CMAT Newsletter: May 2003

Wolfgang M. Hartmann

May 2003

1 General Remarks

1.1 Fixed Bugs

A number of bugs were fixed:

1. some bugs in `sem()` were fixed.
2. lots of bugs in `glim()` were fixed.
3. lots of bugs in `glmixd()` were fixed.

1.2 New Features: `glim()` and `glmixd()`

1. An additional input argument `xini` and two more return arguments `yhat` and `roc` were added to the `glim()` function:

   <gof,parm,sterr,conf,cov,typ1,typ3,yhat,roc> =
   glim(data,model<,optn<,class<,xini<,con>>>>)

   Matrix `yhat` has the same number of rows as the `data` input matrix and either 5 or 6 columns:

   1 the first column contains the observed value y_i of the response,

   2 the second column contains the predicted value $\hat{y}_i$ of the response,

   3 the third column contains the value $y_i - \hat{y}_i$ of the residual,

   4 column four contains the lower confidence limit for $\hat{y}$ with respect to α,

   5 column five contains the upper confidence limit for $\hat{y}$ with respect to α,

   6 only for binary response (target) Y an additional sixth column contains the diagonal entry of the *hat* matrix. Large values indicate outliers in the x space.

   The probability α is by default .05 or may be set in the `optn` argument. The value $alpha = .05$ corresponds to 95 % confidence limits.

   The `roc` output matrix has seven columns

   1 the first column contains the probability $\hat{y}_i$

   2 column 2 contains the number correctly predicted event responses pos_i

   3 column 3 contains the number correctly predicted nonevent responses neg_i

   4 column 4 contains the number of falsely predicted event responses $falpos_i$

   5 column 5 contains the number of falsely predicted event responses $falneg_i$

6 column 6 contains the sensitivity $sensit_i$

7 column seven contains the 1 minus specifity $1mspec_i$.

If CMAT would have a graphical tool, the ROC curve could be drawn by plotting *sensit* against $1mspec$.

The input argument `xini` is a vector specifying initial values of the parameter estimates. The parameters are used in the order of the model effects. A table indicating the parameter usage in the model is printed with the initial output.

The additional `pres` option was included for printing the observational statistics (yobs, yhat, residuals etc.). Note, if the `print` options is larger than three, then `pres` is on by default.

2. An additional input argument `xini` and three more return arguments `yhat`, `theta`, and `roc` were added to the `glmixd()` function:

<gof,parm,sterr,conf,cov,typ1,typ3,yhat,theta,roc> =
glmixd(data,model<,optn<,class<,random<,xini<,con>>>>>)

Matrix `yhat` has the same number of rows as the `data` input matrix and 3 columns:

1 the first column contains the observed value y_i of the response,

2 the second column contains the predicted value $\hat{y}_i$ of the response,

3 the third column contains the value $y_i - \hat{y}_i$ of the residual.

Matrix `theta` contains mean vectors and variance matrices of the random effects Θ_i. This return argument has the same number of rows as there are (first) levels of the ID variable in the input data set. Assuming nr random effects the columns contain the following results:

1 the first nr columns contain the means of the random effects Θ_i,

2 the next $nr(nr+1)/2$ columns contain the lower triangle of the variance-covariance matrix of the random effects Θ_i,

3 only if there is a weight variable used in the data set a last column is added containing the value of the weight for each level of the ID variable.

The `roc` output matrix has seven columns

1 the first column contains the probability $\hat{y}_i$

2 column two contains the number correctly predicted event responses pos_i

3 column three contains the number correctly predicted nonevent responses neg_i

4 column four contains the number of falsely predicted event responses $falpos_i$

5 column five contains the number of falsely predicted event responses $falneg_i$

6 column six contains the sensitivity $sensit_i$

7 column seven contains the 1 minus specifity $1mspec_i$.

If CMAT would have a graphical tool, the ROC curve could be drawn by plotting *sensit* against $1mspec$.

The input argument `xini` is a vector specifying initial values of the parameter estimates. The parameters are used in the order of the model effects

(a) the intercept (if `noint` is not specified)

(b) nf parameters for nf fixed effects,

(c) nr parameters for the means of the nr random effects,

(d) $nr * (nr + 1)/2$ parameters for the lower triangle of the variance-covariance matrix of the nr random effects.

A table indicating the parameter usage in the model is printed with the initial output.

One of the more time consuming code parts was the correct computation of type 1 and type 3 estimates. For random effects this is a more tedious programming effort since effects can correspond to more than one parameter.

The additional `pres` option was included for printing the observational statistics (yobs, yhat, residuals etc.). Note, if the `print` options is larger than three, then `pres` is on by default.

1.3 New Functions

New functions are implemented:

cdf23 Bivariate and Trivariate CDF: This is a fast version of the more general function `cdfmv()` to compute the probabilities of bi- and trivariate normal and t distribution.

promep This function implements a new algorithm (Chang, JSS 2003) for generating partially replicated minimal orthogonal main-effect plans.

noharm This function implements the algorithm by Fraser & Mc Donald(1997) and additional ideas by Maydeu (2001) for the factor analysis of dichotomous data. It is used mainly for IRT.

factor This function implements various algorithms of exploratory factor analysis (EFA).

2 New Developments

New developments in CMAT are described in more detail in a corresponding `newdev.pdf` file on this site. Here only some short comments.

2.1 Function `cdf23`

---

v = cdf23("dis",lim,corr<,par>)

Purpose: Computing the probabilities of bi- and trivariate normal and t distribution. The more general function `cdfmv()` is able to compute the integrals of the multivariate normal and t distribution for more than three dimensions. However, for two and three dimensions, the function `cdf23()` can be much faster and permits vectorized calls. Various algorithms by A. Genz (1992, [**?**], 2000, [**?**], Genz & Bretz, 1999, [**?**]) are used.

Input: sopt The first input argument is a string specifying the distribution. Until now, only the multivariate normal and t distributions are supported.

ilim The second argument is a $n_1 \times k$ matrix, where $k = 2$ for bivariate or $k = 3$ for trivariate distribution, specifying the integration limits for n_1 applications. Each column specifies the lower range.

corr The third argument specifies the $n_2 \times l$, where

- $l = 1$ column for bivariate quadrature: specifies the correlation r_{21}, $-1 \leq r_{21} \leq 1$;
- $l = 3$ columns for trivariate quadrature: specifying the correlations r_{21}, r_{31}, r_{32} in that order.

par The (optional) fourth argument may be used to set additional parameters and it's contents depends on the distribution specified in the first argument. For sopt="norm", the content of the parameter vector `par` is:

par[1,...,6] currently not used;

par[7] absolute error tolerance, default is par[7]=0

For sopt="t", the content of the parameter vector `par` is:

par[1] the degrees of freedom

par[2,...,6] currently not used;

par[7] absolute error tolerance, default is par[7]=0

Specifying the absolute error (`par[7]`) is valid only for trivariate quadrature. For bivariate quadrature the error is always smaller than 1.e-14. Note, that for sopt="t" the degrees of freedom must be specified. In this case, the fourth argument `par` must be set, at least as a scalar.

If the number of rows of the second n_1 (limits) and third n_2 (correlations) input argument are both larger than 1, then both must be equal, $n_1 = n_2$.

Output: The returned $max(n_1, n_2)$ vector `v` contains the estimated values of the bi- or trivariate intergrals.

Restrictions: 1. A missing value for v is returned if the input *lim* or *corr* contains string or complex data.

Relationships: cdf(), cdfmv()

Examples: 1. Comparison between `cdf23`, `cdfmv`, and `binorm`:

```
r21 = -.707107;
corr= [      1.0   -.707107 ,
       -.707107        1.0 ];
li1 = [ .5  1.5 ];
lim = [ .5  . ,
        1.5 . ];
par = [ . . . . 0 25000 5.e-5 5.e-5 ];

vol1 = cdf23("norm",li1,r21,par);
vol2 = cdfmv("norm",lim,corr,par);
vol3 = binorm(li1[1],li1[2],r21);
vol = vol1 -> vol2[1] -> vol3;
vol = cname(vol,[ "CDF23" "CDFNM" "BINORM" ]);
print "Volume:", vol;


Volume:
   |      CDF23      CDFNM     BINORM
-------------------------------------
 1 |   0.000397   0.000397   0.000397
```

Note, the algorithm in `binorm` assumes positive integration limits, however, obtains the best precision of bivariate normal probabilities. The algorithms in `cdf23` are faster and more precise for two and three dimensions, but the algorithms for `cdfmv` can be applied also for larger dimesnions than three.

