WEB-BASED SUPPLEMENTARY MATERIALS TO

A Bivariate Measurement Error Model for Semicontinuous and Continuous Variables: Application to Nutritional Epidemiology.

Victor Kipnis, Laurence S. Freedman, Raymond J. Carroll, Douglas Midthune

WEB APPENDIX A

Likelihood for the Bivariate Measurement Error Model

We now develop the expression for the likelihood for model (8)-(9) that will allow the application of maximum likelihood to estimate the parameters of the model. Let $\Phi_q\left(\xi,\Sigma\right)$ denote the *q*-dimensional cumulative multivariate normal distribution function of a random vector ξ with mean zero and covariance matrix Σ and $\phi_q\left(\xi,\Sigma\right)$ the corresponding density function. Let $\mu_{Cli}=\boldsymbol{\beta}_{Cl}^{\mathrm{t}}\mathbf{X}_i+u_{Cli},\ l=1,2$, and $\mu_{Ei}=\boldsymbol{\beta}_{E}^{\mathrm{t}}\mathbf{X}_i+u_{Ei}$. In what follows, the distributional functions and likelihoods are all conditional on covariates $\mathbf{X}_i$. Define $\Delta_{Eij}=R_{Eij}^{*}\left(\lambda_{R_E}\right)-\mu_{Ei}$ and $\Delta_{C2ij}=R_{C2ij}^{*}\left(\lambda_{R_C}\right)-\mu_{C2i}$ Conditional on random effects $\mathbf{u}_i$, the joint density of observed data $\left(R_{Cij},R_{Eij}\right),\ i=1,\ldots,m,\ j=1,\ldots,J_i$, in the calibration substudy can be written as

$$
\begin{aligned}
f\left(R_{Cij},R_{Eij}\mid\mathbf{X}_i,\mathbf{u}_i\right)&=R_{Eij}^{\lambda_{R_E}-1}\int_{-\infty}^{-\mu_{C1i}}\phi_2\left\{\varepsilon_{C1ij},\Delta_{Eij};\Sigma_{\varepsilon_{C1},\varepsilon_E}\right\}d\varepsilon_{C1ij}I\left[R_{Cij}=0,R_{Eij}>0\right]\\
&+R_{Cij}^{\lambda_{R_C}-1}R_{Eij}^{\lambda_{R_E}-1}\int_{-\infty}^{\mu_{C1i}}\phi_3\left\{\varepsilon_{C1ij},\Delta_{C2ij},\Delta_{Eij};\Sigma_{\varepsilon}\right\}d\varepsilon_{C1ij}I\left[R_{Cij}>0,R_{Eij}>0\right]
\end{aligned}
\tag{A1}
$$

Taking into account the structured variance-covariance matrix Σ_ε of within-person errors $\boldsymbol{\varepsilon}_{ij}$ given by (9), the conditional distributions of ε_{C1ij} given $\varepsilon_{Eij}=\varepsilon_E$ and of ε_{C1ij} given $\varepsilon_{C2ij}=\varepsilon_{C2},\varepsilon_{Eij}=\varepsilon_E$ are, respectively,

$$\left(\varepsilon_{C1ij} \mid \varepsilon_{Eij}\right) \overset{iid}{\sim} \text{Normal}\left(\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\left(\varepsilon_{\text{E}}\right); \sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\right),$$

$$\left(\varepsilon_{C1ij} \mid \varepsilon_{C2ij}, \varepsilon_{Eij}\right) \overset{iid}{\sim} \text{Normal}\left(\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\left(\varepsilon_{C2}, \varepsilon_{\text{E}}\right); \sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\right),$$

where $\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\left(\varepsilon_{\text{E}}\right) = \rho_{\varepsilon_{C1},\varepsilon_E}\dfrac{\varepsilon_E}{\sigma_{\varepsilon_E}}$, $\sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}} = 1-\rho^2_{\varepsilon_{C1},\varepsilon_E}$,

$$\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\left(\varepsilon_{C2}, \varepsilon_{\text{E}}\right) = \frac{\rho_{\varepsilon_{C1},\varepsilon_{\text{E}}}}{1-\rho^2_{\varepsilon_{C2},\varepsilon_{\text{E}}}}\left(\frac{\varepsilon_{\text{E}}}{\sigma_E} - \rho_{\varepsilon_{C2},\varepsilon_{\text{E}}}\frac{\varepsilon_{C2}}{\sigma_{C2}}\right),\ \sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}} = \frac{1-\rho^2_{\varepsilon_{C1},\varepsilon_E}-\rho^2_{\varepsilon_{C2},\varepsilon_E}}{1-\rho^2_{\varepsilon_{C2},\varepsilon_E}}$$

Then

$$\begin{aligned}
&\phi_2\left\{\varepsilon_{C1ij}, g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei};\Sigma_{\varepsilon F1,\varepsilon E}\right\} = \phi_1\left\{\varepsilon_{C1ij}-\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\left(\varepsilon_{Eij}\right);\sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\right\}\phi_1\left(\varepsilon_{Eij};\sigma^2_{\varepsilon_E}\right)\\
&\phi_3\left\{\varepsilon_{C1ij}, g\left(R_{Cij};\lambda_{R_C}\right)-\mu_{C2i}, g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei};\Sigma_{\varepsilon}\right\} = \phi_1\left\{\varepsilon_{C_1ij}-\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\left(\varepsilon_{F2},\varepsilon_{\text{E}}\right);\sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\right\} \quad \text{(A2)}\\
&\qquad\qquad \times\phi_2\left\{g\left(R_{Cij};\lambda_{R_C}\right)-\mu_{C2i}, g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei};\Sigma_{\varepsilon_{CF2},\varepsilon_E}\right\},
\end{aligned}$$

and it follows from (A1) - (A2) that the likelihood for model (8)-(9) is given by

$$L \propto \prod_{i=1}^{n}\int\prod_{j=1}^{m_i} L_{1ij}^{1-I_{Fij}}\left(\mathbf{u}_i\right)L_{2ij}^{I_{Fij}}\left(\mathbf{u}_i\right)\phi_3\left(\mathbf{u}_i;\Sigma_u\right)d\mathbf{u}_i$$

where

$$\begin{aligned}
L_{1ij}\left(\mathbf{u}_i\right) &= R_{Eij}^{\lambda_{R_E}-1}\Phi_1\left[-\mu_{C1i}-\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\left\{g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei}\right\};\sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{\text{E}}}\right]\phi_1\left(\varepsilon_{Eij};\sigma^2_{\varepsilon_E}\right)\\
L_{2ij}\left(\mathbf{u}_i\right) &= R_{Cij}^{\lambda_{R_C}-1}R_{Eij}^{\lambda_{R_E}-1}\Phi_1\left[\mu_{F1i}-\mu_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\left\{g\left(R_{Cij};\lambda_{R_C}\right)-\mu_{C2i}, g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei}\right\};\sigma^2_{\varepsilon_{\text{C1}}\bullet\varepsilon_{C2},\varepsilon_{\text{E}}}\right]\\
&\qquad\times\phi_2\left\{g\left(R_{Cij};\lambda_{R_C}\right)-\mu_{C2i}, g\left(R_{Eij};\lambda_{R_E}\right)-\mu_{Ei};\Sigma_{\varepsilon_{CF2},\varepsilon_E}\right\}
\end{aligned}$$

WEB APPENDIX B

Outline of the Asymptotic Theory

B.1 *Estimation Method*

Let $\boldsymbol{\theta}$ be the collection of parameters in the bivariate measurement error model (8) – (9) and let $\hat{\boldsymbol{\theta}}$ be its maximum likelihood estimate in the calibration sub-study. From the standard asymptotic theory

$$m^{1/2}(\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}) \rightarrow \text{Normal}(0, \Sigma_{\boldsymbol{\theta}})$$

The regression calibration predictors to be used in risk modeling, either with or without transformation, are denoted as $\mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}),\ i=1,...,n$. We denote the vector of risk parameters as $\boldsymbol{\alpha}$. Then $\boldsymbol{\alpha}$ is estimated by the solution, $\hat{\boldsymbol{\alpha}}$, of the estimating equation from the risk model, namely

$$0 = \sum_{i=1}^{n} S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\}.$$

For logistic regression, with $H(.)$ being the logistic distribution function,

$$S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\} = \left\{Y_i - H\left(\alpha_0 - \boldsymbol{\alpha}_T^t \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}) - \boldsymbol{\alpha}_Z^t \mathbf{Z}_i\right)\right\}\left(1, \mathbf{M}^t(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \mathbf{Z}_i^t\right)^t,$$

where $\boldsymbol{\alpha} = \left(\boldsymbol{\alpha}_T^t, \boldsymbol{\alpha}_Z^t\right)^t$. For Cox regression $S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\}$ is given by Wang (1999) using standard counting process notation.

