Course: HS237C, Spring 2005
Instructor: Ettner
Lecture: Multi-level models

```
NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
      Licensed to UNIV OF CALIFORNIA-LOS ANGELES-SYSTEMWIDE-IN, Site 0029107027.
NOTE: This session is executing on the WIN_98  platform.



NOTE: SAS initialization used:
      real time           3.74 seconds

1    libname ssd "c:\simulation\"   ;
NOTE: Libref SSD was successfully assigned as follows:
      Engine:        V8
      Physical Name: c:\simulation
2
3    /*************************************************************************

4
5    Programmer(s):  Susan Ettner and Alfonso Ang, based on GAUSS code written by Ted
Thompson and
5  !  with assistance from Mark Stevens and Neil Steers.
6
7    Date:                   June 21, 2004
8
9    This program estimates multi-level logistic regressions using PROC GLMMIX in SAS.
It outputs
9  !  the estimates to PROC IML to get the associated relative risks and simulated
confidence
9  ! intervals.
10
11   Notes:
12
13   The program is currently set up to run only one outcome at a time.
14
15   The dataset is assumed to contain complete data for all covariates; any imputation
should
15 ! have been performed previously.
16
17   All categorical variables need to have already been turned into a series of mutually
17 ! exclusive and exhaustive dichotomous indicators, as no class statements are used for
the
17 ! covariates.
18
19   The first part of the program needs to be modified by the user.
20
21   The PROC GLMMIX output should be ignored, as the degrees of freedom may not always
be correct
21 !  for the cluster-level covariates.
22
23   The predictions are calculated at zero values for all random effects.
24
25   *************************************************************************/;

26
27   * To run this program, you need to define global variables, as follows;
28   * iter is the number of simulations desired, typically 10,000;
29
```

```
30   %let dataset    =  ssd.test2;
31   %let outcome    =  lpa;
32   %let regs       =  sex_r age latino;
33   %let tit1       =  Dependent Variable is Lipid Profile Assessment;
34   %let tit2       =  Configuration 0 is female, configuration 1 is male;
35   %let tit3       =  Predicted probabilities, difference in comparing men to women;

36   %let tit4       =  Predicted probabilities and RR;
37   %let alpha      =  .05;
38   %let id1        =  ohealthplan;
39   %let id2        =  cluster;
40   %let iter       =  200;
41
42
43   /* If you want to set a fixed seed in order to replicate or compare results, you
need to give
43 !  a value >0 for the following global variable.  If you specify a value <=0, then the
seed
43 ! will be generated from the system clock instead */;
44   %let seed       =   1;
45
46   /* Note that the degrees of freedom used by glimmix for the multi-level models will
be
46 ! incorrect.  We set method to "Kenward and Roger" to let SAS approximate the degrees
of
46 ! freedom */;
47
48   *  The libnames and options can be modified if desired;
49
50   libname ssd '.';
NOTE: Libref SSD was successfully assigned as follows:
      Engine:        V8
      Physical Name: C:\simulation
51   libname library '.';
NOTE: Libname LIBRARY refers to the same physical library as SSD.
NOTE: Libref LIBRARY was successfully assigned as follows:
      Engine:        V8
      Physical Name: C:\simulation
52
53   options ps=64 ls=64;
54   options nofmterr;
55
56
57   /* The SAS macro glmm800 is needed to run glmmix.  The path
57 !  needs to be modified if this macro is not in the same
57 ! directory */;
58
59   %inc 'C:\simulation\glmm800.sas' /nosource ;
1775
1776  * This code makes sure that only observations with
1776! complete data for the dependent variable are retained;
1777  * It does not need to be modified;
1778
1779  data completedata;
1780  set &dataset;
1781  array colmiss[*] &regs &outcome;
1782     do i=1 to dim(colmiss);
```

```
    if colmiss[i]= . then delete;
    one=1;
  end;
  run;

NOTE: There were 2147 observations read from the data set
      SSD.TEST2.
NOTE: The data set WORK.COMPLETEDATA has 980 observations and 9
      variables.
NOTE: DATA statement used:
      real time           0.26 seconds


/**********************************************************
*****************

The following code needs to be modified to construct the
two datasets with the configurations of covariate values
that you wish to compare in calculating relative risks or
differences in predicted probabilities.

**********************************************************
****************/;

data data0;
        set completedata;
        keep one &regs;
        * ADD CODE HERE FOR THE FIRST CONFIGURATION OF
COVARIATE VALUES;
        sex_r=0;
run;

NOTE: There were 980 observations read from the data set
      WORK.COMPLETEDATA.
NOTE: The data set WORK.DATA0 has 980 observations and 4
      variables.