2. Patefields Problems with $h, k > 0$:

```
/* Patefield's Problems */
biva = [ 1.0   0.0    0.5     ,
         0.0   1.0    0.5     ,
         1.0   3.0    0.5     ,
```

```
          3.0   3.393  0.99    ,
          2.0   6.0    0.85385 ,
          2.5   7.5    0.85385 ,
          2.0   0.0    0.9     ];
 nr = nrow(biva);
 vol = cons(nr,3);
 vol = cname(vol,[ "CDF23" "CDFNM" "BINORM" ]);

 par = [ . . . . 0 25000 5.e-5 5.e-5 ];
 vol[,1] = cdf23("norm",biva[,1:2],biva[,3],par);

 cor = ide(2);
 lim  = cons(2,2,.);
 for (i = 1; i <= nr; i++) {
   li = biva[i,1:2]; ci = biva[i,3];
   lim[,1] = li'; cor[1,2] = cor[2,1] = ci;
   vol[i,2] = cdfmv("norm",lim,cor,par);
   vol[i,3] = binorm(li[1],li[2],ci);
 }
 vol = cname(vol,[ "CDF23" "CDFNM" "BINORM" ]);
 print "Volume:", vol;
```

```
Volume:
   |     CDF23     CDFNM    BINORM
---------------------------------
 1 |   0.12740   0.12740   0.12740
 2 |   0.12740   0.12740   0.12740
 3 |   0.00104   0.00104   0.00104
 4 |   0.00035   0.00035   0.00035
 5 |    1e-009    1e-009    1e-009
 6 |    3e-014    3e-014    3e-014
 7 |   0.02275   0.02275   0.02275
 8 |   0.00000    2e-016   0.00000
```

2.2 Function `factor`

< gof,est,resi,cov> = factor(data,optn<,wgt<,init<,prior>>>)

Purpose: The `factor` function implements a number of exploratory factor analysis methods which can be used for ULS, ML, GLS, WLS, and DWLS estimation. It incorporates robust (nonnormal, S-B) goodness of fit indices and ASEs. This function is not ready yet and will be documented in more detail in the July 2003 newsletter.

2.3 Function `noharm`

< gof,est,resi,cov > = noharm(data,optn<,guess<,init>>)

Purpose: The `noharm` function performs factor analysis for dichotomous variables assuming that the variables arise from a multinormal distribution that has been categorized (Maydeu, 2001). The term NOHARM refers to *normal ogive harmonic analysis robust model*. A similar algorithm was developed by Fraser & McDonald (1988). Compared to the NOHARM program this function uses by default an exact bivariate quadrature algorithm which is also used by the `cdf23` function and incorporates some additional features like robust ASEs and robust goodness-of-fit evaluation.

Input: The function has a maximum of four input arguments, two of them are optional:

data The input data set can be either a $N \times n$ matrix of binary raw data or a symmetric $n \times n$ of scalar products $X^T X$. Even $n \times n$ covariance or correlation matrices can be used if means and standard deviations are attached to the those matrices in additional rows. Note, that raw data must be specified to compute the robust ASE's and GOF values.

optn : The option argument is specified in form of a two column matrix where the first column defines the option as string value (in quotes) and the second column can be used for a numeric or string specification of the option. See table below for content.

guess (optional) Unused at this time.

init (optional) Specifies initial values for **F** and **P** matrices.

Options Matrix Argument: The option argument is specified in form of a two column matrix:

Option Name	Second Column	Meaning
"alpha"	real	the significance level for confidence intervals; default is $\alpha = 0.05$
"ana"	string	data type for analysis
	"cor"	correlations are analyzed
	"cov"	covariances are analyzed
	"ucor"	uncorrected correlations are analyzed
	"ucov"	uncorrected covariances are analyzed
"asc"	string	type of asymptotic covariances
	"bia"	biased: Browne's (1984) formula (3.4)
	"unb"	unbiased: Browne's (1984) formula (3.8)
	"cor"	Browne & Shapiro (1986) formula (3.2)
"cl"	"none"	do not compute confidence intervals
	"wald"	compute Wald confidence intervals (default)
	"plik"	compute profile likelihood intervals
"comp"		perform principal component analysis
		use only for exploratory FACTOR model
"data"	string	type of input data
	"raw"	$N \times n$ raw data
	"cor"	symmetric correlation matrix
	"cov"	symmetric covariance matrix
	"scp"	scalar product matrix $X^T X$
"freq"	int	column number of FREQ variable when raw data input
"frot"	string	rotation method for exploratory FACTOR model
	"non"	no rotation
	"qua"	quartimax rotation
	"var"	varimax rotation
	"equ"	equamax rotation
	"par"	parsimax rotation
	"pri"	principal component rotation
"fnorm"		Kaiser's factor pattern normalization
		use only for exploratory FACTOR model
"maxfu"	int	maximum nuber function calls for optimization
"maxit"	int	maximum number iterations for optimization
"nfac"	real	number factors for exploratory FACTOR model
"nobs"	real	number observations when COV or CORR data input
		only for single sample analysis
"nopr"		perform no printed output
"nomis"		skip observations with missing values in raw data
"prin"	int	medium amount of printed output
"pall"		large amount of printed output
"psho"		short amount of printed output
"psum"		summary amount of printed output
"sing"	real	singularity threshold (default: 1.e-8)
"seed"	real	seed value for random generator

Option Name	Second Column	Meaning
"tech"	string	optimization technique
	"tr"	trust region
	"lm"	Levenberg-Marquardt
	"cg"	conjugate gradient
	"nrl"	Newton-Raphson line search
	"nrr"	Newton-Raphson ridged
	"dqn"	dual Quasi Newton
	"nms"	Nelder-Mead simplex
	"qpn"	quadratic penalty algorithm (NLC)
	"sqp"	sequential QP algorithm (NLC)
	"cb"	Cobyla (NLC)
"vers"	int	version of algorithm: =1: poynomial appr.; =2: quadrature
"wridge"	real	ridge factor for weight matrix (GLS, WLS, DWLS)
"wpen"	real	penalty weight for WLS and DWLS estimation
"weight"	int	column number of weight variable

Output: The `noharm` function offers a maximum of four return arguments:

gof the first return argument is a vector of goodness-of-fit indices

1. value of the ULS fit function
2. root-mean-squared (RMS) value
3. Tanaka index
4. $NOBS * F_{ULS}$
5. not used
6. Satorra-Bentler mean corrected χ^2
7. probability of S-B mean corrected χ^2
8. *df* of S-B mean corrected χ^2
9. Satorra-Bentler variance corrected χ^2
10. probability of S-B variance corrected χ^2
11. *df* of S-B variance corrected χ^2
12. computer time in seconds

est Depending on the fact whether (robust) approximate standard errors are computed, this return argument has two different forms:

- no ASE's computed: the second return argument is matrix containing the matrix **L** in the first $nfact$ columns and the unique varoiance in the last column.
- robust ASE's are computed: the first column contains all optimal estimates of **L**, **P**, and **U**. The next four columns contain the approximate standard errors and t values, the lower and upper confidence limits.

resi Depending on the fact whether (robust) approximate standard errors are computed, this return argument has two different forms:

- no ASE's computed: the third return argument contains the symmetric $n\times$ residual matrix
- robust ASE's are computed: two residual matrices, the common residuals and the standardized residual matrices stacked vertically.

covm The fourth return argument contains the symmetric $npar \times npar$ (robust) covariance matrix of parameter estimates.

Restrictions: 1. The input data must be binary, i.e. (0,1) data and cannot contain any missing data.

2.