B.2 *Sandwich estimator of standard errors*

Define $\boldsymbol{\alpha}_*$ as the solution to $0 = E[S\left\{Y, \mathbf{Z}, \mathbf{M}(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\alpha}_*\right\}]$. Define

$$S_{\boldsymbol{\theta}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\} = \partial S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\} / \partial \boldsymbol{\theta}^{\mathrm{t}},$$
$$S_{\boldsymbol{\alpha}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\} = \partial S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\} / \partial \boldsymbol{\alpha}^{\mathrm{t}},$$

$$\mathbf{M}_{\boldsymbol{\Theta}}(\mathbf{X}_i, \boldsymbol{\theta}) = \partial \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}) / \partial \boldsymbol{\theta}^{\mathrm{t}}.$$

Then by standard Taylor-series calculations and assuming that $n/m \rightarrow c < \infty$,

$$\begin{aligned} 0 = {} & n^{-1/2} \sum_{i=1}^{n} S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}_*\right\} \\ & + E\left[S_{\boldsymbol{\theta}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}_*\right\}\right] n^{1/2}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \\ & + E\left[S_{\boldsymbol{\alpha}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}_*\right\}\right] n^{1/2}(\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}_*) + o_p(1). \end{aligned}$$

Then, assuming that the main and calibration substudy do not overlap, we have

$$n^{1/2}(\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}_*) \rightarrow \text{Normal}(0, \Sigma_{\boldsymbol{\alpha}}),$$

where

$$\Sigma_{\boldsymbol{\alpha}} = B_{\boldsymbol{\alpha}}^{-1}\left(A + cB_{\boldsymbol{\theta}}\Sigma_{\boldsymbol{\theta}}B_{\boldsymbol{\theta}}^{t}\right)(B_{\boldsymbol{\alpha}}^{-1})^{t}.$$
$$A = COV\left[S_{\boldsymbol{\alpha}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\}\right];$$
$$B_{\boldsymbol{\theta}} = E\left[S_{\boldsymbol{\theta}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\}\right]$$
$$B_{\boldsymbol{\alpha}} = E\left[S_{\boldsymbol{\alpha}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \boldsymbol{\theta}), \boldsymbol{\alpha}\right\}\right].$$

We can estimate $\Sigma_{\boldsymbol{\alpha}}$ by making the substitution $\hat{\Sigma}_{\boldsymbol{\theta}}$ for $\Sigma_{\boldsymbol{\theta}}$ and

$$\hat{B}_{\boldsymbol{\alpha}} = n^{-1}\sum_{i=1}^{n} S_{\boldsymbol{\alpha}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\};$$
$$\hat{B}_{\boldsymbol{\theta}} = n^{-1}\sum_{i=1}^{n_P} S_{\boldsymbol{\theta}}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\};$$
$$\hat{A} = n^{-1}\sum_{i=1}^{n_P} S\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\} S^{t}\left\{Y_i, \mathbf{Z}_i, \mathbf{M}(\mathbf{X}_i, \hat{\boldsymbol{\theta}}), \hat{\boldsymbol{\alpha}}\right\}.$$

WEB APPENDIX C

Details of the Simulation Study

FFQ-reported intakes of a given food group and energy were simulated from the estimated joint distribution of the corresponding FFQ-based intakes in the NIH-AARP study. In each simulation study, we used estimated parameters of the bivariate food group and energy measurement error model (Table 2) to simulate 1000 24 HRs for each person in the main study and two 24HRs in the calibration substudy. The means of the 24HRs in the main study were taken as true usual intakes of the food group and energy, from which we calculated the corresponding densities.

For the main study, we also generated a binary disease variable according to the logistic regression model,

$$P\left\{Y_i = 1 \mid g\left(T_{Di}; \lambda_D\right), g\left(T_{Ei}; \lambda_E\right)\right\} = H\left\{\alpha_0 + \alpha_D g\left(T_{Di}; \lambda_{T_D}\right)/c_D(\lambda_{T_D}) + \alpha_E g\left(T_{Ei}; \lambda_{T_E}\right)/c_E(\lambda_{T_E})\right\}$$

where $H\left(v\right)$ denotes the logistic function and $T_{Di} = \left(T_{Fi}/T_{Ei}\right)\times 1000$ is true food group density. The

functions $c_D(\lambda_{T_D})$ and $c_E(\lambda_{T_E})$ are scaling factors to make α_D and α_E, the log odds ratios for chosen two levels of food group density and energy intakes, respectively, on the original scale, equal to some predefined values. In the first simulation study, we chose $c_D(\lambda_{T_D})$ so that α_D was equal to the log odds ratio for a change in red meat density intake from 10 to 60 g/1000 kcal per day, and in the second α_D was equal to the log odds ratio for a change in orange vegetable density intake from 0.02 to 0.10 caps/1000 kcal per day. In both studies, $c_E(\lambda_{T_E})$ was chosen so that α_E was equal to the log odds ratio for a change in energy intake from 1500 to 3000 kcal per day.

In the first simulation study, we set $\alpha_D = -0.4$, corresponding to an odds ratio of about 0.67, and in the second $\alpha_D = 0.4$, corresponding to an odds ratio of about 1.49. In both studies we set $\alpha_E = 0.2$, corresponding to an odds ratio of about 1.22. We set

$$\alpha_0 = -2.9 - \left\{\alpha_D E\left[g\left(T_{Di};\lambda_{T_D}\right)\right]/c_D(\lambda_{T_D}) + \alpha_E E\left[g\left(T_{Ei};\lambda_{T_E}\right)\right]/c_E(\lambda_{T_E})\right\},$$

corresponding to an overall probability of disease approximately equal to 0.05.

WEB REFERENCES

Wang, C. Y. (1999). Robust sandwich covariance estimation for regression calibration estimator in Cox regression with measurement error. *Letters in Probability and Statistics*, 45, 371-378.

SAS programs generating example data and implementing the proposed method

This section contains SAS macros and example programs to implement the methods described in the main text. The following programs are included:

1) bivariate.me.macros.sas – A set of SAS macros that can be used to fit a bivariate measurement error model and predict true intakes for regression calibration analysis.
2) example.simulate.data.sas – Example SAS program that creates a SAS data set of simulated red meat, energy and disease outcome data.
3) example.analyze.data.sas – Example SAS program that calls the macros in 1) to analyze the simulated data created in 2). The program fits a bivariate measurement error to the

