

4. Example using infectious disease data

We apply the model to analysing the longitudinal infectious disease data involving 275 preschool-age children who were re-examined in 3 monthly intervals for 18 months, ascertaining the presence of respiratory infection (yes=1, no=0). This dataset was described by

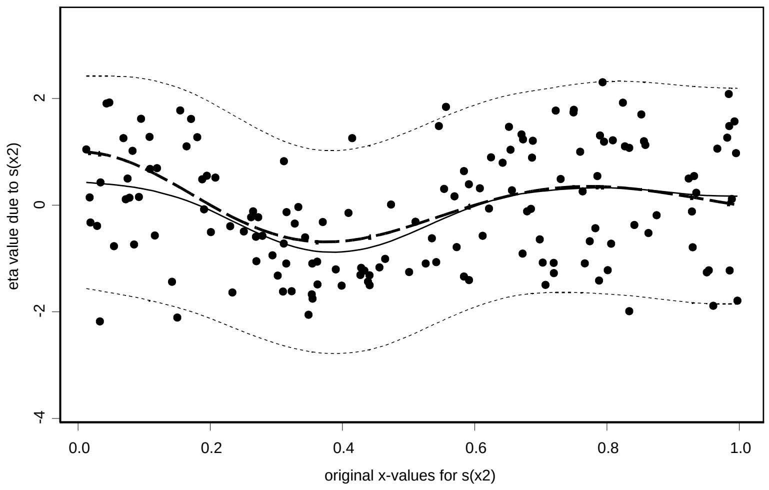

Figure 1: The plot of $\widehat{f}_2(x_2)$ against x_2 where $\widehat{f}$ is the estimation of $f_2(x_2)$ where $f_2(x_2) = \cos(2.5\pi x_2)/(1+3x_2^2)$ with the dots representing the residuals, the solid dashed line represents $f_2(x_2)$, the fine continuous line represents $\widehat{f}_2(x_2)$, and the two fine dotted lines bounding $f_2(x_2)$ and $\widehat{f}_2(x_2)$ on either side represent the 95interval.

Zeger and Karim (1991) and has been used in Lin and Carroll (2001b) and Lin and Carroll (2001a) to perform generalized additive marginal modelling analyses. We have enclosed this dataset, and called it `indon`. The description of each variable is as follows:

`id`	Cluster identifier, where observations from the same cluster share a common number.
`res.infect`	Binary variable with presence of respiratory infecton=1, otherwise=0.
`xeroph`	Presence of Vitamin A deficiency = 1, otherwise = 0.
`cos.visit`	Seasonal cosine.
`sin.visit`	Seasonal sine.
`sex`	Gender of the subject.
`height`	Height for age.
`stunt`	Presence of stunting.
`visit`	The number of visit when the observation was made.
`season`	1=Spring, 2=Summer, 3=Autumn, 4=Winter.
`age`	Age in years at the time of observation.
`baseline.age`	Age in years at the time of recruitment into study.

4.1. Initial phase of data modelling using GAM

Select part of the data for calculation:

```
> indon.sub <- indon[c(1:300),]
> length(unique(indon.sub[,1]))

[1] 71
```

There are 71 clusters in this subset.
Formulate the first step of modelling process using gam:

```
> step1 <- gam(formula = res.infect ~ s(age) + xeroph + cos.visit +
+ sin.visit + sex + height + stunt, family = binomial, x = T, data = indon.sub)
> summary(step1)

Call: gam(formula = res.infect ~ s(age) + xeroph + cos.visit +
sin.visit + sex + height + stunt, family = binomial, data = indon.sub, x = T)
Deviance Residuals:
        Min         1Q     Median         3Q      Max
 -0.9851055 -0.6414985 -0.4321241 -0.2996751 2.709284

(Dispersion Parameter for Binomial family taken to be 1 )

    Null Deviance: 253.6255 on 299 degrees of freedom

Residual Deviance: 228.4742 on 289.1412 degrees of freedom

Number of Local Scoring Iterations: 4

DF for Terms and Chi-squares for Nonparametric Effects

            Df Npar Df Npar Chisq     P(Chi)
(Intercept)  1
     s(age)  1     2.9   8.242089 0.03678064
     xeroph  1
  cos.visit  1
  sin.visit  1
        sex  1
     height  1
      stunt  1
```

