Supplement 9.2. SAS commands for evaluations summarized in Table 9.3 (MQL1, PQL1, RPQL1, and Laplace estimation)

Brian R. Gray, US Geological Survey, Upper Midwest Environmental Sciences Center, La Crosse, Wisconsin, USA.

This supplement provides SAS commands for evaluating estimators of variance components using logit-normal models of binomial data derived from two-way cross-classification designs. This SAS routine evaluates MQL1, PQL1, RPQL1, and Laplace estimation. Corresponding routines were used for all scenarios summarized in Table 9.3 of Chapter 9. A .sas format file containing the following commands is available on request from gitzenr@missouri.edu.

```
* PURPOSE: estimate variance components (VCs) using logit-normal models of binomial data derived from 2-way crossed
classification designs;
* CREATED: 11 May 2010 by Brian Gray from corresponding linear models code;
* Revision history:;


/* THIS ALGORITHM IS PROVIDED WITHOUT WARRANTY, WHETHER EXPRESSED OR IMPLIED.  PLS SEND QUESTIONS OR ERRORS TO BRIAN GRAY,
brgray@usgs.gov */


* ENTER PARAMETER/DESIGN VALUES;
%let beta0 = -2;                        * assign mean on logit link scale (median on measurement scale);
%let var_site = .3;                     * assign among-site intercept variation on log scale;
%let var_yr = .3;                       * assign among-year intercept variation on log scale;
%let var_sitebyyr = .15;       * assign among-sitexyear intercept variation on log scale;
%let n = 5;                             * assign sample size w/in site-year clusters;
%let sites = 20;                        * assign number sites;
%let years = 5;                         * assign number years;
%let simulations = 200;                 * determine number simulated data sets;

options nocenter;
title1 "2-way factorial design: sites = &sites, years=&years, index = &n. var(yr)=&var_yr, var(site)=&var_site,
var(sitebyyr)=&var_sitebyyr, sims=&simulations";

* delete old datasets;
proc datasets library=work kill; run;

* GENERATE INPUT DATASET;
* generate year effects;
data yr_effects;
        do sim = 1 to &simulations;
        do year=1 to &years;
                yr_effect=sqrt(&var_yr)*rannor(123);
                do site=1 to &sites;
                        output;
                end;
        end;
        end;
run;
proc sort data=yr_effects out=yr_effects_sort; by sim site year; run;

* generate site effects;
data site_effects;
        do sim = 1 to &simulations;
        do site = 1 to &sites;
                site_effect = sqrt(&var_site)*rannor(234);
                do year= 1 to &years;
```

```
                        sitebyyr_effect = sqrt(&var_sitebyyr)*rannor(345);
                output;
                end;
  end;
        end;
run;

* merge site and year effects, and generate outcomes;
data twoway;
        merge yr_effects_sort site_effects;
        by sim site year;
        eta = &beta0 + yr_effect + site_effect + sitebyyr_effect;
        eta_inv=(1+exp(-eta))**(-1);
        y=ranbin(456,&n,eta_inv);
        n=&n;
        var_yij=eta_inv*(1-eta_inv);
run;


* ESTIMATE VARIANCE COMPONENTS BY METHOD;
* per SAS email 5/26/10, SAS' ?SPL and ?MPL correspond to Breslow and Claytons' PQL1 and MQL1,respectively.
*       see also 4/14/2008 SAS email and pp 539, 541, Littell et al (2006);
ods listing close;
proc glimmix data=twoway method=mmpl;
        class year site;
        model y / n = / s;
        random int / sub=site;
        random int / sub=year;
        random int / sub=year*site;
        by sim;
  ods output convergencestatus=convMQL iterhistory=iterMQL parameterestimates=parmestsMQL covparms=covparmsMQL;
run;

* estimate variance components using PQL;
proc glimmix data=twoway method=mspl;
        class year site;
        model y / n = / s;
        random int / sub=site;
        random int / sub=year;
        random int / sub=year*site;
        by sim;
  ods output convergencestatus=convPQL iterhistory=iterPQL parameterestimates=parmestsPQL covparms=covparmsPQL;
run;

* estimate variance components using RPQL;
proc glimmix data=twoway method=rspl;
        class year site;
```

```
        model y / n = / s;
        random int / sub=site;
        random int / sub=year;
        random int / sub=year*site;
        by sim;
  ods output convergencestatus=convRPQL iterhistory=iterRPQL parameterestimates=parmestsRPQL covparms=covparmsRPQL;
run;

* estimate variance components using Laplace estimation;
proc glimmix data=twoway method=laplace;
        class year site;
        model y / n = / s;
        random int / sub=site;
        random int / sub=year;
        random int / sub=year*site;
        by sim;