simulated data, predicts true intakes of log red meat intake and log energy intake, and fits a logistic regression of disease outcome on log red meat and log energy using regression calibration.

```
/* SAS Program 1: example.simulate.data.sas.  */

 /*
   This program creates a SAS data set of simulated red meat, energy and disease outcome data.
   The simulated data can be analyzed using the program example.analyze.data.sas.
 */

title1 "Simulate Red Meat Data, Box-Cox Transformations in Risk Model = (0,0)";

 /* specify data library. */

libname lib "h:\documents\bivariate.me\example\data";

options center linesize=120 pagesize=42;

 /* specify parameters in measurement error model. */

%let seed1     = 8373027;
%let seed2     = 3401541;
%let n         = 100000;
%let ncalib    = 1000;

%let a1_lambda =   0.37;
%let a2_lambda =   0.27;

%let p1_beta0  =   1.48;
%let p1_beta1  =   0.22;
%let p1_beta2  =  -0.18;

%let a1_beta0  =  11.79;
%let a1_beta1  =   0.36;
%let a1_beta2  =  -0.24;

%let a2_beta0  =  17.74;
%let a2_beta1  =  -0.02;
%let a2_beta2  =   0.58;

%let var_u1    =   0.38;
%let var_u2    =   4.12;
%let var_u3    =   3.25;
%let corr_u1u2 =   0.56;
%let corr_u1u3 =   0.16;
%let corr_u2u3 =   0.30;

%let var_e1    =   1;
```

```
%let var_e2    =  18.50;
%let var_e3    =   4.87;
%let corr_e1e2 =   0;
%let corr_e1e3 =   0.22;
%let corr_e2e3 =   0.29;

%let mean_q1   =   8.74;
%let mean_q2   =  14.63;
%let var_q1    =   9.35;
%let var_q2    =   1.83;
%let corr_q1q2 =   0.58;

 /* specify parameters in risk model. */

%let alpha0    =   -2.9;
%let alpha1    =    0.40;
%let alpha2    =    0.30;

%let lambda_td =    0;
%let lambda_t2 =    0;
%let shift_t   =    0;

 /* simulate true and reported intake. */

proc iml;
  seed1  = &seed1;
  n      = &n;
  ncalib = &ncalib;

  a1_lambda = &a1_lambda;
  a2_lambda = &a2_lambda;

  p1_beta  = &p1_beta0 || &p1_beta1 || &p1_beta2;
  a1_beta  = &a1_beta0 || &a1_beta1 || &a1_beta2;
  a2_beta  = &a2_beta0 || &a2_beta1 || &a2_beta2;

  var_u1    = &var_u1;
  var_u2    = &var_u2;
  var_u3    = &var_u3;
  corr_u1u2 = &corr_u1u2;
  corr_u1u3 = &corr_u1u3;
  corr_u2u3 = &corr_u2u3;

  var_e1    = &var_e1;
  var_e2    = &var_e2;
  var_e3    = &var_e3;
  corr_e1e2 = &corr_e1e2;
  corr_e1e3 = &corr_e1e3;
  corr_e2e3 = &corr_e2e3;
```

```
mean_q    = &mean_q1 // &mean_q2;
var_q1    = &var_q1;
var_q2    = &var_q2;
corr_q1q2 = &corr_q1q2;

beta = p1_beta // a1_beta // a2_beta;

cov_q1q2 = sqrt(var_q1 * var_q2) * corr_q1q2;

cov_q    = var_q1 // cov_q1q2 // var_q2;
cov_q    = sqrsym(cov_q);
root_q   = root(cov_q)`;

cov_u1u2 = sqrt(var_u1 * var_u2) * corr_u1u2;
cov_u1u3 = sqrt(var_u1 * var_u3) * corr_u1u3;
cov_u2u3 = sqrt(var_u2 * var_u3) * corr_u2u3;

cov_u    = var_u1 // cov_u1u2 // var_u2 // cov_u1u3 // cov_u2u3 // var_u3;
cov_u    = sqrsym(cov_u);
root_u   = root(cov_u)`;

cov_e1e2 = sqrt(var_e1 * var_e2) * corr_e1e2;
cov_e1e3 = sqrt(var_e1 * var_e3) * corr_e1e3;
cov_e2e3 = sqrt(var_e2 * var_e3) * corr_e2e3;

cov_e    = var_e1 // cov_e1e2 // var_e2 // cov_e1e3 // cov_e2e3 // var_e3;
cov_e    = sqrsym(cov_e);
root_e   = root(cov_e)`;

names = {"id", "q1", "q2", "r11", "r12", "r21", "r22", "td", "t2"};
rec   = repeat(.,1,nrow(names));
create simdata from rec [colname=names];

q = repeat(.,2,1);
u = repeat(.,3,1);
y = repeat(.,1,1);

do id = 1 to n;
  call rannor(seed1, q);
  call rannor(seed1, u);

  q = mean_q + root_q * q;
  u = root_u * u;

  xbeta = beta * (1 // q);
  xbetau = xbeta + u;

  nrec = 1000;
  e = repeat(.,3,nrec);
```

```
   call rannor(seed1, e);

   e = root_e * e;
   w = repeat(xbetau,1,nrec) + e;

   mina = 0.01;
   temp = w[2,] # a1_lambda + 1;
   temp = (temp <> 0)##(1/ a1_lambda);
   temp = temp <> mina;
   r1   = (w[1,] > 0) # temp;
   temp = w[3,] # a2_lambda + 1;
   r2   = (temp <> 0)##(1 / a2_lambda);

   t1 = r1[:];
   t1 = max(t1, mina);
   t2 = r2[:];
   td = 1000 * t1 / t2;

   if (id <= ncalib) then do;
     r1 = r1[1:2];
     r2 = r2[1:2];
     end;
   else do;
     r1 = repeat(.,2,1);
     r2 = repeat(.,2,1);
     end;

 /* transform true intake to scale on which risk model is linear. */

   lambda_td = &lambda_td;
   lambda_t2 = &lambda_t2;
   shift     = &shift_t;

   if (lambda_td  = 0) then boxcox_td  = log(td + shift);
   else boxcox_td  = ((td + shift)**lambda_td - 1) / lambda_td;

   if (lambda_t2  = 0) then boxcox_t2  = log(t2);
   else boxcox_t2  = (t2**lambda_t2 - 1) / lambda_t2;

   rec = id || q` || r1` || r2` || boxcox_td || boxcox_t2;
   append from rec;
   end;

  close simdata;
  quit;

 /* simulate dependent variable in risk model. */

proc means data=simdata;
  output out=outmeans mean(td t2)=mean_td mean_t2;
```

```
 run;

proc print data=outmeans;  run;

data simdata;
 set simdata;
 if (_n_ = 1) then set outmeans(keep=mean_td mean_t2);

 /* specify unit for transformed red meat. */

 lambda_td = &lambda_td;
 lambda_t2 = &lambda_t2;
 shift     = &shift_t;

 x0 = 10;
 x1 = 60;

 if (lambda_td  = 0) then do;
   x0 = log(x0 + shift);
   x1 = log(x1 + shift);
   end;
 else do;
   x0 = ((x0 + shift)**lambda_td - 1) / lambda_td;
   x1 = ((x1 + shift)**lambda_td - 1) / lambda_td;
   end;

 unit_td  = x1 - x0;

 /* specify unit for transformed energy. */

 x0 = 1500;
 x1 = 3000;

 if (lambda_t2  = 0) then do;
   x0 = log(x0);
   x1 = log(x1);
   end;
 else do;
   x0 = (x0**lambda_t2 - 1) / lambda_t2;
   x1 = (x1**lambda_t2 - 1) / lambda_t2;
   end;

 unit_t2  = x1 - x0;

 /* generate dependent variable in risk model. */

 alpha0 = &alpha0;
 alpha1 = &alpha1;
 alpha2 = &alpha2;
```

```
  retain seed2 &seed2;

  talpha = alpha0 + alpha1 * (td-mean_td)/unit_td + alpha2 * (t2-mean_t2)/unit_t2;
  p = 1 / (1 + exp(-talpha));
  call ranbin(seed2, 1, p, y);

  keep id q1 q2 r11 r12 r21 r22 td t2 y;
  run;

proc means data=simdata;  run;

data lib.simdata;
  set simdata;
  run;


 /* SAS Program 2:  example.analyze.data.sas.  */

/*
   This program analyzes a SAS data set of simulated red meat, energy and disease outcome data.
   The program calls a set of SAS macros that fit the bivariate measurement error model and
   predict true intakes for red meat and energy,  and then fits a logistic regression model for the
   disease outcome.
   The simulated data was created using the program example.simulate.data.sas.
   The SAS macros are contained in bivariate.me.macros.sas.
 */

title1 "Analysis of Simulated Red Meat Data, Box-Cox Transformations in Risk Model = (0,0)";

 /* specify data library. */

libname lib "h:\documents\bivariate.me\example\data";

 /* include bivariate measurement error macros. */

%include "h:\documents\bivariate.me\example\bivariate.me.macros.sas";

options center linesize=120 pagesize=42;

 /* specify box-cox transformations in risk model. */

%let lambda_td = 0;
%let lambda_t2 = 0;

 /* get simulated data. */

data simdata;
  set lib.simdata;
  run;
```

```
proc means data=simdata;
  title3 "Means for Simulated Data";
  run;

title3;

 /* create calibration substudy with two records per subject. */

data calib (keep=id repeat r1 r2 q1 q2);
  set simdata;

  if (n(of r11 r12 r21 r22) = 0) then delete;

  repeat = 1;
  r1 = r11;
  r2 = r21;
  output;

  repeat = 2;
  r1 = r12;
  r2 = r22;
  output;
  run;

 /* fit bivariate measurement error model in calibration substudy. */

%Bivariate_ME (data       = calib,
               subject    = id,
               repeat     = repeat,
               response1  = r1,
               response2  = r2,
               covariates = q1 q2,
               ntitle     = 1);

data parms_b;
  set parms_b;
  min_a1 = 0;
  min_a2 = 0;
  run;

 /* save parameter estimates for me model. */

data lib.parms_b;
  set parms_b;
  run;

 /* call macro NumInt_Density to calculate predicted values in main study. */

%NumInt_Density (data      = simdata,
                 subject   = id,
```

```
                param     = parms_b,
                covars    = q1 q2,
                lambda_td = &lambda_td,
                lambda_t2 = &lambda_t2,
                ntitle    = 1);

data pred_t;
  set pred_t;
  lambda_td = &lambda_td;
  lambda_t2 = &lambda_t2;
  pred_td = pred_td_1;
  pred_t2 = pred_t2_1;
  run;

 /* save predicted values. */

data lib.pred_t;
  set pred_t;
  run;

proc means data=pred_t;
  var lambda_td lambda_t2 pred_td pred_t2;
  title3 "Mean Predicted Intakes on Transformed Scales";
  run;

title3;

 /* scale predicted intakes so that 1 unit equals the difference between */
 /* transformed values x0 and x1.                                        */

data pred_t;
  merge simdata pred_t;
    by id;

 /* specify unit for transformed red meat. */

  x0_td = 10;
  x1_td = 60;

  if (lambda_td  = 0) then do;
    x0_td_bc = log(x0_td);
    x1_td_bc = log(x1_td);
    end;
  else do;
    x0_td_bc = (x0_td**lambda_td - 1) / lambda_td;
    x1_td_bc = (x1_td**lambda_td - 1) / lambda_td;
    end;

  unit_td = x1_td_bc - x0_td_bc;
  pred_td = pred_td / unit_td;
```

```
 /* specify unit for transformed energy. */

  x0_t2 = 1500;
  x1_t2 = 3000;

  if (lambda_t2  = 0) then do;
    x0_t2_bc = log(x0_t2);
    x1_t2_bc = log(x1_t2);
    end;
  else do;
    x0_t2_bc = (x0_t2**lambda_t2 - 1) / lambda_t2;
    x1_t2_bc = (x1_t2**lambda_t2 - 1) / lambda_t2;
    end;

  unit_t2 = x1_t2_bc - x0_t2_bc;
  pred_t2 = pred_t2 / unit_t2;
  run;

proc means data=pred_t;
  var lambda_td x0_td x1_td x0_td_bc x1_td_bc unit_td pred_td
      lambda_t2 x0_t2 x1_t2 x0_t2_bc x1_t2_bc unit_t2 pred_t2;
  title3 "Mean Predicted Intakes After Standardization";
  title4 "Intakes Standardized so that One Unit Equals the Distance Between Transformed X0 and
X1";
  run;

 /* fit logistic regression risk model in main study. */

proc logistic data=pred_t;
  model y(event="1") = pred_td pred_t2;
  title3 "Logistic Regression Risk Model";
  title4 "Intakes Standardized so that One Unit Equals the Distance Between Transformed X0 and
X1";
  run;


/* SAS Program 3: bivariate.me.macros.sas.  */

/*
   This set of SAS macros can be used to fit a bivariate measurement error model and predict true
   intakes for regression calibration analysis.  The following macros are included:

   Bivariate_ME     – fits a bivariate measurement error model  This macro calls macros
                      Food_Univariate and Food_Bivariate to fit univariate and bivariate
                      measurement error models.
   Food_Univariate – fits a univariate measurement error model.  This macro is used to provide
                      initial parameter estimates to the bivariate measurement error model.
   Food_Bivariate   – fits a bivariate measurement error model.
   Rename           – renames parameters created by the Food_Univariate macro for use with
```

```
                        Food_Bivariate macro.
  Numint_Density  – calculates predicted values for the bivariate measurement error model.
 */

/*************************************************************************************
*******
 * Macro Bivariate_ME fits a bivariate measurement error model for intakes of an     *
 * episodically consumed food and energy.                                            *

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

%macro Bivariate_ME (data       = ,
                     subject    = id,
                     repeat     = repeat,
                     response1  = r1,
                     response2  = r2,
                     covariates = ,
                     ntitle     = 1);

%let ntitle1 = %eval(&ntitle+1);
%let ntitle2 = %eval(&ntitle+2);

%let nloptions = technique=newrap;

 /*** fit univariate measurement error models to get initial parameter ***/
 /*** estimates for bivariate measurement error model.                ***/

title&ntitle2 "Univariate Measurement Error Model for Episodically Consumed Food";

%let status = 1;

%Food_Univariate (data          = &data,
                  subject       = &subject,
                  repeat        = &repeat,
                  response      = &response1,
                  modeltype     = TWOPART,
                  covars_prob   = &covariates,
                  covars_amt    = &covariates,
                  lambda        = ,
                  link          = PROBIT,
                  nloptions     = &nloptions,
                  print         = N,
                  ntitle        = &ntitle2);

 /* check for convergenge. */

data _null_;
  set conv_u;
  call symput("status",trim(left(put(status, 6.))));
```

```
  run;

%if (&status ^= 0) %then %goto exit;

data parms_u1;
  set parms_u;
  run;

title&ntitle2 "Univariate Measurement Error Model for Energy";

%let status = 1;

%Food_Univariate (data          = &data,
                  subject       = &subject,
                  repeat        = &repeat,
                  response      = &response2,
                  modeltype     = ONEPART,
                  covars_prob   = ,
                  covars_amt    = &covariates,
                  lambda        = ,
                  link          = ,
                  nloptions     = &nloptions,
                  print         = N,
                  ntitle        = &ntitle2);

 /* check for convergenge. */

data _null_;
  set conv_u;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

data parms_u2;
  set parms_u;
  run;

 /* call macro Rename to rename univariate parameters to bivariate parameters. */

%Rename (data       = parms_u1,
         foodnum    = 1,
         modeltype  = TWOPART,
         covars     = &covariates);

%Rename (data       = parms_u2,
         foodnum    = 2,
         modeltype  = ONEPART,
         covars     = &covariates);
```

```
data parms_u;
  merge parms_u1 parms_u2;
  run;

 /*** call macro Food_Bivariate to fit bivariate measurement error model. ***/

title&ntitle2 "Bivariate Measurement Error Model for Episodically Consumed Food and
Energy";

%let status = 1;

%Food_Bivariate (data          = &data,
                 subject       = &subject,
                 repeat        = &repeat,
                 response1     = &response1,
                 response2     = &response2,
                 modeltype     = TWOPART,
                 covars_prob1  = &covariates,
                 covars_amt1   = &covariates,
                 covars_amt2   = &covariates,
                 lambda1       = ,
                 lambda2       = ,
                 link          = PROBIT,
                 threshold     = Y,
                 init_parms    = parms_u,
                 nloptions     = &nloptions,
                 print         = Y,
                 ntitle        = &ntitle2);

 /* check for convergenge. */

data _null_;
  set conv_b;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

%exit:

title&ntitle1;

%mend Bivariate_ME;


/*****************************************************************************
*
 *  end of macro Bivariate_ME                                    *
```