Since the approximate nonparametric degree of freedoms for `s(age)` is 2.9, the `step1` is refitted with `s(age, df=3)` so that it provides a better set of starting values of age for `cdgam()` formulation. `step1` provides the starting values for model fitting in subsequent sections:

```
> step1 <- gam(formula = res.infect ~ s(age,df=3) + xeroph + cos.visit +
+ sin.visit + sex + height + stunt, family = binomial, x = T,data = indon.sub)
```

The degree of freedom for a nonparametric covariate is related to the smoothing parameter λ used in `smooth.spline()` called from `gam()`. When the nonparametric covariate is specified as `s(age)`, the `smooth.spline()` algorithm optimize the value of λ used in smoothing, and it is reflected as the nonparametric degree of freedom. Hence, refitting the model by specifying the nonparametric covariate as above using the specification `s(age, df=3)`, the degree of smoothing is controlled so that the value of λ used in `gam()` is that optimized by `smooth.spline()`. We should refrain from using the nonparametric degree of freedom produced with `s(age, df=3)` to refit using `gam()` because the aim is to obtain the optimal smoothing for the nonparametric covariate age, as opposed to obtain the optimal smoothing for the smoothed for of the nonparametric covariate with 3 degrees of freedom `s(age, df=3)`.

4.2. Modelling under exchangeable correlation structure

We first fit the model assuming exchangeable correlation structure:

```
> step2.ex.1 <- cdgam(formula = res.infect~xeroph + cos.visit + sin.visit +
+ sex + height + stunt, id = id, family = binomial, corstr = "exchangeable",
+ gamob = step1, data = indon.sub)
> summary(step2.ex.1)

$call:
cdgam(formula = res.infect ~ xeroph + cos.visit + sin.visit + sex + height +
stunt, id = id, family = binomial, corstr = "exchangeable", gamob = step1,
data = indon.sub)

$parametric.coefficients:
               Estimate Naive S.E.     Naive z Robust S.E.    Robust z
(Intercept) -2.59349718  0.3184684 -8.14365674  0.26718656 -9.70669032
     xeroph  0.28062820  1.2883989  0.21781158  0.80096584  0.35036225
  cos.visit -0.48070271  0.3361932 -1.42984087  0.28653001 -1.67766966
  sin.visit -0.23239116  0.3201893 -0.72579301  0.22724533 -1.02264439
        sex -0.01935372  0.3939345 -0.04912928  0.31438041 -0.06156147
     height -0.05338726  0.0532786 -1.00203941  0.04880356 -1.09392144
      stunt -0.18177428  0.8179407 -0.22223407  0.67382854 -0.26976340

$coef.smooth:
               Df exact Df approx empirical Chisq    P(Chi)
s(age, df = 3) 57.34003  62.75955        20.52969 0.9999969

$scale:
[1] 1.473711

$alpha:
[1] -0.03638948

attr(, "class"):
[1] "summary.cdgam"
```

Plotting the fitted function of age against age shows that the risk of respiratory infection is seen to increase until the age of 2, and then decrease after that. (See Figure 2)

```
> plot(step2.ex.1, ci = T, resid = T)
```