Relationships: factor(), sem()

Examples: 1. Analysis of the LSAT6 Data: Version with bivariate quadrature

```
  options NOECHO;
  lsat6 = [
#include "..\\tdata\\lsat6.dat"
           ];
  options ECHO;

  lsat6 = shape(lsat6,.,5);
  name = [ "V1" : "V5" ];
  lsat6 = cname(lsat6,name);


  optn = [ "data"    "raw" ,
           "nfac"        1 ,
           "cl"     "wald" ,
           "frot"   "prom" ,
           "prin"        3 ];
  < gof,parm,resi,covm > = noharm(lsat6,optn);


            ****************************************
            Fitting a (multidimensional) Normal Ogive
               by Harmonic Analysis - Robust Method
                   (Using Bivariate Quadrature)
                  (Exploratory Factor Analysis)
            ****************************************

    Number of Items . . . . . .          5
    Number of Dimensions. . . . .        1
    Number of Subjects. . . . . .     1000


                  Sample Product-Moment Matrix

              V1         V2         V3         V4         V5
    V1   0.9240000
    V2   0.6640000  0.7090000
    V3   0.5240000  0.4180000  0.5530000
    V4   0.7100000  0.5530000  0.4450000  0.7630000
    V5   0.8060000  0.6300000  0.4900000  0.6780000  0.8700000


                    Item Covariance Matrix

              V1         V2         V3         V4         V5
    V1   0.0702240
    V2   0.0088840  0.2063190
    V3   0.0130280  0.0259230  0.2471910
```

```
    V4   0.0049880  0.0120330  0.0230610  0.1808310
    V5   0.0021200  0.0131700  0.0088900  0.0141900  0.1131000

Initial Constants
-----------------

  1 :      -1.602    -0.6154     -0.149    -0.8005     -1.259

                  Trust Region Optimization
                  Without Parameter Scaling
                   User Specified Gradient
         Hessian Computed by Finite Differences (dense)
                  (Using Analytic Gradient)


   Iteration Start:
   N. Variables              5
   Criterion       0.000227690        Max Grad Entry  0.001052410
   TR Radius       1.000000000

  Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius

     1    0    0   0  9.5e-005 1.3e-004  6e-005 0.00000  1.00000
     2    0    0   0  9.5e-005 5.7e-007  8e-007 0.00000  0.19658

   Successful Termination After     2 Iterations
   ABSGCONV convergence criterion satisfied.
   Criterion       9.4660e-005        Max Grad Entry  7.5537e-007
   Ridge (lambda)  0.000000000        TR Radius       0.196581224
   Act.dF/Pred.dF  0.991503127
   N. Function Calls         0        N. Hessian Calls          4
   Preproces. Time           0        Time for Method           0
   Effective Time            0

Threshold Values
----------------

  1 :      -1.433    -0.5505    -0.1332     -0.716     -1.126

Unique Variances
----------------

  1 :      0.8377     0.8509     0.7742     0.8608     0.8815

Unrotated Factor Loadings
-------------------------
```

```
 1 :      0.4029     0.3861     0.4752     0.3731     0.3443
```

Covariance Matrix of Threshold Estimates

	T_1	T_2	T_3	T_4	T_5
T_1	0.0034345				
T_2	1.81e-004	0.0017552			
T_3	2.30e-004	1.91e-004	0.0015810		
T_4	1.13e-004	1.14e-004	1.89e-004	0.0018971	
T_5	7.01e-005	1.82e-004	1.06e-004	2.17e-004	0.0025273

Covariance Matrix of Factor Model Estimates

	L_1	L_2	L_3	L_4	L_5
L_1	0.0124289				
L_2	-7.02e-005	0.0073387			
L_3	-0.0017358	-0.0021169	0.0086842		
L_4	-2.11e-004	-8.84e-004	-0.0017486	0.0075326	
L_5	-3.20e-004	-3.39e-004	-0.0014112	2.25e-004	0.0091070

```
Fit criterion . . . . . .     0.0001
Nobs * OptCrit. . . . . .     0.0947
Root mean square of residuals .     0.0031
Tanaka index of goodness of fit .     0.9988
Satorra-Bentler Mean (df= 5). .  5.0235 Prob>chi**2 = 0.4130
Sat.-B. M-V (df= 4.323) . . . .  4.3433 Prob>chi**2 = 0.4069
```

Estimates and Asymptotic Standard Errors

Parameter	Estimate	AStdErr	T Value	Low Wald	Upp Wald
T_1	-1.43250271	0.0586047	-24.4434609	-1.5473659	-1.3176395
T_2	-0.55046572	0.0418946	-13.1393052	-0.6325776	-0.4683538
T_3	-0.13324453	0.0397614	-3.35110364	-0.2111754	-0.0553136
T_4	-0.71598601	0.0435556	-16.4384427	-0.8013534	-0.6306186
T_5	-1.12639118	0.0502723	-22.4058231	-1.2249230	-1.0278594
L_1	0.40290050	0.1114852	3.61393783	0.1843936	0.6214074
L_2	0.38609060	0.0856665	4.50690387	0.2181874	0.5539938
L_3	0.47518498	0.0931893	5.09913683	0.2925373	0.6578326
L_4	0.37310262	0.0867905	4.29888539	0.2029963	0.5432090
L_5	0.34430156	0.0954308	3.60786758	0.1572607	0.5313424

LORD'S Parametrization - Unidimensional Case

```
VECTOR A : Discrimination parameters
------------------------------------

  1 :      0.4402     0.4185     0.5401     0.4021     0.3667

VECTOR B : Difficulty parameters
--------------------------------

  1 :       3.555      1.426     0.2804      1.919      3.272
```

2. Analysis of the LSAT6 Data: Version with polynomial approximation (like original NOHARM program)

```
optn = [ "data"    "raw" ,
         "nfac"        1 ,
         "vers"        1 ,
         "cl"     "wald" ,
         "frot"   "prom" ,
         "prin"        3 ];
< gof,parm,resi,covm > = noharm(lsat6,optn);
```

Since the results are almost identical we skip most of the printed output:

```
            ****************************************
            Fitting a (multidimensional) Normal Ogive
               by Harmonic Analysis - Robust Method
               (Original Polynomial Approximation)
                  (Exploratory Factor Analysis)
            ****************************************

   Number of Items . . . . . . . . . . . . . . . .         5
   Number of Dimensions. . . . . . . . . . . . . .         1
   Number of Subjects. . . . . . . . . . . . . . .      1000


                  Trust Region Optimization
                  Without Parameter Scaling
                   User Specified Gradient
         Hessian Computed by Finite Differences (dense)
                  (Using Analytic Gradient)


   Iteration Start:
   N. Variables              5
   Criterion       0.000227018        Max Grad Entry  0.001046349
   TR Radius       1.000000000

  Iter rest nfun act   optcrit  difcrit maxgrad  lambda   radius

     1    0    0   0  9.5e-005 1.3e-004  6e-005 0.00000  1.00000
```

```
  2    0    0   0  9.5e-005 5.7e-007  8e-007 0.00000  0.19761

   Successful Termination After     2 Iterations
   ABSGCONV convergence criterion satisfied.
   Criterion       9.4656e-005        Max Grad Entry  7.9426e-007
   Ridge (lambda)  0.000000000        TR Radius       0.197609696
   Act.dF/Pred.dF  0.992954259
   N. Function Calls         0        N. Hessian Calls          4
   Preproces. Time           0        Time for Method           0
   Effective Time            0
```

Threshold Values

1 :	-1.433	-0.5505	-0.1332	-0.716	-1.126

Unique Variances

1 :	0.8376	0.8509	0.7742	0.8608	0.8815

Unrotated Factor Loadings

1 :	0.403	0.3862	0.4752	0.3732	0.3443

Covariance Matrix of Threshold Estimates

	T_1	T_2	T_3	T_4	T_5
T_1	0.0034345				
T_2	1.81e-004	0.0017552			
T_3	2.30e-004	1.91e-004	0.0015810		
T_4	1.13e-004	1.14e-004	1.89e-004	0.0018971	
T_5	7.01e-005	1.82e-004	1.06e-004	2.17e-004	0.0025273