ods output convergencestatus=convLaplace iterhistory=iterLaplace parameterestimates=parmestsLaplace covparms=covparmsLaplace;
run;
ods listing;


* COMBINE RESULTS FOR ALL ESTIMATION METHODS;
* create all-method convergence information dataset;
data conv;
        length method $10.;
        set convMQL(in=MQL) convPQL(in=PQL) convRPQL(in=RPQL) convLaplace(in=Laplace);
        if MQL then method="_1MQL"; else if PQL then method="_2PQL"; else if RPQL then method="_3RPQL";
        else if Laplace then method="_4Laplace"; else method = "_6ERROR";
        if status = 1 or pdg=0 then convpdg=0; else convpdg=1;
        converge=1-status;
run;

* create all-method parameter estimate dataset;
data parmests;
        length method $10.;
        set parmestsMQL(in=MQL) parmestsPQL(in=PQL) parmestsRPQL(in=RPQL) parmestsLaplace(in=Laplace);
        if MQL then method="_1MQL"; else if PQL then method="_2PQL"; else if RPQL then method="_3RPQL"; else if Laplace then
                method="_4Laplace"; else method = "_6ERROR";
        inthat=estimate;
        SEinthat=stderr;
        keep sim method inthat SEinthat;
run;

* create all-method estimated covariance matrix dataset;
data covparms;
        length method $10.;
        set covparmsMQL(in=MQL) covparmsPQL(in=PQL) covparmsRPQL(in=RPQL) covparmsLaplace(in=Laplace);
```

```
        if MQL then method="_1MQL"; else if PQL then method="_2PQL"; else if RPQL then method="_3RPQL"; else if Laplace then
                method="_4Laplace"; else method = "_6ERROR";
        varhat =estimate;
        SDhat =sqrt(estimate);
        SE_varhat=stderr;
        keep sim method subject covparm estimate varhat SDhat SE_varhat;
run;

* merge convergence and parameter information datasets;
data covparmsplus;
        merge covparms parmests conv;
        by method sim;
run;
proc sort data=covparmsplus out=covparmsplussort;
        by subject method sim;
run;


* CALCULATE MODEL CONVERGENCE PROPORTIONS, AND PREVALENCES OF NON-POSITIVE DEFINITE HESSIAN MATRICES;
* estimate model convergence proportions;
title2 "Convergence proportions";
proc means data=conv noprint; var converge; output out=converge mean=probconverge; by method;
proc print data=converge(drop=_TYPE_ _FREQ_) noobs; run;
title2 "Proportion models with positive definite Hessian matrices (of models that converged)";
proc means data=conv noprint; var pdg; output out=pdg mean=probpdg; by method; where converge;
proc print data=pdg(drop=_TYPE_ _FREQ_) noobs; run;


* DETERMINE NUMBER OF 0 OR NEAR-0 (<1e-4 units) VARIANCE ESTIMATES FOR MODEL FITS WITH non-PD G MATRICES;
title2 "% zero estimates|model convergence";
data minvar; set covparmsplussort; if varhat lt 1e-4 then varhat0=1; else varhat0=0;
proc means data=minvar noprint; var varhat0; by subject method sim; output out=varhat0stats sum=sum0s; where converge; run;
proc means data=varhat0stats noprint; var sum0s; by subject method ; output out=varhat0statssum mean=mean median=median
min=min max=max; run;
proc print data=varhat0statssum(drop=_TYPE_ _FREQ_) noobs; run;


* CALCULATE VARIANCE STATISTICS (conditional on convergence and, if selected, positive definite G matrix);
title2 "Variance component estimates";
proc means data=covparmsplussort noprint;
        var varhat SDhat SE_varhat inthat SEinthat;
        output out=varcompstats mean=meanvarhat meanSDhat meanSEvarhat meaninthat meanSEinthat stddev=SEvarhat SESDhat;
        by subject method;
        where converge;* and pdg;
run;
data varcompstats2;
        set varcompstats(drop=_TYPE_);
```

```
        pctconverge=_freq_/&simulations;
        * calculate RMSE(var_hat);
        if subject="site" then MSEvarhat=(meanvarhat-&var_site)**2 + SEvarhat**2;
                else if subject="year" then MSEvarhat=(meanvarhat-&var_yr)**2 + SEvarhat**2;
                else if subject="year*site" then MSEvarhat=(meanvarhat-&var_sitebyyr)**2 + SEvarhat**2;
        RMSEvarhat=sqrt(MSEvarhat);
        * calculate RMSE(SD_hat);
        if subject="site" then MSESDhat=(meanSDhat-sqrt(&var_site))**2 + SESDhat**2;
                else if subject="year" then MSESDhat=(meanSDhat-sqrt(&var_yr))**2 + SESDhat**2;
                else if subject="year*site" then MSESDhat=(meanSDhat-sqrt(&var_sitebyyr))**2 + SESDhat**2;
        RMSESDhat=sqrt(MSESDhat);
proc print data=varcompstats2 noobs;
        format _numeric_ 8.3 pctconverge 8.2 MSEvarhat MSESDhat 8.4 _freq_;
        var subject method pctconverge meanvarhat meanSEvarhat SEvarhat RMSEvarhat meanSDhat SESDhat RMSESDhat meaninthat
        meanSEinthat;
run;


* PLOT ESTIMATES;
proc boxplot data=covparmsplussort;
        plot SDhat*method;
        by subject;
        where converge;
run;
quit;
```