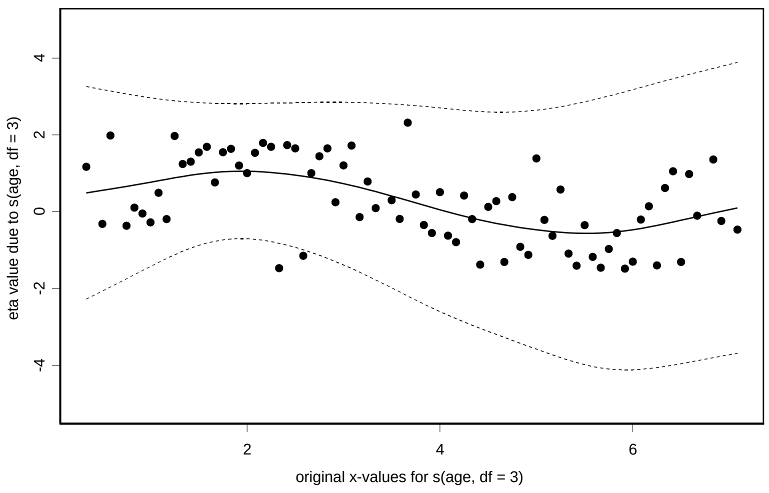

Figure 2: The plot of $\widehat{f}_{age}$ against age where the dots represent residuals from the indon data subset estimated assuming exchangeable intracluster correlation structure, the fine continuous line represents estimated contribution to the risk of respiratory infection over age, and the two dotted fine lines its 95% confidence interval bound.

4.3. Modelling under AR(1) correlation structure

Fitting the model assuming an AR(1) correlation structure as it involves time factor, and specifying the parameter visit as the correlation metameter:

```
> step2.ar1.1 <- cdgam(formula = res.infect~xeroph + cos.visit + sin.visit +
+ sex + height + stunt, id = id, cor.met = visit, family = binomial,
+ corstr = "ar1", gamob = step1, data = indon.sub)
> summary(step2.ar1.1)
```

```
$call:
cdgam(formula = res.infect ~ xeroph + cos.visit + sin.visit + sex + height +
stunt, id = id, cor.met = visit, family = binomial, corstr = "ar1",
gamob = step1, data = indon.sub)

$parametric.coefficients:
               Estimate Naive S.E.    Naive z Robust S.E.   Robust z
(Intercept) -2.47755331 0.30066914 -8.2401316  0.27204927 -9.1070022
     xeroph  0.19130901 1.20910944  0.1582231  0.73388062  0.2606814
  cos.visit -0.46533788 0.30787297 -1.5114606  0.28895306 -1.6104273
  sin.visit -0.19771496 0.29324839 -0.6742235  0.22741069 -0.8694181
```

```
        sex -0.09426448 0.37910997 -0.2486468  0.31784530 -0.2965735
     height -0.05628223 0.05030511 -1.1188174  0.04832306 -1.1647075
      stunt -0.19887753 0.77582578 -0.2563430  0.66140927 -0.3006875

$coef.smooth:
                Df exact Df approx empirical Chisq P(Chi)
s(age, df = 3) 55.90675  60.93737        15.06681      1

$scale:
[1] 1.267612

$alpha:
[1] -0.05534506

attr(, "class"):
[1] "summary.cdgam"
```

Plotting the fitted function of age against age shows there is over fitting due to a small value of λ, retrievable from `step2.ar1.1`. (See Figure 3)

```
> plot(step2.ar1.1, ci = T, resid = T)
```

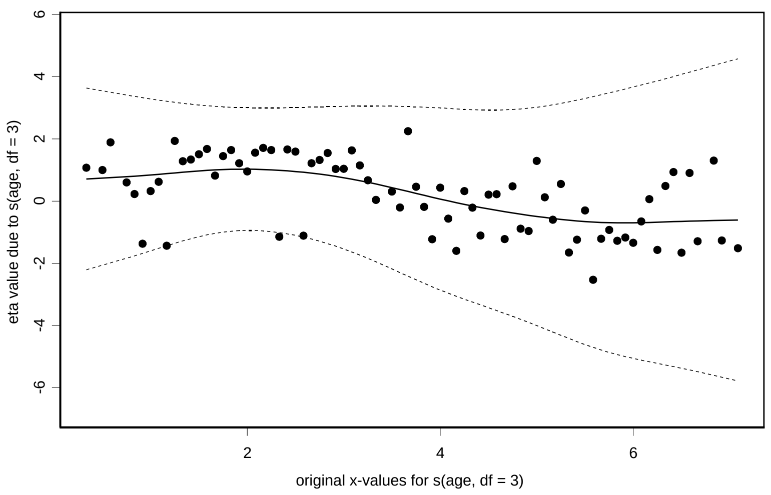

Figure 3: The plot of $\widehat{f}_{age}$ against age where the dots represent residuals from the indon data subset estimated assuming AR(1) intracluster correlation structure with $\lambda = 4.478831 \times 10^{-6}$, the fine continuous line represents estimated contribution to the risk of respiratory infection over age, and the two dotted fine lines its 95% confidence interval bound.