Covariance Matrix of Factor Model Estimates

	L_1	L_2	L_3	L_4	L_5
L_1	0.0124264				
L_2	-7.08e-005	0.0073381			
L_3	-0.0017348	-0.0021165	0.0086814		
L_4	-2.11e-004	-8.85e-004	-0.0017478	0.0075313	
L_5	-3.20e-004	-3.40e-004	-0.0014099	2.24e-004	0.0091046

```
   Fit criterion . . . . . .     0.0001
   Nobs * OptCrit. . . . . .     0.0947
   Root mean square of residuals .     0.0031
```

```
   Tanaka index of goodness of fit . . . . . .     0.9988
   Satorra-Bentler Mean (df= 5). .  5.0234 Prob>chi**2 = 0.4130
   Sat.-B. M-V (df= 4.323) . . . .  4.3433 Prob>chi**2 = 0.4069


            ****************************************
            Estimates and Asymptotic Standard Errors
            ****************************************

Parameter    Estimate    AStdErr     T Value   Low Wald   Upp Wald
      T_1 -1.43250271  0.0586047 -24.4434609 -1.5473659 -1.3176395
      T_2 -0.55046572  0.0418946 -13.1393052 -0.6325776 -0.4683538
      T_3 -0.13324453  0.0397614 -3.35110364 -0.2111754 -0.0553136
      T_4 -0.71598601  0.0435556 -16.4384427 -0.8013534 -0.6306186
      T_5 -1.12639118  0.0502723 -22.4058231 -1.2249230 -1.0278594
      L_1  0.40296690  0.1114737  3.61490404  0.1844824  0.6214514
      L_2  0.38617440  0.0856625  4.50809139  0.2182790  0.5540698
      L_3  0.47519424  0.0931743  5.10005762  0.2925760  0.6578125
      L_4  0.37315357  0.0867828  4.29985526  0.2030624  0.5432448
      L_5  0.34427663  0.0954182  3.60808219  0.1572604  0.5312928
```

2.4 Function promep

< plan,planb > = promep(levs<,optn>)

Purpose: Function for generating partially replicated minimal orthogonal main-effect plans (Chang, JSS 2003).

Input: The first input argument must be a vector with three integers $s_1 >= s_2 >= s_3$ specifying the number of levels of the three largest factors. An optional second input argument can be used to specify the amount of printed output.

Output: The function can have no more than two returns:

1. The first return `plan` is a $nr \times nf$ minimal PROMEP, where nf is the number of factors.
2. Only if $s_2 = s_3 = s_1 - 1$ an additional three factor minimal PROMEP is returned as `planb`. If this relationship between level numbers is not satisfied the second return `planb` is a missing value.

Restrictions:

Relationships:

Examples: 1. Examples shown in Chang (2003):

```
/* Example 1 */
lev1 = [ 5 4 4 ];
< plan1,plan1a > = promep(lev1);
print plan1, plan1a;


Min. PROMEP 5 x 4 x 4 // 5 * 5 with 2 Duplicate Points
*****************************************************
```

Dense Matrix (25 by 3)

	1	2	3
1	0	0	0
2	0	0	0
3	0	1	2
4	0	2	1
5	0	3	3
6	1	0	0
7	1	0	0
8	1	1	3
9	1	2	2
10	1	3	1
11	2	0	1
12	2	0	2
13	2	1	0
14	2	2	3
15	2	3	0
16	3	0	3
17	3	0	1
18	3	1	0
19	3	2	0
20	3	3	2
21	4	0	2
22	4	0	3
23	4	1	1
24	4	2	0
25	4	3	0

Minimal PROMEP with 2 Maximum Duplicate Points

Min. PROMEP 5 x 4 x 4^4 // 5 * 5 with 1 Dupl. Points
****

Dense Matrix (25 by 6)

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	1	1	1	1	1
4	0	2	2	2	2	2
5	0	3	3	3	3	3
6	1	0	0	1	2	3
7	1	0	1	2	3	0
8	1	1	2	3	0	0
9	1	2	3	0	0	1
10	1	3	0	0	1	2

```
   11 |       2      0      1      3      0      2
   12 |       2      0      2      0      1      3
   13 |       2      1      3      0      2      0
   14 |       2      2      0      1      3      0
   15 |       2      3      0      2      0      1
   16 |       3      0      2      0      3      1
   17 |       3      0      3      1      0      2
   18 |       3      1      0      2      0      3
   19 |       3      2      0      3      1      0
   20 |       3      3      1      0      2      0
   21 |       4      0      0      3      2      1
   22 |       4      0      3      2      1      0
   23 |       4      1      0      0      3      2
   24 |       4      2      1      0      0      3
   25 |       4      3      2      1      0      0
```

```
   /* Example 2 */
   lev2 = [ 7 4 4 ];
   plan2 = promep(lev2);
   print plan2;
```

```
Min. PROMEP 7 x 4 x 4^3 // 8 * 4 with 4 Dupl. Points
****************************************************

         Dense Matrix (32 by 5)

     |       1      2      3      4      5
    ------------------------------------
    1 |       0      0      0      0      0
    2 |       0      1      1      1      1
    3 |       0      2      2      2      2
    4 |       0      3      3      3      3
    5 |       1      0      1      2      3
    6 |       1      1      0      3      2
    7 |       1      2      3      0      1
    8 |       1      3      2      1      0
    9 |       2      0      2      3      1
   10 |       2      1      3      2      0
   11 |       2      2      0      1      3
   12 |       2      3      1      0      2
   13 |       3      0      3      1      2
   14 |       3      0      3      1      2
   15 |       3      1      2      0      3
   16 |       3      1      2      0      3
   17 |       3      2      1      3      0
   18 |       3      2      1      3      0
   19 |       3      3      0      2      1
   20 |       3      3      0      2      1
   21 |       4      0      0      0      0
```

```
   22 |       4      1      1      1      1
   23 |       4      2      2      2      2
   24 |       4      3      3      3      3
   25 |       5      0      1      2      3
   26 |       5      1      0      3      2
   27 |       5      2      3      0      1
   28 |       5      3      2      1      0
   29 |       6      0      2      3      1
   30 |       6      1      3      2      0
   31 |       6      2      0      1      3
   32 |       6      3      1      0      2

          Minimal PROMEP with 4 Maximum Duplicate Points


   /* Example 3 */
   lev3 = [ 4 3 3 ];
   plan3 = promep(lev3);
   print plan3;


Min. PROMEP 4 x 3 x 3^3 // 4 * 4 with 1 Dupl. Points
**************************************************

          Dense Matrix (16 by 5)

     |       1      2      3      4      5
   ---------------------------------------
    1 |       0      0      0      0      0
    2 |       0      0      0      0      0
    3 |       0      1      1      1      1
    4 |       0      2      2      2      2
    5 |       1      0      1      2      0
    6 |       1      0      2      1      0
    7 |       1      1      0      0      2
    8 |       1      2      0      0      1
    9 |       2      0      1      0      2
   10 |       2      0      2      0      1
   11 |       2      1      0      2      0
   12 |       2      2      0      1      0
   13 |       3      0      0      1      2
   14 |       3      0      0      2      1
   15 |       3      1      2      0      0
   16 |       3      2      1      0      0


   /* Example 4 */
   lev4 = [ 11 7 3 ];
   plan4 = promep(lev4);
   print plan4;
```

Min. PROMEP 11 x 7 x 3^3 // 12 * 8 with 19 Dupl. Points
****

Dense Matrix (96 by 5)

	1	2	3	4	5
1	0	0	0	0	0
2	0	1	1	1	1
3	0	2	2	2	2
4	0	3	0	0	0
5	0	3	0	0	0
6	0	4	0	0	0
7	0	5	1	1	1
8	0	6	2	2	2
9	1	0	1	2	0
10	1	1	0	0	2
11	1	2	0	0	1
12	1	3	2	1	0
13	1	3	2	1	0
14	1	4	1	2	0
15	1	5	0	0	2
16	1	6	0	0	1
17	2	0	2	0	1
18	2	1	0	2	0
19	2	2	0	1	0
20	2	3	1	0	2
21	2	3	1	0	2
22	2	4	2	0	1
23	2	5	0	2	0
24	2	6	0	1	0
25	3	0	0	1	2
26	3	0	0	1	2
27	3	1	2	0	0
28	3	1	2	0	0
29	3	2	1	0	0
30	3	2	1	0	0
31	3	3	0	2	1
32	3	3	0	2	1
33	3	3	0	2	1
34	3	3	0	2	1
35	3	4	0	1	2
36	3	4	0	1	2
37	3	5	2	0	0
38	3	5	2	0	0
39	3	6	1	0	0
40	3	6	1	0	0
41	4	0	0	0	0
42	4	1	1	1	1
43	4	2	2	2	2

44	4	3	0	0	0
45	4	3	0	0	0
46	4	4	0	0	0
47	4	5	1	1	1
48	4	6	2	2	2
49	5	0	1	2	0
50	5	1	0	0	2