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


/*************************************************************************
**
* Macro Rename renames univariate parameters to bivariate parameters.          *
*************************************************************************
*/

%macro Rename (data       = ,
               foodnum    = ,
               modeltype  = ,
               covars     = );

%let modeltype = %upcase(&modeltype);
%if (&foodnum = 2) %then %let modeltype = ONEPART;

%let p_names  = p_intercept;
%let p_names2 = p&foodnum._intercept;
%let a_names  = a_intercept;
%let a_names2 = a&foodnum._intercept;
%let ncov     = 0;

%do %while (%scan(&covars,%eval(&ncov+1),%str( )) ^= %str());
  %let ncov = %eval(&ncov+1);
  %let xi = %scan(&covars,&ncov,%str( ));
  %let p_names  = &p_names p_&xi;
  %let p_names2 = &p_names2 p&foodnum._&xi;
  %let a_names  = &a_names a_&xi;
  %let a_names2 = &a_names2 a&foodnum._&xi;
  %end;

%let a_names  = &a_names a_LogSDe;
%let a_names2 = &a_names2 a&foodnum._LogSDe;

data &data;
  set &data;

  %if (&modeltype = TWOPART) %then %do;
    array p  (*) &p_names;
    array p2 (*) &p_names2;

    do i = 1 to dim(p);
      p2(i) = p(i);
      end;
    %end;

  array a  (*) &a_names;
```

```
 array a2 (*) &a_names2;

 do i = 1 to dim(a);
  a2(i) = a(i);
  end;

 LogSDu3 = LogSDu2;
 a&foodnum._lambda = a_lambda;

 keep %if (&modeltype = TWOPART) %then &p_names2 &a_names2 LogSDu1 LogSDu2
corr_u1u2;
     %else %if (&foodnum = 1) %then &a_names2 LogSDu2;
     %else %if (&foodnum = 2) %then &a_names2 LogSDu3;
     a&foodnum._lambda;
 run;

%mend rename;


/*************************************************************************
*
 * End of macro Rename                                          *

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


 /*************************************************************************
 *                                                                       *
 * SAS macro Food_Univariate fits a univariate model for a food/nutrient.    *
 * The food/nutrient can be episodically consumed or consumed every day.     *
 *                                                                       *
 * Model for episodically consumed foods/nutrients (two-part model):         *
 * For episodically consumed foods/nutrients, the macro fits a two-part      *
 * nonlinear mixed model, where the first part is the probability to         *
 * consume and the second part is the amount consumed on a consumption day.  *
 * The model allows for covariates in each part, includes a random effect    *
 * for each part, and allows the random effects to be correlated.           *
 *                                                                       *
 * Model for foods/nutrients consumed every day (one-part model):            *
 * For foods/nutrients consumed every day, the macro fits a one-part         *
 * nonlinear mixed model of the amount consumed (the probability to consume  *
 * is assumed to be 1). The model allows for covariates and includes a       *
 * random effect.                                                        *
 *                                                                       *
 * For a food/nutrient that is consumed nearly every day by nearly everyone, *
 * so that the number of zero values is small, it may be preferable to use   *
 * the one-part (consumed every day) model, since the two-part model may     *
 * have trouble modeling the probability to consume in such a situation.     *
 *                                                                       *
```

```
* Note, however, that the one-part model requires all responses to be       *
* greater than zero (zero values are treated as missing values).            *
* Before fitting the one-part model to a food/nutrient that has some zero   *
* values, replace the zero values with a small positive value, such as      *
* half the smallest observed nonzero value.                                 *
*                                                                           *
* The macro calls the NLMixed procedure to fit the model.                   *
*                                                                           *
*****************************************************************************
*                                                                           *
* Macro Parameters:                                                         *
*                                                                           *
*   Required Parameters:                                                    *
*     data         = name of SAS data set containing the data to be       *
*                    analyzed. The data set has multiple observations     *
*                    for each subject, one for each reptition of the      *
*                    24-hour recall (or other dietary instrument).        *
*     subject      = name of the variable that uniquely identifies each   *
*                    subject (i.e., ID variable).                         *
*     repeat       = name of the variable that indexes repeated           *
*                    observations for each subject.                       *
*     response     = name of the food/nutrient variable to be modeled     *
*                    (24-hour recall variable for the food/nutrient).     *
*     modeltype    = model for food/nutrient:                             *
*                    to fit the two-part (epsisodic) model, specify       *
*                      modeltype = TWOPART                                *
*                    to fit the one-part (every day) model, specify       *
*                      modeltype = ONEPART                                *
*                                                                         *
*   Optional Parameters:                                                  *
*     covars_prob  = list of variables that are covariates in the         *
*                    probability part of the two-part model.              *
*                    if modeltype=ONEPART, then covars_prob is ignored.   *
*     covars_amt   = list of variables that are covariates in the         *
*                    one-part model or the amount part of the             *
*                    two-part model.                                      *
*     link         = link function for the probability part of the two-   *
*                    part model. to fit a logistic model, specify         *
*                      link = logit                                       *
*                    to fit a probit model, specify                       *
*                      link = probit                                      *
*                    by default, link = probit.                           *
*                    if modeltype = ONEPART, then link is ignored.        *
*     lambda       = Box-Cox transformation parameter for the amount      *
*                    part of the model. If lambda is not specified,       *
*                    then it is estimated as part of the model.           *
*     var_u1       = variance of the random effect in the probability     *
*                    part of the two-part model.                          *
*                    If var_u1 is not specified, then it is estimated     *
*                    as part of the model.                                *
```

```
*                 if modeltype = ONEPART, then var_u1 is ignored.     *
*     var_u2      = variance of the random effect in the one-part model *
*                 or the amount part of the two-part model.           *
*                 If var_u2 is not specified, then it is estimated    *
*                 as part of the model.                               *
*     indep_u     = Y if random effects u1 and u2 are independent.     *
*                 = N if random effects u1 and u2 are dependent.      *
*                 by default, indep_u = N.                            *
*                 if modeltype = ONEPART, then indep_u is ignored.    *
*     replicate_var = name of the sampling weight variable if the data    *
*                 is from a complex survey with weights.              *
*                 by default, the macro performs an unweighted        *
*                 analysis (assumes a simple random sample).          *
*     nloptions   = options for the NLMixed procedure that are        *
*                 appended to the PROC NLMIXED statement, e.g.,       *
*                   nloptions = technique=newrap maxiter=200,         *
*     init_parms  = name of SAS data set that contains initial        *
*                 parameter estimates. See the description of output  *
*                 data set parms_u (below) for further information.   *
*                 if init_parms is not specified, then the macro      *
*                 calculates initial parameter estimates.             *
*     print       = Y to print the output from the model.             *
*                 = N to supress printing the output from the model.  *
*                 = V (verbose) to print extra output.                *
*                 by default, print = Y.                              *
*     ntitle      = number of titles defined by the user.             *
*                 by default, ntitle = 2.                             *
*                                                                     *
***********************************************************************
*                                                                     *
* Output Data Sets:                                                   *
*                                                                     *
*  parms_u = data set containing parameter estimates for the model.   *
*          parms_u contains the following variables:                  *
*                                                                     *
*             A_Intercept = intercept in the amount part of the model.  *
*             A_varname   = regression slope for covariate "varname"    *
*                         in the amount part of the model.            *
*             A_LogSDe    = Log(Sqrt(Var_e))                          *
*             LogSDu2     = Log(Sqrt(Var_u2))                         *
*             Var_e       = variance of the within-person error in the  *
*                         amount part of the model.                   *
*             Var_u2      = variance of the random effect in the      *
*                         amount part of the model.                   *
*                                                                     *
*          if fitting the two-part model, then parms_u also contains  *
*          the following variables:                                   *
*                                                                     *
*             P_Intercept = intercept in the prob. part of the model.   *
*             P_varname   = regression slope for covariate "varname"    *
```