Therefore, a new value for λ is specified, arbitrarily set as 10 times that in `step2.ar1.1` and a new model is fitted. (See Section 5 for the rationale of such choice of factor of expansion for λ)

```
> step2.ar1.1$lambda

[[1]]:
[1] 4.478831e-006

> lambda.new <- 10*unlist(step2.ar1.1$lambda)
> step2.ar1.2 <- cdgam(formula = res.infect~xeroph + cos.visit + sin.visit +
+ sex + height + stunt, id = id, lambda=lambda.new, cor.met = visit,
+ family =binomial, corstr = "ar1", gamob = step1, data = indon.sub)
```

The summary results of model fit with the new smoothing parameter λ' is as follows:

```
> summary(step2.ar1.2)

$call:
cdgam(formula = res.infect ~ xeroph + cos.visit + sin.visit + sex + height +
stunt, id = id, lambda = lambda.new, cor.met = visit, family = binomial,
corstr = "ar1", gamob = step1, data = indon.sub)

$parametric.coefficients:
               Estimate Naive S.E.    Naive z Robust S.E.   Robust z
(Intercept) -2.67058236  0.2950029 -9.0527314   0.2711807 -9.8479804
     xeroph  0.23805501  1.2055512  0.1974657   0.7619607  0.3124243
  cos.visit -0.46098502  0.3018453 -1.5272228   0.2858164 -1.6128711
  sin.visit -0.19783700  0.2882948 -0.6862315   0.2278433 -0.8683028
        sex -0.07251887  0.3741501 -0.1938229   0.3198790 -0.2267072
     height -0.05315972  0.0495429 -1.0730039   0.0470079 -1.1308678
      stunt -0.32053131  0.7654951 -0.4187242   0.6768653 -0.4735526

$coef.smooth:
               Df exact Df approx empirical Chisq     P(Chi)
s(age, df = 3) 40.20359  41.77025        54.38034 0.05418916

$scale:
[1] 1.242216

$alpha:
[1] -0.04773346

attr(, "class"):
[1] "summary.cdgam"
```

Plotting the fitted function of age against age shows a reasonable degree of roughness penalty. (See Figure 4)

```
> plot(step2.ar1.2, ci = T, resid = T)
```

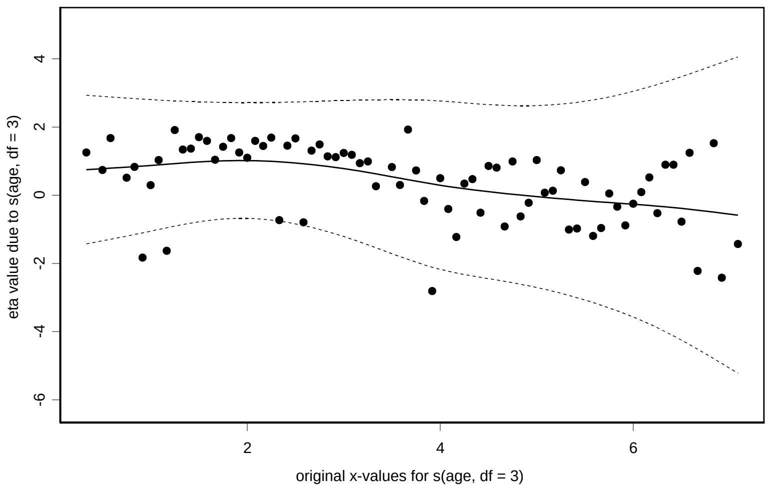

Figure 4: The plot of $\widehat{f}_{age}$ against age where the dots represent residuals from the indon data subset estimated assuming AR(1) intracluster correlation structure with $\lambda' = 10 \times \lambda = 4.478831 \times 10^{-5}$, the fine continuous line represents estimated contribution to the risk of respiratory infection over age, and the two dotted fine lines its 95% confidence interval bound.