	1	2	3	4	5
51	5	2	0	0	1
52	5	3	2	1	0
53	5	3	2	1	0
54	5	4	1	2	0
55	5	5	0	0	2
56	5	6	0	0	1
57	6	0	2	0	1
58	6	1	0	2	0
59	6	2	0	1	0
60	6	3	1	0	2
61	6	3	1	0	2
62	6	4	2	0	1
63	6	5	0	2	0
64	6	6	0	1	0
65	7	0	0	1	2
66	7	1	2	0	0
67	7	2	1	0	0
68	7	3	0	2	1
69	7	3	0	2	1
70	7	4	0	1	2
71	7	5	2	0	0
72	7	6	1	0	0
73	8	0	0	0	0
74	8	1	1	1	1
75	8	2	2	2	2
76	8	3	0	0	0
77	8	3	0	0	0
78	8	4	0	0	0
79	8	5	1	1	1
80	8	6	2	2	2
81	9	0	1	2	0
82	9	1	0	0	2
83	9	2	0	0	1
84	9	3	2	1	0
85	9	3	2	1	0
86	9	4	1	2	0
87	9	5	0	0	2
88	9	6	0	0	1
89	10	0	2	0	1
90	10	1	0	2	0

```
   91 |      10      2      0      1      0
   92 |      10      3      1      0      2
   93 |      10      3      1      0      2
   94 |      10      4      2      0      1
   95 |      10      5      0      2      0
   96 |      10      6      0      1      0

          Minimal PROMEP with 19 Maximum Duplicate Points


   /* Example 5 */
   lev5 = [ 11 6 6 ];
   plan5 = promep(lev5);
   print plan5;


Min. PROMEP 11 x 6 x 6 // 12 * 6 with 6 Duplicate Points
************************************************************

   Dense Matrix (72 by 3)

      |       1      2      3
    -------------------------
    1 |       0      0      0
    2 |       0      1      1
    3 |       0      2      2
    4 |       0      3      3
    5 |       0      4      4
    6 |       0      5      5
    7 |       1      0      1
    8 |       1      1      2
    9 |       1      2      3
   10 |       1      3      4
   11 |       1      4      5
   12 |       1      5      0
   13 |       2      0      2
   14 |       2      1      3
   15 |       2      2      4
   16 |       2      3      5
   17 |       2      4      0
   18 |       2      5      1
   19 |       3      0      3
   20 |       3      1      4
   21 |       3      2      5
   22 |       3      3      0
   23 |       3      4      1
   24 |       3      5      2
   25 |       4      0      4
   26 |       4      1      5
   27 |       4      2      0
   28 |       4      3      1
```

29	4	4	2
30	4	5	3
31	5	0	5
32	5	0	5
33	5	1	0
34	5	1	0
35	5	2	1
36	5	2	1
37	5	3	2
38	5	3	2
39	5	4	3
40	5	4	3
41	5	5	4
42	5	5	4
43	6	0	0
44	6	1	1
45	6	2	2
46	6	3	3
47	6	4	4
48	6	5	5
49	7	0	1
50	7	1	2

	1	2	3
51	7	2	3
52	7	3	4
53	7	4	5
54	7	5	0
55	8	0	2
56	8	1	3
57	8	2	4
58	8	3	5
59	8	4	0
60	8	5	1
61	9	0	3
62	9	1	4
63	9	2	5
64	9	3	0
65	9	4	1
66	9	5	2
67	10	0	4
68	10	1	5
69	10	2	0
70	10	3	1
71	10	4	2
72	10	5	3

Minimal PROMEP with 6 Maximum Duplicate Points

3 Illustration: Estimating the Area under a ROC Curve

3.1 Use of `glim()` Function

After correcting some older code and including some new features of the observationwise statistics in the `glim` function we are now able to compute better estimates of the area under the ROC curve. Some SAS Macros by DeLong et.al. (1988) which are available at the website `http://ftp.sas.com/techsup/download/stat` illustrate the computation of the area under a ROC curve and the comparison of the area of different curves. For the computation of the predicted scores $\hat{y}_i$ the SAS PROC LOGISTIC is being used. Now, the same results can also be obtained by the `glim` function in CMAT. This is illustrated by the following small example taken from one of the SAS macros:

```
print "EXAMPLE 1 -- Single Predictor";
print "Plot with area under the ROC curve in title";

data = [  0 14 25,
          0 20 35,
          0 19 45,
          7 18 55,
          6 12 65,
         17 17 75 ];


cnam = [ "disease" "n" "age" ];
data = cname(data,cnam);
print " Nobs=", nrow(data),
      " Nvar=", ncol(data);


model = " 1/2 = 3 ";
optn = [ "print"           3 ,
         "pres"              ,
         "link"      "logit" ,
         "dist"      "binom" ,
         "tech"     "dbldog" ,
         "maxit"         500 ,
         "maxfu"        1000 ];
< gof,parm,ase,cli,cov > = glim(data,model,optn);
```

Due to bugs in `glim` function this example cannot be run in the earlier distributed versions of CMAT. The response is an effects/trial variable, that means the effect numbers (in column 1 of data) count the occurences of y=1 at the trial numbers (column 2 in data). The number of failures, i.e. the number of nonevents with y=0, is given by trial-minus-effect, i.e. as column 2 - column 1 of the data. Such a data set could also be written in a different form:

- the event and trial variables could be replaced by a binary response with value 1 for event and 0 for nonevent,
- and a frequency variable which contains event frequency for response value $y = 1$ and the trial minus event frequency for response value $y = 0$.

****

Model Information
****

Number Valid Observations 6
Events/Trial Resp [1]/[2]
N Independend Variables 1
Error Distribution BINOMIAL
Link Function LOGIT

Trial Variable Column 2
Significance Level: 0.0500000
Design Coding: Full-Rank
Hessian Matx. for Optimization
Hessian Matx. for Covariance M
No Variable Selection Process

****
Model Effects
****

Intercept + X3

****
Simple Statistics
****

Column	Nobs	Mean	Std Dev	Skewness	Kurtosis
Y[1]	6	0.3148148	0.4016427	1.0873358	0.5039404
X[3]	6	50.000000	18.708287	0.0000000	-1.2000000

****
Parameter Information
****

Parameter	Meaning
1	Intercept
2	X3

****
Events/Trials Response Variable
****

Value	Nobs	Proportion
Event	30	30.000000
NonEvent	70	70.000000

****

```
                    Goodness of Model Fit
                    **********************

Log Likelihood     -24.234022      Degrees of Freedom          4
Deviance            7.7755662      Pearson ChiSquare   6.6020397
SSE                 8.2937941      MSE = SSE/nobs      1.3822990

AIC (Intercept)     124.17286      AIC (All Param.)    52.468044
SBC (Intercept)     126.77803      SBC (All Param.)    57.678384
-2logL (Intercept)  122.17286      -2logL (All Param.) 48.468044
-2logL (ChiSqu.)    73.704817      Pvalue (df=      1) 0.0000000
Score ChiSqu. Test  55.327354      Pvalue (df=      1) 0.0000000


          ****************************************
          Analysis of Effects and Parameter Estimates
          ****************************************

  Parameter        DF  Estimate Std_Error WaldChiSq Pr>ChiSq

  Intercept         1 -12.50164  2.555528 23.931658 0.000001
  X3                1  0.206587  0.042755 23.347511 0.000001
  Scale             0  1.000000  0.000000     .        .

      Covariance Matrix and ASE Based on Hessian Matrix


               ******************************
               Confidence Limits of Parameters
               ******************************

     Parameter          Estimate   LowWaldCL   UppWaldCL

     Intercept        -12.501643 -17.5103871 -7.49289973
     X3                0.2065871  0.12278963  0.29038463

      Wald Confidence Intervals Based on Hessian Matrix


            ***********************************
            Odds Ratio and Standardized Estimates
            ***********************************

             Parameter        OddsRatio   StndEst

             X3               1.2294749  2.130826
```

Earlier versions of `glim` did not compute the correct numbers of concordant and discordant pairs which could be used for the computation of the area under the ROC curve.

```
                  ******************************
```

```
            Evaluation of Training Data Fit
            *******************************

Index                              Value       StdErr
Absolute Classification Error         13            .
Concordant Pairs               92.61904762           .
Discordant Pairs               2.000000000           .
Tied Pairs                     5.380952381           .
Classification Accuracy        87.00000000           .
Goodman-Kruskal Gamma          0.944517834  0.032751331
Kendall Tau_b                  0.687698721  0.079996134
Stuart Tau_c                   0.572000000  0.086546173
Somers D C|R                   0.680952381  0.084157569
```