NOTE: DATA statement used:
      real time           0.05 seconds



data data1;
        set completedata;
        keep one &regs;
        * ADD CODE HERE FOR THE SECOND CONFIGURATION OF
COVARIATE VALUES;
        sex_r=1;
run;

NOTE: There were 980 observations read from the data set
      WORK.COMPLETEDATA.
NOTE: The data set WORK.DATA1 has 980 observations and 4
      variables.
NOTE: DATA statement used:
      real time           0.04 seconds
```

```
/**********************************************************************
*****************

The remainder of the code creates the macro to run the
multi-level logit regression with relative risks and runs
the macro for the dependent variable specified above.  No
modification is necessary past this point.

**********************************************************************
*****************/;

%macro multilevel_logit;

%glimmix(data=completedata,
   procopt=%str(order=internal covtest),
   stmts=%str(
   class &id2 &id1;
   model &outcome=&regs / ddfm=kenwardroger covb solution;
   random intercept &id2/ subject=&id1 ;
ods output  Covb=mixcovb;

),
   error=binomial, link=logit
  );
run;

************************;

* Get initial seed value.  If seed<=0, then generate seed
from the system clock;

data _null_;
   if &seed le 0 then do;
      seed = int(time());   /* get clock time in integer
seconds */
      put seed=;
      call symput('seed',left(put(seed,12.)));   /* store
seed as macro variable */
   end;
run;

* Use matrix mode to perform the rest of the calculations;

proc iml;

call symput ('n',left(char(&iter)));

* Create design matrices for the two configurations of
covariate values being compared;

use data0;
        read all var {one &regs} into x0;

* Get number of observations;
numobs = nrow(x0);
call symput ('numobs',char(numobs));
```

```
use data1;
        read all var {one &regs} into x1;

* Create vector of coefficient estimates;

use _soln;
        * Read in column corresponding to coefficient estimates;
        read all var {estimate} into beta;
        * Read in column corresponding to SEs;
        read all var { stderr } into stderr;
        * Read in column corresponding to t-statistics;
        read all var { tvalue } into tstat;
        * Read in column corresponding to p-values;
        read all var { probt } into pvalue;
        * Get the number of parameter estimates and create global for it;
         k = nrow(beta);
                 numobs = nrow(x0);

call symput ('numpar',left(char(k)));
%let colg = col&numpar;

* Create variance-covariance matrix;
use mixcovb;

read all var("col1":"&colg") into varcov;
se=sqrt(vecdiag(varcov));

* Print out sample size, #covariates, coefficients, SEs and p-values as a test;

print "Number of observations in the sample";
print numobs;
print "";

print "Number of covariates in the model including the intercept";
print k;
print "";

stats={Coef StdErr TStat PValue};
varname={one &regs};

Original_Estimate=beta||stderr||tstat||pvalue;

print "Estimated coefficients, standard errors, t-statistics and p-values";
print Original_Estimate[format=8.4 rowname=varname colname=stats];
print "" ;


* Create vector of zeros with the same number of rows as desired simulations;
```

```
* J(&n,1,0) creates a vector with &n rows and 1 column;
* All values in these vectors are initialized to 0;

simprob0        = J(&n, 1, 0);
simprob1        = J(&n, 1, 0);
simdiff         = J(&n, 1, 0);
simrr           = J(&n, 1, 0);

* Figure out lower and upper cutoffs for empirical
confidence intervals;
* ceil rounds up;

lowern = ceil(&alpha/2 * (&n +1)) ;
uppern = ceil( &n +1 - lowern );

* Create an &n x k matrix of simulated values called bstar
;
* Simulated values are random draws from a N(beta, varcov)
 distribution;
* The bstar matrix is &n rows of beta coefficients with
random noise added;
* Note that &n is the number of simulations, not the
number of observations;

seed  = &seed;
l     = t(root(varcov))                     /* calculate
cholesky root of VCV matrix */;
z     = normal(j(&numpar,&n,&seed)) /* generate
nvars*samplesize normals */;
x     = l*z                                 /* premultiply
 by cholesky root */;
x     = repeat(beta,1,&n)+x                 /* add in the
means */;
bstar = t(x)                            /* transpose to
get the final vector */;


* Check dimensions of bstar;

numsim  = nrow(bstar);
numparm = ncol(bstar);

print "Number of Simulations=";
print numsim;
print "";
print "Number of Parameter Estimates=";
print numparm;

*/ A loop is run &n times to fill in the zero vectors with
 the desired statistics.  Predicted probabilities are
calculated using the logit formula, multiplying each
configuration of covariate values by the ith row of
simulated beta coefficients.  Note that
exp(xb)/[1+exp(xb)] = 1/1+exp(-xb).