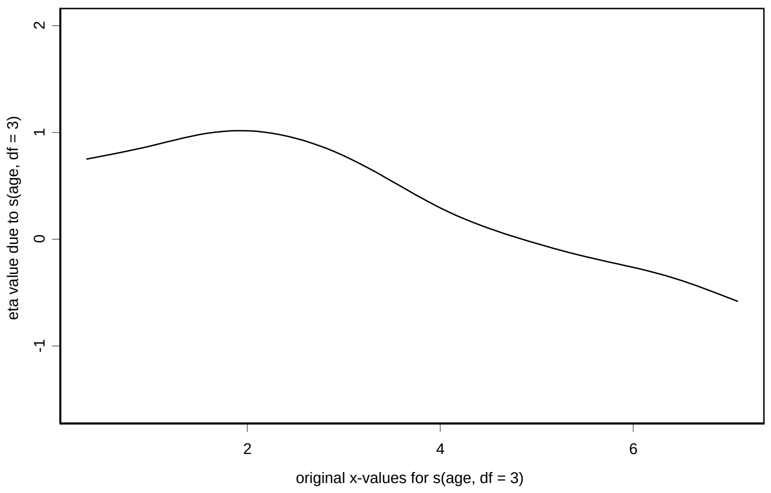

Figure 5: The plot of $\widehat{f}_{age}$ against age where the dots represent residuals from the indon data subset estimated assuming AR(1) intracluster correlation structure with $\lambda' = 10 \times \lambda = 4.478831 \times 10^{-5}$, the fine continuous line represents estimated contribution to the risk of respiratory infection over age, and the two dotted fine lines its 95% confidence interval bound.

Plotting the fitted function of age against age without standard error bands and residuals allows a closer inspection of the trend of respiratory infection risk against age. (See Figure 5) The risk of respiratory infection is noted to increase since birth to 2 years old, and then decreases thereafter. Similar observation is reported in Figures 3 and 4 of Lin and Carroll (2001b).

```
> plot(step2.ar1.2)
```

Due to the slow convergence rate, modelling for `step2.ex.1`, `step2.ar1.1`, and `step2.ar1.2`, require up to 6 back-fitting loops, and up to 200 Gauss-Seidel interations within each loop.

References

Berhane K, Tibshirani RJ (1998). "Generalized Additive Models for Longitudinal Data." *The Canadian Journal of Statistics*, **26**, 517–535.

Chambers JM, Hastie TJ (1993). *Statistical Models in S*. Chapman and Hall, London.

Green PJ, Silverman BW (1994). *Nonparametric Regression and Generalized Linear Models*. Chapman and Hall, London.

Hastie TJ, Tibshirani RJ (1990). *Generalized Additive Models*. Chapman and Hall, London.

Lee AJ (1993). "Generating Random Binary Deviates Having Fixed Marginal Distribution and Specified Degrees of Association." *The American Statistician*, **47**, 209–215.

Liang KY, Zeger SL (1986). "Longitudinal Data Analysis Using Generalized Linear Models." *Biometrika*, **73**, 13–22.

Lin X, Carroll RJ (2001a). "Semiparametric Regression for Clustered Data." *Biometrika*, **88**, 1179–1185.

Lin X, Carroll RJ (2001b). "Semiparametric Regression for Clustered Data Using Generalized Estimating Equations." *Journal of American Statistical Association*, **96**, 1045–1056.

Lin X, Wang N, Welsh AH, Carroll RJ (2004). "Equivalent Kernels of Smoothing Splines in Nonparametric Regression for Clustered/Longitudinal Data." *Biometrika*, **91**, 177–193.

McCullagh P, Nelder JA (1989). *Generalized Linear Models*. Chapman and Hall, London.

Ruppert D (1997). "Empirical-Bias Bandwith for Local Polynomial Nonparametric Regression and Density Estimation." *Journal of American Statistical Association*, **92**, 1049–1062.

Zeger SL, Karim MR (1991). "Generalized Linear Models with Random Effects: A Gibb's Sampling Approach." *Journal of American Statistical Association*, **86**, 79–86.

A. Instructions for using `cdgam()`

The code is available as an archived directory containing the files listed below. Copy the entire directory called **cdgam** to the library folder in S-PLUS. The code comes in two versions, one designed to run under S-PLUS 2000 Professional Edition for Windows, while the other under S-PLUS 5.1 on UNIX.