The area is computed by two different algorithms, one similar to the one used in PROC LOGISTIC (discrete) and one (continuous) used by Liu & Wu (2003). The latter is more precise but computationally more expensive and assumes the predicted values (scores) incore. The approximation used in PROC LOGISTIC can be used for data with very large observation numbers and does not need the predicted values incore:

```
                Classification Table
                --------------------

                        |    Predicted
                Observed |    NoEvt   Event
                ---------|-----------------
                   NoEvt |       64       6
                   Event |        7      23

          Area under the ROC Curve (Disc.) = 0.953095
          Area under the ROC Curve (Cont.) = 0.953095


************************************************************
    Nobs Yobs       Ypred     LowerCL     UpperCL     Diag(H)
************************************************************

       1    0  6.506e-004  3.403e-005  0.01230171  0.02063942
       2    0  0.00511172  5.989e-004  0.04219521  0.12225030
       3    0  0.03896967  0.01001221  0.13984747  0.35720388
       4    7  0.24243952  0.12544299  0.41657823  0.55424703
       5    6  0.71636878  0.51309868  0.85822606  0.48496613
       6   17  0.95222922  0.81451293  0.98906918  0.46069324

         SSE=8.29379 MSE=1.3823 Misclassification=13

      Pearson ChiSquare=6.60204 Deviance=7.77557 (df=4)
```

3.2 Use of `glmixd()` Function

The logistic approach was extended by Liu & Wu (2003) to the logistic mixed model, i.e. data which have a repeated measures design. Their data set contains a maximum of 12 measurements (every 4-week period, called wave, for 48 weeks) for each patient indicated by an ID variable. Note, that the replications are not independent and must be considered correlated. Of course, the data set contains a large number of missing values. Here some small part of their data:

```
  options ls=68 ps=100;
  options NOECHO;
  anal = [
#include "..\\tdata\\anal.dat"
           ];
  options ECHO;
  anal = shape(anal,.,5);

  cnam = [ "ID" "LOWESTCD" "WAVE" "PC" "bi_mems1" ];
  anal = cname(anal,cnam);
  print " Nobs=", nrow(anal),
        " Nvar=", ncol(anal);
```

with the data for the first two patients:

```
1001 1130 1 0.97 1
1001 1130 2 0.86 0
1001 1130 3 1 1
1001 1130 4 0.82 0
1001 1130 5 0.44 0
1001 1130 6 0.75 0
1001 1130 7 0.49 0
1001 1130 8 0.69 0
1001 1130 9 0.77 0
1001 1130 10 0.9 0
1001 1130 11 1 0
1001 1130 12 0.82 0
1002 9 1 1 1
1002 9 2 0.97 1
1002 9 3 . 1
1002 9 4 1 1
1002 9 5 0.95 0
1002 9 6 1 1
1002 9 7 0.865 1
1002 9 8 0.925 1
1002 9 9 0.99 0
1002 9 10 0.945 1
1002 9 11 0.88 1
1002 9 12 0.72 0
.........................
```

We have here a slight problem since the `glmixd` function in CMAT in this version is not able to model correlated repeated measurements. But it can treat the repeated measurements as independent and obtain results which

should be close to those by Liu & Wu (2003). Not much software is avaliable which can treat the repeated measurements as correlated. Liu & Wu use the SAS macro `glimmix` and include a small discussion of other related software in their paper. For example, the SAS PROC NLMIXED or Don Hedeker's MIXNO or MIXOR (on which our `glmixd` is based) cannot take the correlated measurements into account. It is not entirely clear whether the `nlme` program by Pinheiro & Bates can solve the problem correctly. Anyway, we want to illustrate the approach using the `glmixd` function even though that it's result may not be entirely correct. In addition we want to show the new results for the Type 1 and Type 3 analyses.

```
model = " 5 = 2 3 4 ";
random = 2 ;    /* random points to effects */
class  = 5 ;    /* class points to columns */
optn = [ "print"           3 ,
         "popt"            3 ,
         "type1"             ,
         "type3"             ,
         "pres"              ,
         "link"      "logit" ,
         "efdis"    "normal" ,
         "yscale"        "n" ,
         "idvar"           1 ,
         "nquad"          20 ,
         "tech"     "duquan" ,
         "maxit"         500 ,
         "maxfu"        1000 ];
< gof,parm,ase,cli,cov > = glmixd(anal,model,optn,class,random);
```

```
                  *****************
                  Model Information
                  *****************

            Number Valid Observations 1226
            Response Variable          Y[5]
            N Independend Variables       3
            Error Distribution       NORMAL
            Link Function             LOGIT
            Effect Distribution      NORMAL
            Nominal Response with 2 Categ.
            Number Random Effects         1
            Number Fixed Effects          3
            Quad Points per Dimension    20
            Subject Variable Column       1
            Significance Level:  0.0500000
            Design Coding:       Full-Rank


                   *************
                   Model Effects
                   *************

              Intercept + X2 + X3 + X4
```

The following output of the class level information is skipped for space reasons.

```
                      *****************
                      Simple Statistics
                      *****************

 Column  Nobs        Mean     Std Dev    Skewness    Kurtosis
 X[2]    1226   150.19739   181.35627   2.2784429   7.5932885
 X[3]    1214   6.3080725   3.5262274   0.0687174  -1.2475950
 X[4]     987   0.8614565   0.2044903  -2.4453487   8.1100696


                     *********************
                     Parameter Information
                     *********************

  Number  | Coefficient  | Effect
  ------------------------------------------------------------
        1 | Alph_1_1     | Fixed Intercept
        2 | Alph_2_1     | X2
        3 | Alph_3_1     | X4
        4 | Mean_1_1     | X3
        5 | COV_1_1      | X3                X3
  ------------------------------------------------------------
```

A sever bug for the computation of starting values was corrected:

```
                      ***************
                      Starting Values
                      ***************

 Covariates
 ----------

    1|     -0.2333   0.0002309      -1.506

 Means
 -----

   1 :     0.05056

 Homogeneous COV Parameter Matrix
 --------------------------------

    1|      0.5736
```

The DUQUAN (or DBLDOG) algorithm is saver in the presence of bad initial values than an algorithm which would use the second ordr derivatives (like TRUREG or NRRIDG):

```
                  Dual Quasi-Newton Optimization
```

```
 Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
                   User Specified Gradient


 Iteration Start:
 N. Variables              5
 Criterion      -727.0990236        Max Grad Entry  11734.46983

Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope
   1    0    7   0 -692.9859 34.11311 3393.70  4e-005 -1377257
   2    0    8   0 -665.9089 27.07706 7121.63 0.22124 -385.469
   3    0   10   0 -657.4577 8.451175 5987.03 0.05620 -231.297
   4    0   12   0 -583.0190 74.43867 2080.88 0.72442 -228.174
   5    0   14   0 -578.3306 4.688438 1156.84 3.40333 -2.74300
   6    0   15   0 -576.5770 1.753621 5242.29 4.07595 -2.87567
   7    0   17   0 -570.0687 6.508208 1008.85 1.04995 -11.9374
   8    0   18   0 -563.1108 6.957903 622.061 2.47221 -6.58137
   9    0   19   0 -551.8181 11.29274 638.766 2.01923 -8.61453
  10    0   20   0 -546.4876 5.330482 2949.50 3.03226 -9.31049
  11    0   21   0 -543.4784 3.009240 1367.80 1.62064 -8.22281
  12    0   23   0 -541.9957 1.482702 442.628 1.02986 -2.30343
  13    0   25   0 -541.9416 0.054049 29.2999 1.00000 -0.10931
  14    0   27   0 -541.9286 0.012992 19.5271 1.25688 -0.02069
  15    0   29   0 -541.9280 6.0e-004 4.03505 1.06954  -1e-003
  16    0   31   0 -541.9280 3.0e-005 1.46013 1.26364  -5e-005

 Successful Termination After    16 Iterations
 GCONV convergence criterion satisfied.
 Criterion      -541.9280069        Max Grad Entry  1.460132805
 Slope SDirect. -4.7915e-005
 N. Function Calls        32        N. Gradient Calls        21
 N. Line Searches         16        Preproces. Time           2
 Time for Method          22        Effective Time           25


            ************************************
            Maximum Marginal Likelihood Estimates
            ************************************

Total Iterations            16    N Function Calls            33
N Gradient Calls            21    N Hessian Calls              2
Log Likelihood    -541.9280069    Deviance (-2logL)  1083.856014
AIC               -1073.856014    SBC               -1048.298453

 -------- ------------ ------------ ------------ ------------
 Variable     Estimate Stand. Error      Z Value      p-Value
 -------- ------------ ------------ ------------ ------------

 RESPONSE CODE    1 vs CODE 0
 ---------------------------
```

```
Fixed Effects
Alph_1   -4.909490436  0.796025685 -6.167502539  6.9e-010 (2)
Alph_2    0.000879822  0.000761776  1.154960506  0.248107 (2)
Alph_3    6.094408479  0.855627040  7.122739460  1.1e-012 (2)

Random Effects
Mean_1   -0.501839910  0.069646976 -7.205480203  5.8e-013 (2)

Random Effect Variance Term: Expressed as Standard Deviation
Mean_1    0.453733955  0.051939388  8.735835660  1.2e-018 (1)
               Note: (1) = 1-tailed p-value
                     (2) = 2-tailed p-value


                     **********************
                     Wald Confidence Limits
                     **********************

     Parameter          Estimate   LowWaldCL   UppWaldCL
     Alph_1_1         -4.9094904 -6.46967211 -3.34930876
     Alph_2_1          8.80e-004 -6.132e-004  0.00237288
     Alph_3_1          6.0944085  4.41741030  7.77140666
     Mean_1_1         -0.5018399 -0.63834548 -0.36533434
     COV_1_1           0.4537340  0.35193463  0.55553328

      Covariance Matrix and ASE Based on Hessian Matrix
      Wald Confidence Intervals Based on Hessian Matrix
```