```
*                    in the prob. part of the model.          *
*          LogSDu1    = Log(Sqrt(Var_u2))                     *
*          z_u1u2     = Fisher transformation of Corr_u1u2:   *
*                     z = ln[(1+corr)/(1-corr)] / 2           *
*          Var_u1     = variance of the random effect in the  *
*                     prob. part of the model.                *
*          Cov_u1u2   = covariance of random effects u1 and u2.*
*          Corr_u1u2  = correlation of random effects u1 and u2.*
*                                                             *
*        note: if specifying initial parameter estimates using the *
*          init_parms option, the init_parms data set should have *
*          the same variables as parms_u, except it should not *
*          include var_e, var_u2, var_u1, cov_u1u2 or corr_u1u2 *
*          (these are derived parameters, i.e., functions of the *
*           other parameters).                                *
*                                                             *
*  pred_x_u = data set containing predicted values for the model.  *
*        pred_x_u contains all the variables in the input data set, *
*        plus the following variable:                         *
*                                                             *
*          pred_x_a = predicted mean amount on consumption day. *
*                                                             *
*        if fitting the two-part model, then pred_x_u also contains *
*        the following variable:                              *
*                                                             *
*          pred_x_p = predicted probability of consumption.   *
*                                                             *
*************************************************************************/

%macro Food_Univariate (data          = ,
                        subject       = ,
                        repeat        = ,
                        response      = ,
                        modeltype     = TWOPART,
                        covars_prob   = ,
                        covars_amt    = ,
                        link          = PROBIT,
                        lambda        = ,
                        var_u1        = ,
                        var_u2        = ,
                        indep_u       = N,
                        replicate_var = ,
                        nloptions     = ,
                        init_parms    = ,
                        print         = Y,
                        ntitle        = 2);

%let modeltype  = %upcase(&modeltype);
%let link       = %upcase(&link);
%let indep_u    = %upcase(%substr(&indep_u,1,1));
```

```
%let print      = %upcase(%substr(&print,1,1));

%if (&link      ^= LOGIT)  %then %let link = PROBIT;
%if (&modeltype ^= ONEPART) %then %let modeltype = TWOPART;

%if (&modeltype = ONEPART) %then %do;
  %let link = NONE;
  %let var_u1 = 0;
  %end;
%if (&var_u1 = 0 | &var_u2 = 0) %then %let indep_u = Y;

%if (&print = V) %then %let nloptions = &nloptions itdetails;

%if (&replicate_var ^= %str()) %then %do;
  footnote "Note: Standard Errors are not valid if data are not from a simple random sample";
  %end;

proc sort data=&data;  by &subject &repeat;  run;

data parms_u;
  run;

data pred_x_u;
  run;

data conv_u;
  status = 1;
  run;

 /* determine number of covariates in each part of the model. */
 /* get initial parameter estimate for box-cox transformation parameter lambda. */

data _null_;
  set &data;

  %if (&covars_prob = %str()) %then nx_p = 0%str(;);
  %else %do;
     array _p (*) &covars_prob;
     nx_p = dim(_p);
     %end;

  %if (&covars_amt = %str()) %then nx_a = 0%str(;);
  %else %do;
     array _a (*) &covars_amt;
     nx_a = dim(_a);
     %end;

  call symput("nx_p",trim(left(put(nx_p, 3.))));
  call symput("nx_a",trim(left(put(nx_a, 3.))));
  run;
```

```
 /* create indicator variable _consume that equals 0 if response = 0, 1 otherwise.          */

data _data;
  set &data;

  if (&response = 0) then _consume = 0;
  if (&response > 0) then _consume = 1;
  run;

 /*** calculate initial parameter estimates for the probability part of the model. ***/

%if (&init_parms = %str() & &modeltype ^= ONEPART) %then %do;

%if (&link = LOGIT) %then title%eval(&ntitle+1) "Calculate Initial Estimates for Probability
Model (Link = Logit)" %str(;);
%else                   title%eval(&ntitle+1) "Calculate Initial Estimates for Probability Model (Link
= Probit)" %str(;);

 /* fit probability model without random effects (proc logistic). */

data _conv_p;
  status = 1;
  run;

%if (&print ^= V) %then ods exclude all %str(;);
%if (&print = V) %then ods exclude oddsratios %str(;);

proc logistic data=_data namelen=50;
  model _consume(event="1") = &covars_prob / link=&link;
  ods output ParameterEstimates=_init_p;
  ods output ConvergenceStatus=_conv_p;
  run;

ods exclude none;

 /* check for convergenge. */

data _null_;
  set _conv_p;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

 /* save parameters. */

proc transpose data=_init_p out=_init_p(drop=_name_);
  id variable;
  var estimate;
```

```
  run;

data _init_p;
  set _init_p;

  rename intercept = P_Intercept
    %do i = 1 %to &nx_p;
      %let xi = %scan(&covars_prob,&i,%str( ));
      &xi = P_&xi
      %end;
    ;
  run;

 /* fit probability model with random effects (proc nlmixed). */

%if (&var_u1 = %str()) %then %do;

data _init_p1;
  set _init_p;
  LogSDu1 = 0;
  run;

data _conv_p;
  status = 1;
  run;

%if (&print ^= V) %then ods exclude all %str(;);

proc nlmixed data=_data &nloptions;
  parms /data=_init_p1;

  %if (&replicate_var ^= %str()) %then replicate &replicate_var%str(;);    /* weight variable */

 /* reparameterize the variance/covariance of u. */

  VAR_U1 = EXP(2*LogSDu1);

  x_p = P_Intercept;
  %do i = 1 %to &nx_p;
    %let xi = %scan(&covars_prob,&i,%str( ));
    x_p = x_p + P_&xi * &xi;
    %end;
  xu_p = x_p + u1;

 /* probit or logit link. */

  %if (&link = LOGIT) %then _p = 1 /(1 + exp(-xu_p)) %str(;);
  %else _p = probnorm(xu_p) %str(;);

  model _consume ~ binary(_p);
```

```
  random u1 ~ normal(0,var_u1) subject=&subject;

  estimate "VAR_U1" VAR_U1;

  ods output ParameterEstimates=_init_p;
  ods output ConvergenceStatus=_conv_p;
  run;

%if (&print ^= V) %then ods exclude none %str(;);

 /* check for convergenge. */

data _null_;
  set _conv_p;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

 /* save parameters. */

proc transpose data=_init_p out=_init_p(drop=_name_);
  id parameter;
  var estimate;
  run;

%end;  /* %if (&var_u1 = %str()) %then %do; */

title%eval(&ntitle+1);

%end;  /* %if (&init_parms = %str() & &modeltype ^= ONEPART) %then %do */

 /*** calculate initial parameter estimates for the amount part of the model (proc transreg). ***/

%if (&init_parms = %str()) %then %do;

%if (&print ^= V) %then ods exclude all %str(;);
ods output boxcox=_boxcox;
ods output coef=_coef;
ods output fitstatistics=_fitstatistics;

proc transreg data=_data(where=(&response > 0));
  %if (&lambda = %str()) %then %do;
    model boxcox(&response / lambda=0 to 1 by 0.1) = identity(&covars_amt) / ss2;
    %end;
  %else %do;
    model boxcox(&response / lambda=&lambda) = identity(&covars_amt) / ss2;
    %end;
  title%eval(&ntitle+1) "Calculate Initial Parameter Estimates for Amount Model";
```

```
  run;

title%eval(&ntitle+1);

ods exclude none;

 /* save parameters. */

data _boxcox (keep=_lambda);
  set _boxcox end=eof;
  retain _lambda _maxlike;
  if (_n_ = 1) then _maxlike = loglike;
  if (loglike >= _maxlike) then do;
    _lambda = lambda;
    _maxlike = loglike;
    end;
  if (eof) then output;
  run;

data _fitstatistics (keep=rmse);
  set _fitstatistics (rename=(value1=rmse));
  if (upcase(label1) = "ROOT MSE") then output;
  run;

data _coef (keep=_varname coefficient);
  set _coef;
  if (upcase(variable) = "INTERCEPT") then _varname = variable;
  else _varname = scan(variable,2,"()");
  output;
  run;

proc transpose data=_coef out=_coef(drop=_name_);
  id _varname;
  var coefficient;
  run;

data _init_a (drop=rmse);
  merge _coef _boxcox _fitstatistics;

  %if (&var_u2 = %str(0)) %then %do;
    var_e = rmse;
    %end;
  %else %do;
    var_u2 = rmse**2 / 2;
    var_e  = rmse**2 / 2;
    %end;

  rename _lambda    = A_Lambda
         intercept = A_Intercept
        %do i = 1 %to &nx_a;
```

```
        %let xi = %scan(&covars_amt,&i,%str( ));
        &xi = A_&xi
      %end;
   ;
  run;

 /*** combine initial parameter estimates from probability and amount parts of the model. ***/

data _init_parms (drop=var_e %if (&var_u2 ^= %str(0)) %then var_u2; %if (&lambda ^=
%str()) %then a_lambda;);
  merge %if (&modeltype ^= ONEPART) %then _init_p; _init_a;

  A_LogSDe = log(var_e) / 2;
  %if (&var_u2 = %str()) %then LogSDu2 = log(var_u2) / 2 %str(;);
  %if (&indep_u ^= Y) %then Z_u1u2 = 0 %str(;);
  run;

%let init_parms = _init_parms;

%end;  /* %if (&init_parms = %str()) %then %do */

%if (&print = V) %then %do;
  proc print data=&init_parms noobs;
    title%eval(&ntitle+1) "Initial Parameter Estimates for Food/Nutrient Model";
    run;
  %end;