Listed below is the detailed alternative for each input and a description of the output. This is available as a help file for the library.

Usage

```
cdgam(formula, id, lambda = NULL, weights = NULL, cor.met = NULL,
      family = gaussian, alpfun = NULL, scalefun = BT.scalefun,
      wcorigen = identni, tol = 0.001, contrasts = NULL,
      corstr = c("independence", "exchangeable", "ar1", "unstructured"),
      maxiter = 25, verbose = F, gamob, data)
```

Required arguments

`formula`	Follows the Response∼covariates convention, but the covariates should include only the paramtetric terms only.
`id`	A vector of numbers serving as cluster pointer, where datapoints from the same cluster share the same number.
`corstr`	Specified correlation structure, taking either independence, exchangeable, AR(1) or unstructured. If it is not supplied, then, a combination of `cor.met`, `alpfun` and `wcorigen` need to be specified depending on the correlation structure.
`gamob`	The object of the S-plus generic function `gam()`, the formula of which contain all parametric and nonparametric terms, and the option for design matrix return is turned on.
`data`	data frame.

Optional arguments

`lambda`	The smoothing parameter referred to as λ in the cubic spline smoothing that minimizes $(y-g)^T(y-g)+\lambda \mathrm{g}\mathbf{K}\mathrm{g}$. New values of λ, in numeric or vector form, can be supplied to alter the degree of smoothing.
`weights`	Optional input allowing pre-specified weights
`cor.met`	The corelation metameter used for correlation calculation.
`family`	The family of the link function, covering the entire exponential family as supported by S-PLUS.
`alpfun`	This supplies the estimator function of correlation matrix $\mathbf{R}(\alpha)$ for each cluster.
`scalefun`	The default scale function is BT.scalefun.
`wcorigen`	No need to specify `wcorigen` if `corstr` is specified. However, if `corstr` is not specified, but `alpfun` is specified instead, then, `wcorigen` need to be specified.

`tol`	The tolerance limit defining convergence of the local scoring procedure.
`contrasts`	Optional entry for constrast can be supplied here.
`maxiter`	The maximum local scoring iterations allowed by default is 25.
`verbose`	logical.

Details

Corresponding `cor.met` for each correlation structure is as below:

Correlation structure	**cor.met**
exchangeable	need not specify
stratified exchangeable	a vector of numbers serving as cluster pointer, identical to the entry for input id.
AR(1)-evenly spaced time scale	a vector the times at which the corresponding data points were recorded.
AR(1)-unevenly spaced time scale	a matrix with two columns; the first being the cluster pointer, the second the times at which the corresponding data points were recorded.
unstructured	a vector providing the means for selecting appropriate elements for incomplete clusters from the time-saturated correlation matrix.

Corresponding `alpfun` for each correlation structure is as below:

Correlation structure	**alpfun**
exchangeable	`BT.exchalp`
stratified exchangeable	`BT.strat.exchalp`
AR(1)- evenly spaced time scale	`BT.prop.ar1alp`
AR(1)- unevenly spaced time scale	`LZ.ar1alp`
unstructured	`BT.prop.unstruc.alp`
independence	(no need to specify)

Corresponding `wcorigen` for each `alpfun` is as below:

alpfun	**wcorigen**
`BT.exchalp`	`excoriput` or `excori`
`BT.strat.exchalp`	`strat.excoriput`
`BT.prop.ar1alp`	`ar1.coriput`
`LZ.ar1alp`	`LZ.ar1.coriput`
`BT.prop.unstruc.alp`	`unstr.coriput`
independence (no `alpfun` needed)	`identni`

Output

The output is a list containing the following elements:

`para`	output from `cdgam.par()` pertaining to the parametric portion of the modelling process
`nonparametric`	output from `cdgam.nonpar()` pertaining to the nonparametric portion of the modelling process
`lambdaK`	the product of the smoothing parameter and the cubic spline basis matrices (Green & Silverman pp 13)
`lambda`	the value(s) of the smoothing parameter used in smoothing
`x.smooth`	the covariates requiring smoothing. When more than one, it is ordered by column from left to right as appeared in the formula supplied in `gam(..., x = T)`.
`incidence.matrix`	the list of incidence matrices used by each nonparametric term that allow for ties (Green & Silverman pp 65)
`final.eta`	the smoothed values of the nonparametric terms. When more than one, it is ordered by column from left to right as appears in the formula supplied in `gam(..., x = T)`.
`call`	the call that produce the results