For the Type 1 analysis, starting from the intercept single effects are being added to the model in the order in which they are specified in the model statement. Therefore, the last line of the table corresponds to the analysis of the complete model as reported above.

```
                  ********************************
                  LR Statistics for Type 1 Analysis
                    (Effects are Added One-by-one)
                  ********************************

Effect           Npar  Deviance          F   Pr>F  ChiSquare Pr>Chi
Intercept           1 1554.1833     .        .        .        .
X2                  2 1548.0512  6.1320404 0.0134  6.1320404 0.0133
X3                  4 1156.5658  195.74272 0.0000  391.48545 0.0000
X4                  5 1083.8560  72.709777 0.0000  72.709777 0.0000
```

For the Type 3 analysis, each separate effect in the model statement is being dropped (and later added again). The table shows which of the effects could be redundand and perhaps could be dropped for reducing the model. We can see, that dropping the fixed effect X4 shows the smallest increase in the deviance, but dropping the random effect X3 shows the largest increase:

```
                  ********************************
```

```
                LR Statistics for Type 3 Analysis
                   (Single Effects are Dropped)
                *********************************

Effect          Npar  Deviance          F   Pr>F  ChiSquare Pr>Chi
X2                 4 1133.8512  49.995232 0.0000  49.995232 0.0000
X3                 3 1403.2126  159.67827 0.0000  319.35654 0.0000
X4                 4 1085.2370  1.3809494 0.2402  1.3809494 0.2399
```

When random effects are present we also print a large table of the *empirical Bayes estimates* of the random effects as is recommended by Hedeker (1999):

```
            *******************************************
            Empirical Bayes Estimates of Random Effect(s)
            *******************************************

               N    ID_Var  Nobs       P_Mean     P_StdDev

               1      1001    12 -0.512947835  0.564980235
               2      1002    12  1.050199166  0.073941271
               3      1004    12  1.988080475  0.438772111
              ..........................................
             139      1144     8  0.160099832  0.421944276
             140      1145    12 -1.081161153  0.701567907
```

Using those *empirical Bayes estimates* Θ_i of the random effects we apply formula (Hedeker, 1999):

$$z_{ijk} = \mathbf{x}_{ij}^T(\mathbf{T}_k\mathbf{\Theta}_i + \mu_k) + \mathbf{w}_{ij}^T\alpha_k$$

to obtain the scores x_{ijk} of the linear model. Afterward we apply formulas(1) and (2) by Hedeker (1999) to score the logistic model:

$$P_{ijk} = P(y_{ij} = k|\beta, \alpha) = \frac{exp(z_{ijk})}{1 + \sum_{h=1}^{K} exp(z_{ijh})}$$

for $k = 1, \ldots, K$ and

$$P_{ij0} = P(y_{ij} = 0|\beta, \alpha) = \frac{1}{1 + \sum_{h=1}^{K} exp(z_{ijh})}$$

and obtain a very large table (here for space reasons reduced):

```
                  ****************************
                  Predicted Values and Residuals
                  ****************************

    ID_Var   Rep    Nobs         Yobs         Yhat     Residual
      1001     1       1  1.000000000  1.000000000  0.000000000
               2       2  0.000000000  0.000000000  0.000000000
               3       3  1.000000000  1.000000000  0.000000000
               4       4  0.000000000  0.000000000  0.000000000
               5       5  0.000000000  0.000000000  0.000000000
```

```
            6       6  0.000000000  0.000000000  0.000000000
            7       7  0.000000000  0.000000000  0.000000000
            8       8  0.000000000  0.000000000  0.000000000
            9       9  0.000000000  0.000000000  0.000000000
           10      10  0.000000000  0.000000000  0.000000000
           11      11  0.000000000  0.000000000  0.000000000
           12      12  0.000000000  0.000000000  0.000000000
1002        1      13  1.000000000  0.989142804  0.010857196
            2      14  1.000000000  0.995688960  0.004311040
            3      15  1.000000000  0.998295029  0.001704971
            4      16  1.000000000  0.999326767  0.000673233
            5      17  0.000000000  0.000265670 -0.000265670
            6      18  1.000000000  0.999895188  0.000104812
            7      19  1.000000000  0.999958653  4.1347e-005
            8      20  1.000000000  0.999983690  1.6310e-005
            9      21  0.000000000  6.4336e-006 -6.4336e-006
           10      22  1.000000000  0.999997462  2.5378e-006
           11      23  1.000000000  0.999998999  1.0010e-006
           12      24  0.000000000  3.9487e-007 -3.9487e-007
.............................................................
```

We are absolutely puzzled that we obtain an almost perfect fit. The SEE is about 2.1, the MSE is 10^{-3} and there is only one (very small) misclassification:

```
                ******************************
                Evaluation of Training Data Fit
                ******************************

Index                                 Value       StdErr
Absolute Classification Error             1            .
Concordant Pairs                  100.0000000          .
Discordant Pairs                  0.000000000          .
Tied Pairs                        0.000000000          .
Classification Accuracy           99.91843393          .
Goodman-Kruskal Gamma             1.000000000  0.000000000
Kendall Tau_b                     0.998156963  0.001840587
Stuart Tau_c                      0.882680583  0.018317091
Somers D C|R                      0.998783455  0.001215805


                    Classification Table
                    --------------------

                             |    Predicted
                   Observed  |        0       1
                   ----------|-----------------
                          0  |      821       1
                          1  |        0     404

               Area under the ROC Curve (Disc.) = 1
```

```
Area under the ROC Curve (Cont.) = 1
```

What am I doing wrong here?