*/;

do j = 1 to &n;
```

```

* First calculate estimates of interest for each individual;
p0      = 1 / ( 1 + exp( -x0*bstar[j,]` ) );
p1      = 1 / ( 1 + exp( -x1*bstar[j,]` ) );
p1_p0   = p1 - p0;
rat     = p1 / p0;

* Then take averages over the entire sample;
simprob0[j,]    =   p0[+,]    / &numobs;
simprob1[j,]    =   p1[+,]    / &numobs;
simdiff[j,]     =   p1_p0[+,] / &numobs;
simrr[j,]       =   rat[+,]   / &numobs;

end;

* Get the predicted risk and CI for first configuration of covariate values;
* Using the original coefficient estimates;

* predicted risk for each individual and then averaged over sample;
xbeta  = -x0*beta ;
risk0  = 1 / ( 1 + exp( xbeta ) );
mrisk0 = risk0[+,] / &numobs;

* sort the simulations using the rank function;
* rank creates vector with the ranks of the corresponding elements;

tsimprb0 = t(simprob0);
sprob0 = tsimprb0;
tsimprb0[,rank(tsimprb0)] = sprob0;

* pick off simulation corresponding to lower cutoff;
lower0= tsimprb0[,lowern];

* pick off simulation corresponding to upper cutoff;
upper0= tsimprb0[,uppern];

* Get the predicted risk and CI for second configuration of covariate values;
* Using the original coefficient estimates;

* predicted risk for each individual and then averaged over sample;
risk1  = 1 / ( 1 + exp( -x1*beta ) );

mrisk1 = risk1[+,] / &numobs;

* sort the simulations using the rank function;
* rank creates vector with the ranks of the corresponding elements;

tsimprb1 = t(simprob1);
sprob1 = tsimprb1;
tsimprb1[,rank(tsimprb1)] = sprob1;
```

```
* pick off simulation corresponding to lower cutoff;
lower1= tsimprb1[,lowern];

* pick off simulation corresponding to upper cutoff;
upper1= tsimprb1[,uppern];

* Get the difference in predicted risk and CI;

* predicted difference for each individual and then averaged over sample;
diff  = risk1 - risk0;
mdiff = diff[+,] / &numobs;

* sort the simulations using the rank function;
* rank creates vector with the ranks of the corresponding elements;

tsimdiff = t(simdiff);
sdiff = tsimdiff;
tsimdiff[,rank(tsimdiff)] = sdiff;

* pick off simulation corresponding to lower cutoff;
lowerdiff= tsimdiff[,lowern];

* pick off simulation corresponding to upper cutoff;
upperdiff= tsimdiff[,uppern];

* Get the relative risk and CI;

* predicted RR for each individual and then averaged over sample;
rr  = risk1 / risk0;
mrr = rr[+,] / &numobs;

* sort the simulations using the rank function;
* rank creates vector with the ranks of the corresponding elements;

tsimrr = t(simrr);
srr = tsimrr;
tsimrr[,rank(tsimrr)] = srr;

* pick off simulation corresponding to lower cutoff;
lowerrr= tsimrr[,lowern];

* pick off simulation corresponding to upper cutoff;
upperrr= tsimrr[,uppern];

* Print the output with labels;

stats1={LowerCL Risk0 UpperCL};
stats2={LowerCL Risk1 UpperCL};
stats3={LowerCL RiskDiff UpperCL};
stats4={LowerCL RR UpperCL};

mean_risk_0=lower0||mrisk0||upper0;
```

```
mean_risk_1=lower1||mrisk1||upper1;
mean_diff_in_risk=lowerdiff||mdiff||upperdiff;
mean_relative_risk=lowerrr||mrr||upperrr;

print "&tit1";
print "&tit2";
print "&tit3";
print "&tit4";

print mean_risk_0[format=10.2 colname=stats1];
print "";

print mean_risk_1[format=10.2 colname=stats2];
print "";

print mean_diff_in_risk[format=10.2 colname=stats3];
print "";

print mean_relative_risk[format=10.2 colname=stats4];
print "";

* Exit SAS-IML;

quit;

%mend;

%multilevel_logit;
```

```
       The GLIMMIX Macro

Data Set            : WORK.COMPLETEDATA
Error Distribution : BINOMIAL
Link Function       : LOGIT
Response Variable   : LPA


    GLIMMIX Iteration History

Iteration    Convergence criterion
    1            0.1833921164
    2            0.0720221664
    3            0.0015557954
    4            5.274373E-6
    5            8.936321E-10

Output from final Proc Mixed run:

NOTE: DATA statement used:
      real time           0.04 seconds


NOTE: IML Ready
NOTE: Exiting IML.
NOTE: 225 workspace compresses.
NOTE: PROCEDURE IML used:
      real time           1.03 seconds
```