The `para` portion of the output contain the followings:

`coefficients`	Values of the coefficients for the parametric terms at local convergence
`naive.parmvar`	the product of the scale parameter and the variance matrix
`robust.parmvar`	the sandwich estimate of variance
`alpha`	correlation parameter(s)
`phi`	scale parameter
`linear.predictors`	eta values for the parametric portion
`fitted.values`	mu values for the parametric portion, i.e., $g(\mu) = \eta$.
`pearson.resid`	Pearson's residuals
`iter`	number of Fisher's scoring loop for the last local scoring iteration at convergence.
`family`	the family of the link function

`rank`	the number of parametric parameters in the `cdgam.par()` fitting
`errorcode`	the error messages generated during calculation of `cdgam.par()`
`Rinv.i`	the list containing inverse of the working correlation matrix for each cluster to be used in `cdgam.nonpar()`
`MX`	the matrix containing all covariates information, by column, as ordered in the formula of `gam(..., x = T)`.
`b0`	fitted values of beta, the coefficients for the parametric terms

The `nonparametric` portion of the output contain the followings:

`T.s.emp`	Empirical calculation of chi-squared values for nonparametric terms (Berhane & Tibshirani Section 4.2)
`T.s.mb`	Model-based calculation of chi-squared values for nonparametric terms (Berhane & Tibshirani Section 4.2)
`T.s.chisq`	p-value for chi-squared based on approximate degrees of freedom (Berhane & Tibshirani Section 4.2)
`df.nl`	Degree of freedom according to Berhane & Tibshirani Equation 8.
`df.approx`	Degree of freedom approximated by 1.25trace(S)-0.5.
`eta.nonpar`	Sum of eta values for smoothing term(s) for each observation point.
`se.emp`	Empirical calculation of standard error (Berhane & Tibshirani Equation 26)
`se.mb`	Model-based calculation of standard error (Berhane & Tibshirani Equation 25)
`smooth.terms`	Matrix containing individual eta for each smoothed term, the addition of which by row-wise produces `eta.nonpar`
`smoother`	Smoother matrix for each smoothing term at convergence
`weight`	Weight matrices at convergence
`A.nonpar`	Variance matrix ($q \times q$)
`V.nonpar`	sandwich estimate of variance (non-parametric form of Berhane & Tibshirani Equation 4)
`V.inv`	inverse of `V.nonpar`

B. Instructions for using `plot.cdgam()`

Usage

```
plot(x, ci.option = c("mb", "emp"), j = NULL, ci = F, resid = F)
```

Required arguments

`x`	The fitted object from class `cdgam`

Optional arguments

`ci.option`	The option of using model-based (`ci.option = "mb"`) standard deviation for error-band plotting, or to use empirical (`ci.option = "emp"`) standard deviation. Default uses the model-based version (`ci.option = "mb"`).
`j`	The option of selecting the nonparametric term to plot. Default is plotting all nonparametric terms.
`ci`	The option of plotting the pointwise 95% standard error bands for the `eta.value` plotted. Default is not plotting the standard error bands.
`resid`	The option of plotting the fitted values for the `eta.value` plotted. Default is not plotting the points.

Details

The specification of `j` follows the sequence of the formula in the cdgam call; eg., a model fitting `y ~ x1 + x2 + s(x3) + s(x4)` corresponds to the following:

`j = 1`	`x3` versus `eta.value` for `x3`
`j = 2`	`x4` versus `eta.value` for `x4`

Output

A plot as specified by the options.