3.3 Comparison with PROC NLMIXED

The following would be an application of PROC NLMIXED. Note, that we had to use excellent starting values to obtain results:

```
proc nlmixed data=anal qmax=500
 tech=nrridg qtol=.01 absgconv=.1;
   parms b0 -4  b1 .0 b2 .5 b3 .5 var 3.;
   bounds var >= 0;
   eta = b0 + b1*LOWESTCD + b2*PC + (b3+u)*WAVE;
   expeta = exp(eta);
   p = expeta/(1+expeta);
   model bi_mems1 ~ binary(p);
   random u ~ normal(0,var) subject=ID;
run;
```

The NLMIXED Procedure

Specifications

Data Set	WORK.ANAL
Dependent Variable	bi_mems1
Distribution for Dependent Variable	Binary
Random Effects	u
Distribution for Random Effects	Normal
Subject Variable	ID
Optimization Technique	Newton-Raphson Ridge
Integration Method	Adaptive Gaussian

Dimensions

Observations Used	987
Observations Not Used	239
Total Observations	1226
Subjects	118
Max Obs Per Subject	12
Parameters	5
Quadrature Points	1

Parameters

b0	b1	b2	b3	var	NegLogLike
-4	0	0.5	0.5	3	870.671696

Iteration History

Iter	Calls	NegLogLike	Diff	MaxGrad	Rho
1	22	764.115323	106.5564	7017.128	1
2*	29	729.929363	34.18596	2995.232	0.75483
3	36	725.056892	4.872471	270.4111	1.13672
4	43	722.41746	2.639432	70.32464	1.96758
5	50	712.283668	10.13379	69.19804	1.98395
6	57	675.853737	36.42993	64.5548	1.93408
7*	64	583.629729	92.22401	45.9046	1.72872
8*	71	510.497822	73.13191	511.4089	1.37411
9*	79	496.190285	14.30754	60.15225	0.00049
10	86	490.257278	5.933007	21.84992	2.07585
11	94	486.810479	3.446798	15.05242	0.0009
12	101	484.924463	1.886016	14.57428	2.03731
13	109	480.812949	4.111514	60.46501	0.00134
14	116	479.223742	1.589207	22.67965	1.88025
15	123	477.711656	1.512087	183.3703	1.055
16	130	476.32922	1.382436	3.21744	1.63806
17	137	474.242215	2.087005	14.17542	1.63406
18	144	472.773642	1.468573	23.58471	1.35459
19	151	472.511372	0.26227	6.555877	1.1475
20	158	472.503987	0.007385	0.233048	1.04185
21	165	472.503972	0.000015	0.000553	1.00028

NOTE: ABSGCONV convergence criterion satisfied.

Fit Statistics

-2 Log Likelihood	945.0
AIC (smaller is better)	955.0
AICC (smaller is better)	955.1
BIC (smaller is better)	968.9

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha
b0	-4.3637	0.7709	117	-5.66	<.0001	0.05
b1	0.000639	0.000818	117	0.78	0.4362	0.05
b2	5.6741	0.8108	117	7.00	<.0001	0.05
b3	-0.3792	0.07067	117	-5.37	<.0001	0.05
var	0.1821	0.05589	117	3.26	0.0015	0.05

Parameter Estimates

```
Parameter      Lower       Upper    Gradient

b0           -5.8904     -2.8370    3.113E-6
b1          -0.00098    0.002260    0.000553
b2            4.0682      7.2799    6.519E-8
b3           -0.5191     -0.2392    4.342E-6
var          0.07143      0.2928    -9.31E-6
```

One reason for the differences could be the different treatment of missing values. The `glmixd` function in CMAT by default imputes mean values for missing values. Using the `nomiss` option, the `glmixd` function would ignore all observations with missing values. Using the `nomiss` option with `glmixd` we obtain similar results:

```
                    ******************
                    Optimization Start
                    ******************

                   Parameter Estimates
                   -------------------

          Parameter     Estimate   Gradient

           1 X1      -4.90949040  15.974675
           2 X2       8.800e-004  2092.3887
           3 X3       6.09440850  13.816968
           4 X4      -0.50183990  62.421787
           5 X5       0.45373400  44.071525

        Value of Objective Function =     -473.423


              Dual Quasi-Newton Optimization
  Dual Broyden - Fletcher - Goldfarb - Shanno Update (DBFGS)
                  User Specified Gradient


   Iteration Start:
   N. Variables              5
   Criterion      -473.4230373        Max Grad Entry  2092.388717

  Iter rest nfun act   optcrit  difcrit maxgrad   alpha    slope

Too big for exp: [1] (2366.43 + -0.439418 + -3.7928)=2362.19

     1    0    6   0 -472.8223 0.600730 243.250  3e-005 -43843.8
     2    0    8   0 -472.2393 0.583011 168.698 0.04303 -28.5479
     3    0    9   0 -471.4394 0.799911 105.269 0.76484 -1.90566
     4    0   11   0 -471.2587 0.180708 120.328 2.00639 -0.18013
     5    0   12   0 -471.2434 0.015250 41.0537 1.00000 -0.02178
     6    0   14   0 -471.2335 9.9e-003 77.4488 1.61767 -0.01188
     7    0   17   0 -470.9917 0.241846 16.2045 54.5739  -9e-003
```

```
   8    0   19   0 -470.9802 0.011501 10.3573 1.02689 -0.02441
   9    0   21   0 -470.9801 1.1e-004 0.06150 1.01671  -2e-004
  10    0   23   0 -470.9801 7.9e-008  7e-003 1.00000  -2e-007

 Successful Termination After    10 Iterations
 GCONV convergence criterion satisfied.
 Criterion      -470.9800877        Max Grad Entry  0.007273418
 Slope SDirect. -1.6125e-007
 N. Function Calls        24        N. Gradient Calls        15
 N. Line Searches         10        Preproces. Time          15
 Time for Method         157        Effective Time          179


            ************************************
            Maximum Marginal Likelihood Estimates
            ************************************

Total Iterations            10    N Function Calls            25
N Gradient Calls            15    N Hessian Calls              2
Log Likelihood    -470.9800877    Deviance (-2logL)  941.9601755
AIC               -931.9601755    SBC               -907.4868253

 -------- ------------ ------------ ------------ ------------
 Variable     Estimate Stand. Error      Z Value      p-Value
 -------- ------------ ------------ ------------ ------------

 RESPONSE CODE    1 vs CODE 0
 ----------------------------

 Fixed Effects
 Alph_1   -4.219882032  0.750919711 -5.619618146  1.9e-008 (2)
 Alph_2    0.000566628  0.000736011  0.769863459  0.441381 (2)
 Alph_3    5.603510094  0.800575238  6.999354751  2.6e-012 (2)

 Random Effects
 Mean_1   -0.463918704  0.077729705 -5.968357966  2.4e-009 (2)

 Random Effect Variance Term: Expressed as Standard Deviation
 Mean_1    0.429899073  0.058947135  7.292959624  1.5e-013 (1)
               Note: (1) = 1-tailed p-value
                     (2) = 2-tailed p-value


                    **********************
                    Wald Confidence Limits
                    **********************

     Parameter          Estimate   LowWaldCL   UppWaldCL
     Alph_1_1         -4.2198820 -5.69165762 -2.74810644
     Alph_2_1          5.67e-004 -8.759e-004  0.00200918
     Alph_3_1          5.6035101  4.03441146  7.17260873
```

```
Mean_1_1         -0.4639187 -0.61626613 -0.31157128
COV_1_1           0.4298991  0.31436481  0.54543333

 Covariance Matrix and ASE Based on Hessian Matrix
 Wald Confidence Intervals Based on Hessian Matrix
```

4 Notes on Convergence of `glmixd`

Even though I have found and corrected a number of bugs in the OLS estimation of initial values for function `glmixd()` (my CMAT implementation of Don Hedeker's MIXNO and MIXOR), there may still be convergence problems with bad starting values, especially when the data contain variables with very different scalings. Using the `glim` function I looked if the optimal parameters of the common logistic model would be better initial values than the OLS estimates. The few examples which I ran showed that OLS gave better initial values than LOGISTIC. Maybe that has to do with the fact that the linear OLS is more stable wrt. parms than LOGISTIC since OLS is a linear model and LOGISTIC is nonlinear. Any small model changes could have more impact on the parms of nonlinear modeling. So I was not too convinced by using the LOGISTIC parms as initial values. (If somebody has a different experience about using results of the common LOGISTIC estimation as initial values, please let me know, and I will add a fast approximation for the use as initial values in `glmixd()`.)
Experimenting with starting values I found the following 2-step approach quite useful:

1. First I use an optimization method which uses only the gradient. With my software in CMAT I prefer to use DUQUAN, DBLDOG, or even CONGRA and run a few iterations using a very rough termination criterion.

2. Then I save the resulting "sub-optimal" parameters and use them as initial values with an optimization method that uses an approximation to the Hessian (like you use with the cross product Jacobian in Fisher scoring algorithm). I have methods like TRUREG and NRRIDG. This second step now has good starting values and can result in rather high precision results. Bottleneck of the precision remains the precision of the quadrature algorithm which declines with the number of random effects.

Using only the second step is much more sensitive toward good initial estimates. The TRUREG and NRRIDG optimization techniques print an asterisk in the iteration history when the approximation of the Hessian is singular. In some cases this can be a valuable indication of bad specified models, but may also have to do with overflows of the `exp` function when computing function and derivatives.