 /*** fit two-part (episodically consumed) or one-part (consumed every day) model (proc
nlmixed). ***/

     %if (&link = NONE)  %then title%eval(&ntitle+1) "Food/Nutrient Model"%str(;);
%else %if (&link = LOGIT) %then title%eval(&ntitle+1) "Food/Nutrient Model (Link =
Logit)"%str(;);
%else                     title%eval(&ntitle+1) "Food/Nutrient Model (Link = Probit)"%str(;);

data conv_u;
  status = 1;
  run;

%if (&print = N) %then ods exclude all %str(;);

proc nlmixed  data=&data &nloptions;
  parms / data=&init_parms;

  %if (&replicate_var ^= %str()) %then replicate &replicate_var%str(;);    /* survey weight
variable */

  %if (&lambda ^= %str()) %then A_Lambda = &lambda%str(;);

  %if (&var_u1 = 0) %then u1 = 0%str(;);
```

```
 %if (&var_u2 = 0) %then u2 = 0%str(;);

/* reparameterize the variance of e. */

 VAR_E = EXP(2 * A_LogSDe);

/* reparameterize the variance/covariance of u. */

 %if (&var_u1 ^= 0) %then %do;
  %if (&var_u1 = %str()) %then VAR_U1 = exp(2 * LogSDu1)%str(;);
  %else VAR_U1 = &var_u1%str(;);
  %end;
 %if (&var_u2 ^= 0) %then %do;
  %if (&var_u2 = %str()) %then VAR_U2 = exp(2 * LogSDu2)%str(;);
  %else VAR_U2 = &var_u2%str(;);
  %end;
 %if (&var_u1 ^= 0 & &var_u2 ^= 0) %then %do;
  %if (&indep_u = Y) %then %do;
    CORR_U1U2 = 0;
    COV_U1U2  = 0;
    %end;
  %else %do;
    CORR_U1U2 =(EXP(2 * Z_u1u2) - 1) / (EXP(2 * Z_u1u2) + 1);
    COV_U1U2  = CORR_U1U2 * SQRT(VAR_U1 * VAR_U2);
    %end;
  %end;

/* likelihood for probability of consumption. */

 %if (&modeltype = ONEPART) %then %do;
  ll_b = 0;
  %end;

 %else %do;
  x_p = P_Intercept;
  %do i = 1 %to &nx_p;
    %let xi = %scan(&covars_prob,&i,%str( ));
    x_p = x_p + P_&xi * &xi;
    %end;
  xu_p = x_p + u1;

/* probit or logit link. */

  %if (&link = PROBIT) %then _p = probnorm(xu_p) %str(;);
  %else _p = 1 /(1 + exp(-xu_p)) %str(;);

  if (&response > 0) then _y = 1;
  else if (&response = 0) then _y = 0;
  else _y = .;
  if (_y ^= .) then ll_b = log((1-_p)**(1-_y)) + log(_p**(_y));
```

```
  else ll_b = .;
  %end;

/* likelihood for amount consumed on consumption day. */

 x_a = A_Intercept;
 %do i = 1 %to &nx_a;
  %let xi = %scan(&covars_amt,&i,%str( ));
  x_a = x_a + A_&xi * &xi;
  %end;
 xu_a = x_a + u2;

 pi = arcos(-1);

 if (&response > 0) then do;
  if (A_Lambda = 0) then boxcoxy = log(&response);
  else boxcoxy = (&response**A_Lambda - 1) / A_Lambda;
  ll_n = -log(sqrt(2 * pi * VAR_E)) - (boxcoxy - xu_a)**2 / (2 * VAR_E) +
        (A_Lambda - 1) * log(&response);
  end;

 %if (&modeltype ^= ONEPART) %then %do;
  else if (&response = 0) then do;
    ll_n = 0;
    end;
  %end;

 else ll_n = .;

 ll = ll_b + ll_n;

 model &response ~ general(ll);

/* specify random effects. */

 %if (&var_u1 ^= 0 | &var_u2 ^= 0) %then %do;
       %if (&var_u2 = 0) %then random u1 ~ normal(0, VAR_U1);
  %else %if (&var_u1 = 0) %then random u2 ~ normal(0, VAR_U2);
  %else                        random u1 u2 ~ normal([0,0],[VAR_U1, COV_U1U2, VAR_U2]);
                               subject=&subject /* out=pred_u */;
  %end;

/* estimate additional parameters. */

 %if (&var_u1 ^= 0) %then estimate "VAR_U1" VAR_U1 %str(;);
 %if (&var_u2 ^= 0) %then estimate "VAR_U2" VAR_U2 %str(;);
 %if (&indep_u ^= Y) %then %do;
  estimate "COV_U1U2" COV_U1U2;
  estimate "CORR_U1U2" CORR_U1U2;
  %end;
```

```
  estimate "VAR_E" VAR_E;

  %if (&modeltype ^= ONEPART) %then predict x_p out=pred_x_p%str(;);
  predict x_a out=pred_x_a;
  predict xu_a out=pred_xu_a;

  ods output ParameterEstimates=parms_u;
  ods output AdditionalEstimates=adparms_u;
  ods output FitStatistics=fit_u;
  ods output ConvergenceStatus=conv_u;
  run;

title%eval(&ntitle+1);

%if (&print = N) %then ods exclude none %str(;);

 /* check for convergenge. */

data _null_;
  set conv_u;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

 /*** save parameter estimates and predicted values. ***/

proc transpose data=parms_u out=parms_u(drop=_name_);
  id parameter;
  var estimate;
  run;

proc transpose data=adparms_u out=adparms_u(drop=_name_);
   id label;
   var estimate;
   run;

data parms_u;
  merge parms_u adparms_u;

 /* add fixed parameters. */

  %if (&lambda ^= %str()) %then A_Lambda = &lambda%str(;);

  %if (&modeltype ^= ONEPART & &var_u1 ^= %str()) %then VAR_U1 = &var_u1%str(;);
  %if (&var_u2 ^= %str()) %then VAR_U2 = &var_u2%str(;);
  %if (&modeltype ^= ONEPART & &indep_u = Y) %then %do;
    COV_U1U2  = 0;
    CORR_U1U2 = 0;
```

```
   %end;
  run;

data pred_x_u;
  merge &data
       %if (&modeltype ^= ONEPART) %then pred_x_p(keep=&subject &repeat pred
rename=(pred=pred_x_p));
                                pred_x_a(keep=&subject &repeat pred rename=(pred=pred_x_a));
    by &subject &repeat;

  label %if (&modeltype ^= ONEPART) %then pred_x_p = " "; pred_x_a = " ";
  run;

 /*** delete unneeded data sets. */

proc datasets lib=work nolist;
  delete _data _init_a _init_parms adparms_u pred_x_a
       %if (&modeltype ^= ONEPART) %then _init_p _conv_p pred_x_p; %str(;)
  run;
  quit;

%exit:

title%eval(&ntitle+1);

%if (&replicate_var ^= %str()) %then footnote %str(;);

%mend Food_Univariate;


/*************************************************************************
*
 * End of macro Food_Univariate                                          *

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


 /*************************************************************************
 *                                                                       *
 * SAS macro Food_Bivariate fits a bivariate model for two foods/nutrients.  *
 * The first food/nutrient can be episodically consumed or consumed every    *
 * day, while the second food/nutrient is assumed to be consumed every day.  *
 *                                                                       *
 * Model for episodically consumed foods/nutrients (two-part model):         *
 * For episodically consumed foods/nutrients, the macro fits a two-part      *
 * nonlinear mixed model, where the first part is the probability to         *
 * consume and the second part is the amount consumed on a consumption day.  *
 * The model allows for covariates in each part, includes a random effect    *
 * for each part, and allows the random effects to be correlated.            *
```

```
*                                                                    *
* Model for foods/nutrients consumed every day (one-part model):     *
* For foods/nutrients consumed every day, the macro fits a one-part  *
* nonlinear mixed model of the amount consumed (the probability to consume *
* is assumed to be 1). The model allows for covariates and includes a *
* random effect.                                                     *
*                                                                    *
* For a food/nutrient that is consumed nearly every day by nearly everyone, *
* so that the number of zero values is small, it may be preferable to use *
* the one-part (consumed every day) model, since the two-part model may *
* have trouble modeling the probability to consume in such a situation. *
*                                                                    *
* Note, however, that the one-part model requires all responses to be *
* greater than zero (zero values are treated as missing values).     *
* Before fitting the one-part model to a food/nutrient that has some zero *
* values, replace the zero values with a small positive value, such as *
* half the smallest observed nonzero value.                          *
*                                                                    *
* Note: Initial parameter estimates must be supplied by the user.    *
* They can be estimated using SAS macro Food_Univariate.             *
*                                                                    *
* The macro calls the NLMixed procedure to fit the model.            *
*                                                                    *
**********************************************************************
*                                                                    *
* Macro Parameters:                                                  *
*                                                                    *
*   Required Parameters:                                             *
*     data         = name of SAS data set containing the data to be  *
*                    analyzed. The data set has multiple observations *
*                    for each subject, one for each reptition of the *
*                    24-hour recall (or other dietary instrument).   *
*     subject      = name of the variable that uniquely identifies each *
*                    subject (i.e., ID variable).                    *
*     repeat       = name of the variable that indexes repeated      *
*                    observations for each subject.                  *
*     response1    = name of first food/nutrient variable to be modeled *
*                    (24-hour recall variable for first food/nutrient). *
*     response2    = name of second food/nutrient variable to be modeled *
*                    (24-hour recall variable for second food/nutrient). *
*     modeltype    = model for first food/nutrient:                  *
*                    to fit the two-part (epsisodic) model, specify  *
*                      modeltype = TWOPART                           *
*                    to fit the one-part (every day) model, specify  *
*                      modeltype = ONEPART                           *
*     init_parms   = name of SAS data set that contains initial      *
*                    parameter estimates. See the description of output *
*                    data set parms_b (below) for further information. *
*                                                                    *
*   Optional Parameters:                                             *
```