C. Description of scripts contained in the cdgam library

Function name	**Description**
`cdgam`	Fits the cdgam algorithm
`summary.cdgam`	Displays the summary results of the fitted cdgam object called via summary method to objects of the cdgam class
`plot.cdgam`	Plots the nonparametric terms against their corresponding covariates called via plot method using objects of the cdgam class
`cdgam.par`	Called internally by cdgam to fit the parametric portion of cdgam, and handle the intracluster correlation
`cdgam.nonpar`	Called internally by cdgam to fit the nonparametric portion of cdgam
`BT.exchalp`	Estimation for exchangeable correlation
`BT.strat.exchalp`	Estimation for exchangeable correlation, but different cluster may have different correlation coefficients
`BT.prop.ar1alp`	Estimation for auto-regressive process 1 with evenly spaced measurement time interval
`BT.prop.unstruc.alp`	Estimation of unstructured correlation
`LZ.ar1alp`	Estimation of auto-regressive process 1 allowing for unevenly spaced measurement time interval
`identni`	Produce identity matrix, serve as input for wcorigen under independence correlation structure
`excori`	Calculates inverse working correlation matrix for exchangeable correlation from each cluster
`excoriput`	Performs identical function as excori with improved efficiency
`strat.excoriput`	Calculates inverse working correlation matrix for exchangeable correlation from each cluster, allowing for correlation coefficient to vary between clusters
`ar1.coriput`	Calculates inverse working correlation matrix for AR(1) correlation from each cluster, assuming equally spaced observation
`unstr.coriput`	Calculates inverse working correlation matrix for unstructured correlation from each cluster
`LZ.ar1.coriput`	Calculates inverse working correlation matrix for AR(1) correlation from each cluster, allowing for unequally spaced observations
`yags.links`	Extension of glm.links
`bspline.smoother`	Construct basis matrix K for smoothing.
`pmat`	Sets up a classed list with appropriate class tag
`pmat2mat`	Convert a series of partitioned matrices to a single matrix
`fill`	Produces a list that is treated as a block diagonal matrix with ith block

`sum.pmat`	Sum over structure
`sum.pmat.block`	Sum over structure
`solve.pmat`	Inverse the structure
`solve.pmat.block`	Inverse the structure
`solve.pmat.block.default`	Inverse the structure
`solve.pmat.block.diag`	Inverse the structure
`t.pmat`	Transpose the structure
`t.pmat.block`	Transpose the structure
`det`	Calculate determinant of structure
`det.default`	Calculate determinant of structure
`det.pmat`	Calculate determinant of structure
`det.pmat.block`	Calculate determinant of structure
`det.pmat.block.default`	Calculate determinant of structure
`det.pmat.block.diag`	Calculate determinant of structure
`cdim`	Assignment of attribute to object in support of pmat2mat
`rdim`	Assignment of attribute to object in support of pmat2mat
`msplit`	Convert a matrix to a partitioned matrix
`dist2full`	Matrix manipulation for lower triangle
`full2tri` & `tri2full`	Matrix manipulation for lower triangle
`split.preserveord`	Create pointer for partitioning
`load.clustered.design`	Partition the design matrix
`load.clustered.outcome`	Partition the response variable
`load.bd.weight`	Partition the weight vector
`vsplit`	General function for matrix partition
`transfer.matfun`	Support load.clustered.design and load.clustered.outcome for matrix partition
`cdgam.data`	Data set for Section 9
`indon`	Actual dataset for Section 10
`ngau.m2ll`	Evaluates 2 times the Gaussian log likelihood of the residuals
`gau.hetex.alp`	Script asssignment for alp optional argument in cdgam to model Gaussian estimation for correlation parameters
`nexinv`	Analytic form of inverse of compound symmetry matrix
`make.exch.cor.genarg`	Script to support calculation of inverse working correlation matrix for Gaussian estimation for correlation parameters
`exch.cor`	Function in conjunction with make.exch.cor.genarg
`exch.gaussian.loglik`	Function in conjunction with make.exch.cor.genarg

Affiliation:

Lin Yee Hin
Private Medical Practitioner
Hong Kong
E-mail: lyhin@netvigator.com

Vincent Carey
Associate Professor of Medicine (Biostatistics)
Harvard Medical School
Channing Laboratory
181 Longwood Ave Boston MA
02115 USA
E-mail: stvjc@channing.harvard.edu

Journal of Statistical Software
August 2005, Volume 14, Issue 8.
http://www.jstatsoft.org/

Submitted: 2005-01-27
Accepted: 2005-08-31