```
*     covars_prob1  = list of variables that are covariates in the       *
*                     probability part of the two-part model for the     *
*                     first food/nutrient.                               *
*                     if modeltype = ONEPART, then covars_prob is ignored.*
*     covars_amt1   = list of variables that are covariates in the       *
*                     one-part model or the amount part of the           *
*                     two-part model for the first food/nutrient.        *
*     covars_amt2   = list of variables that are covariates in the       *
*                     one-part model for the second food/nutrient        *
*     link          = link function for the probability part of the two- *
*                     part model for the first food/nutrient.            *
*                     to fit a logistic model, specify                   *
*                       link = logit                                     *
*                     to fit a probit model, specify                     *
*                       link = probit                                    *
*                     by default, link = probit.                         *
*                     if modeltype = ONEPART, then link is ignored.      *
*     lambda1       = Box-Cox transformation parameter for the first     *
*                     food/nutrient. If lambda1 is not specified, then   *
*                     it is estimated as part of the model.              *
*     lambda2       = Box-Cox transformation parameter for the second    *
*                     food/nutrient. If lambda2 is not specified, then   *
*                     it is estimated as part of the model.              *
*     var_u1        = variance of the random effect in the probability   *
*                     part of the model for the first food/nutrient.     *
*                     If var_u1 is not specified, then it is estimated   *
*                     as part of the model.                              *
*                     if modeltype = ONEPART, then var_u1 is ignored.    *
*     var_u2        = variance of the random effect in the amount        *
*                     part of the model for the first food/nutrient.     *
*                     If var_u2 is not specified, then it is estimated   *
*                     as part of the model.                              *
*     var_u3        = variance of the random effect in the amount        *
*                     part of the model for the second food/nutrient.    *
*                     If var_u3 is not specified, then it is estimated   *
*                     as part of the model.                              *
*     indep_u1      = Y if random effect u1 is independent of u2 and u3. *
*                   = N otherwise. by default, indep_u1 = N.             *
*                     if modeltype = ONEPART, then indep_u1 is ignored.  *
*     indep_u2      = Y if random effect u2 is independent of u1 and u3. *
*                   = N otherwise. by default, indep_u2 = N.             *
*     indep_u3      = Y if random effect u3 is independent of u1 and u2. *
*                   = N otherwise. by default, indep_u3 = N.             *
*     threshold     = Y to fit a latent variable threshold model.        *
*                   = N otherwise. by default, threshold = Y.            *
*                     if threshold = Y, then the probit model is fit.    *
*                     if modeltype = ONEPART, then threshold is ignored. *
*     replicate_var = name of the sampling weight variable if the data   *
*                     is from a complex survey with weights.             *
*                     by default, the macro performs an unweighted       *
```

```
*                 analysis (assumes a simple random sample).          *
*     nloptions     = options for the NLMixed procedure that are        *
*                 appended to the PROC NLMIXED statement, e.g.,       *
*                  nloptions = technique=newrap maxiter=200,          *
*     print         = Y to print the output from the model.           *
*                 = N to supress printing the output from the model.  *
*                 = V (verbose) to print extra output.                *
*                  by default, print = Y.                             *
*     ntitle        = number of titles defined by the user.           *
*                  by default, ntitle = 2.                            *
*                                                                     *
***********************************************************************
*                                                                     *
* Output Data Sets:                                                   *
*                                                                     *
*   parms_b = data set containing parameter estimates for the model.  *
*           parms_b contains the following variables:                 *
*                                                                     *
*             A1_Intercept = intercept in the amount part of the model  *
*                          for the first food/nutrient.               *
*             A1_varname   = regression slope for covariate "varname"   *
*                          in the amount part of the model for the    *
*                          first food/nutrient.                       *
*             A2_Intercept = intercept for the second food/nutrient.    *
*             A2_varname   = regression slope for covariate "varname"   *
*                          for the second food/nutrient.              *
*             A1_LogSDe    = Log(Sqrt(Var_e2))                          *
*             A2_LogSDe    = Log(Sqrt(Var_e3))                          *
*             z_e2e3       = Fisher transformation of Corr_e2e3:        *
*                            z = ln[(1+corr)/(1-corr)] / 2              *
*             Var_e2       = variance of within-person error e2 (amount *
*                          part of model for first food/nutrient).    *
*             Var_e3       = variance of within-person error e3 (second *
*                          food/nutrient).                            *
*             Var_u2       = variance of random effect u2 (amount part  *
*                          of model for first food/nutrient).         *
*             Var_u3       = variance of random effect u3 (second       *
*                          food/nutrient).                            *
*                          of the model for the second food).         *
*             Cov_e2e3     = covariance of random errors e2 and e3.     *
*             Corr_e2e3    = correlation of random errors e2 and e3.    *
*             Cov_u2u3     = covariance of random effects u2 and u3.    *
*             Corr_u2u3    = correlation of random effects u2 and u3.   *
*                                                                     *
*           if fitting the two-part model for the first food/nutrient,    *
*           then parms_b also contains the following variables:           *
*                                                                     *
*             P1_Intercept = intercept in the prob. part of the model   *
*                          for the first food/nutrient.               *
*             P1_varname   = regression slope for covariate "varname"   *
```

```
*                    in the prob. part of the model for the     *
*                    first food/nutrient.                       *
*            Var_u1      = variance of random effect u1 (prob. part   *
*                    of model for first food/nutrient).         *
*            Cov_u1u2    = covariance of random effects u1 and u2.    *
*            Cov_u1u3    = covariance of random effects u1 and u3.    *
*            Corr_u1u2   = correlation of random effects u1 and u2.   *
*            Corr_u1u3   = correlation of random effects u1 and u3.   *
*                                                               *
*          note: initial parameter estimates must be supplied by the     *
*              user using the init_parms option.                *
*              the user-supplied data set will have the same variables *
*              as data set parms_b, except it should not include the   *
*              following variables:                             *
*                  z_e2e3                                       *
*                  Cov_e2e3                                     *
*                  Corr_e2e3                                    *
*                  Cov_u1u2 Cov_u1u3 Cov_u2u3                   *
*                  Corr_u1u3 Corr_u2u3                          *
*              All the necessary initial parameter estimates can be    *
*              estimated using the SAS macro Food_Univariate.          *
*                                                               *
*   pred_x_b = data set containing predicted values for the model.         *
*            pred_x_b contains all the variables in the input data set,   *
*            plus the following variables:                      *
*                                                               *
*              pred_x_a1 = predicted mean amount on consumption day for  *
*                      the first food/nutrient.                 *
*              pred_x_a2 = predicted mean amount for the second         *
*                      food/nutrient.                           *
*                                                               *
*          if fitting the two-part model for the first food/nutrient,    *
*          then pred_x_b also contains the following variable:          *
*                                                               *
*              pred_x_p1 = predicted probability of consumption for      *
*                      the first food/nutrient.                 *
*                                                               *
*****************************************************************************/

%macro Food_Bivariate (data          = ,
                       subject       = ,
                       repeat        = ,
                       response1     = ,
                       response2     = ,
                       modeltype     = TWOPART,
                       init_parms    = ,
                       covars_prob1  = ,
                       covars_amt1   = ,
                       covars_amt2   = ,
                       link          = PROBIT,
```

```
                lambda1       = ,
                lambda2       = ,
                var_u1        = ,
                var_u2        = ,
                var_u3        = ,
                corr_u1u2     = ,
                corr_u1u3     = ,
                corr_u2u3     = ,
                threshold     = Y,
                replicate_var  = ,
                init_cov_zero  = Y,
                cov_u_sds     = Y,
                nloptions     = ,
                print         = Y,
                ntitle        = 2);

%let print      = %upcase(%substr(&print,1,1));
%let modeltype  = %upcase(&modeltype);
%let link       = %upcase(&link);
%let init_cov_zero = %upcase(%substr(&init_cov_zero,1,1));
%let threshold  = %upcase(%substr(&threshold,1,1));

%if (&link      ^= LOGIT)   %then %let link = PROBIT;
%if (&link       = LOGIT)   %then %let threshold = N;
%if (&modeltype ^= ONEPART) %then %let modeltype = TWOPART;

%if (&modeltype = ONEPART) %then %do;
  %let link = NONE;
  %let var_u1 = 0;
  %let threshold = N;
  %end;

%if (&print = V) %then %let nloptions = &nloptions itdetails;

%if (&replicate_var ^= %str()) %then %do;
  footnote "Note: Standard Errors are not valid if data are not from a simple random sample";
  %end;

proc sort data=&data;  by &subject &repeat;  run;

 /*** determine number and order of random effects u. ***/

%let dim_u   = 0;
%let index_u = ;

%if (&modeltype ^= ONEPART & &var_u1 ^= 0) %then %do;
  %let dim_u   = %eval(&dim_u + 1);
  %let index_u = &index_u 1;
  %end;
```

```
%if (&var_u2 ^= 0) %then %do;
  %let dim_u   = %eval(&dim_u + 1);
  %let index_u = &index_u 2;
  %end;

%let dim_u_f1 = &dim_u;

%if (&var_u3 ^= 0) %then %do;
  %let dim_u   = %eval(&dim_u + 1);
  %let index_u = &index_u 3;
  %end;

 /*** determine number of covariates in each part of the model. ***/

data _null_;
  set &data;

  %if (&covars_prob1 = %str()) %then nx_p1 = 0%str(;);
  %else %do;
    array _p1 (*) &covars_prob1;
    nx_p1 = dim(_p1);
    %end;

  %if (&covars_amt1 = %str()) %then nx_a1 = 0%str(;);
  %else %do;
    array _a1 (*) &covars_amt1;
    nx_a1 = dim(_a1);
    %end;

  %if (&covars_amt2 = %str()) %then nx_a2 = 0%str(;);
  %else %do;
    array _a2 (*) &covars_amt2;
    nx_a2 = dim(_a2);
    %end;

  call symput("nx_p1",trim(left(put(nx_p1, 3.))));
  call symput("nx_a1",trim(left(put(nx_a1, 3.))));
  call symput("nx_a2",trim(left(put(nx_a2, 3.))));
  run;

 /*** calculate initial estimates of covariance parameters ***/
 /*** for random effects u and within-person errors e.     ***/

%if (&init_cov_zero = Y) %then %do;

proc iml;

  z_e2e3  = 0;
  z_gamma = 0;
  z_theta = 0;
```

```
 initcov = %if (&modeltype = ONEPART | &threshold ^= Y) %then z_e2e3;
         %else  z_gamma || z_theta; %str(;)

 names   = {%if (&modeltype = ONEPART | &threshold ^= Y) %then "Z_E2E3";
          %else "Z_GAMMA" "Z_THETA"; };

 %if (&cov_u_sds = Y & &dim_u > 0) %then %do;
   use &init_parms;
    read all var{%do i = 1 %to &dim_u;
                 %let ii = %scan(&index_u,&i,%str( ));
                 LogSDu&ii
                 %end;
               } into logsd_u;
   %if (&modeltype ^= ONEPART) %then %do;
    read all var{corr_u1u2} into corr_u1u2;
    %end;
   close &init_parms;

   stddev_u = exp(logsd_u)`;
   dim_u    = nrow(stddev_u);

   corr_u = i(dim_u);
   %if (&modeltype ^= ONEPART) %then %do;
    corr_u[1,2] = corr_u1u2;
    if (corr_u[1,2] < -0.8) then corr_u[1,2] = -0.8;
    if (corr_u[1,2] >  0.8) then corr_u[1,2] =  0.8;
    corr_u[2,1] = corr_u[1,2];
    %end;

   cov_u  = corr_u # (stddev_u * stddev_u`);

/* parameterize cov(u) as cov(u) = SDS', where S is a lower triangular matrix with 1's  */
/* on the diagonal, and D is a diagonal matrix with positive values on the diagonal.    */

   root_u = root(cov_u)`;

   sqrt_d  = vecdiag(root_u);
   log_d   = log(sqrt_d##2);
   %if (&dim_u > 1) %then %do;
    s = root_u / repeat(sqrt_d`,dim_u,1);
    s_lower = symsqr(s[2:dim_u,1:(dim_u-1)]);
    %end;

 initcov = initcov || log_d`
         %if (&dim_u > 1) %then || s_lower`; %str(;)

 names   = names ||
         {%do i = 1 %to &dim_u;
            "LOG_D&i"
```

```
            %end;
          %do i = 2 %to &dim_u;
            %do j = 1 %to %eval(&i-1);
              "S&i&j"
              %end;
            %end; };

  %end;  /* %if (&cov_u_sds = Y & &dim_u > 0) %then %do */

  create _initcov from initcov [colname = names];
  append from initcov;
  close _initcov;
  quit;

 /* combine initial parameter estimates. */

  data _init_parms;
    merge &init_parms _initcov;

 /* drop uneeded parameters. */

    %if (&cov_u_sds = Y & &dim_u > 0) %then %do;
      drop LogSDu2 LogSDu3
          %if (&modeltype ^= ONEPART) %then LogSDu1 corr_u1u2;
          ;
      %end;
    run;

  %end;   /* %if (&init_cov_u = Y | &init_cov_e = Y) %then %do */

%else %do;
  data _init_parms;
    set &init_parms;
    run;
  %end;

%if (&print = V) %then %do;
  proc print data=_init_parms noobs;
    title%eval(&ntitle+1) "Initial Parameter Estimates for Bivariate Foods/Nutrients Model";
    run;
  %end;

 /*** fit model using proc NLMixed. ***/

     %if (&link = NONE)  %then title%eval(&ntitle+1) "Bivariate Foods/Nutrients
Model"%str(;);
%else %if (&link = LOGIT) %then title%eval(&ntitle+1) "Bivariate Foods/Nutrients Model
(Link = Logit)"%str(;);
%else                     title%eval(&ntitle+1) "Bivariate Foods/Nutrients Model (Link =
Probit)"%str(;);
```

```
data parms_b;
  run;

data pred_x_b;
  run;

data conv_b;
  status = 1;
  run;

%if (&print = N) %then ods exclude all %str(;);

proc nlmixed  data=&data &nloptions;
  parms / data=_init_parms;

  %if (&replicate_var ^= %str()) %then replicate &replicate_var%str(;);    /* weight variable */

  %if (&lambda1 ^= %str()) %then A1_Lambda = &lambda1%str(;);
  %if (&lambda2 ^= %str()) %then A2_Lambda = &lambda2%str(;);

  %if (&var_u1 = 0) %then u1 = 0 %str(;);
  %if (&var_u2 = 0) %then u2 = 0 %str(;);
  %if (&var_u3 = 0) %then u3 = 0 %str(;);

  %if (&var_u1 ^= %str()) %then var_u1 = &var_u1 %str(;);
  %if (&var_u2 ^= %str()) %then var_u2 = &var_u2 %str(;);
  %if (&var_u3 ^= %str()) %then var_u3 = &var_u3 %str(;);

  %if (%quote(&corr_u1u2) ^= %str()) %then corr_u1u2 = &corr_u1u2 %str(;);
  %if (%quote(&corr_u1u3) ^= %str()) %then corr_u1u3 = &corr_u1u3 %str(;);
  %if (%quote(&corr_u2u3) ^= %str()) %then corr_u2u3 = &corr_u2u3 %str(;);

 /* reparameterize the variance/covariance of e. */

  VAR_E2 = exp(2*A1_LogSDe);
  VAR_E3 = exp(2*A2_LogSDe);

  %if (&modeltype = ONEPART | &threshold ^= Y) %then %do;
    CORR_E1E3 = 0;
    CORR_E2E3 = (exp(2*Z_E2E3) - 1) / (exp(2*Z_E2E3) + 1);
    %end;
  %else %do;
    GAMMA = (exp(2*Z_GAMMA) - 1) / (exp(2*Z_GAMMA) + 1);
    THETA = (exp(2*Z_THETA) - 1) / (exp(2*Z_THETA) + 1);
    CORR_E1E3 = GAMMA * THETA;
    CORR_E2E3 = GAMMA * sqrt(1 - THETA**2);
    %end;

  COV_E1E3  = CORR_E1E3 * sqrt(VAR_E3);
```

```
 COV_E2E3  = CORR_E2E3 * sqrt(VAR_E2 * VAR_E3);

/* reparameterize the variance/covariance of u. */

 %if (&cov_u_sds = Y & &dim_u > 0) %then %do;
  %do i = 1 %to &dim_u;
   %let ii = %scan(&index_u,&i,%str( ));
   %if (&&var_u&ii ^= %str()) %then D&i = &&var_u&ii%str(;);
   %else D&i = exp(LOG_D&i)%str(;);
   %if (&&var_u&ii ^= %str()) %then %do j = 1 %to &dim_u;
    %if (&j < &i) %then S&i&j = 0%str(;);
    %if (&j > &i) %then S&j&i = 0%str(;);
    %end;
   %end;

  %do i = 1 %to &dim_u;
   %let ii = %scan(&index_u,&i,%str( ));
   VAR_U&ii = D&i
       %do k = 1 %to %eval(&i-1);
         + D&k * S&i&k**2
         %end;
       ;
     %do j = 1 %to %eval(&i-1);
      %let jj = %scan(&index_u,&j,%str( ));
      COV_U&jj.U&ii = D&j * S&i&j
          %do k = 1 %to %eval(&j-1);
            + D&k * S&i&k * S&j&k
            %end;
          ;
      CORR_U&jj.U&ii = COV_U&jj.U&ii / sqrt(VAR_U&jj * VAR_U&ii);
      %end;
   %end;
  %end;  /* %if (&cov_u_sds = Y & &dim_u > 0) %then %do */

 %else %do;
  %do i = 1 %to &dim_u;
   %let ii = %scan(&index_u,&i,%str( ));
   %if (&&var_u&ii ^= %str()) %then VAR_U&ii = &&var_u&ii %str(;);
   %do j = 1 %to %eval(&i-1);
    %let jj = %scan(&index_u,&j,%str( ));
    %if (%quote(&&corr_u&jj.u&ii) ^= %str()) %then %do;
     CORR_U&jj.U&ii = %quote(&&corr_u&jj.u&ii);
        COV_U&jj.U&ii = sqrt(var_u&jj * var_u&ii) * corr_u&jj.u&ii;
     %end;
    %else %do;
     CORR_U&jj.U&ii = COV_U&jj.U&ii / sqrt(VAR_U&jj * VAR_U&ii);
     %end;
    %end;
   %end;
  %end;  /* %else %do */
```

```
/* get response data. */

 %if (&modeltype = ONEPART) %then %do;
  if (&response1 > 0) then do;
    if (A1_Lambda = 0) then _boxcoxy1 = log(&response1);
    else _boxcoxy1 = (&response1**A1_Lambda - 1) / A1_Lambda;
    end;
  else do;
    _boxcoxy1 = .;
    end;
  %end;

 %else %do;
  if (&response1 > 0) then do;
    _y1 = 1;
    if (A1_Lambda = 0) then _boxcoxy1 = log(&response1);
    else _boxcoxy1 = (&response1**A1_Lambda - 1) / A1_Lambda;
    end;
  else if (&response1 = 0) then do;
    _y1 = 0;
    _boxcoxy1 = .;
    end;
  else do;
    _y1 = .;
    _boxcoxy1 = .;
    end;
  %end;

 if (&response2 > 0) then do;
  if (A2_Lambda = 0) then _boxcoxy2 = log(&response2);
  else _boxcoxy2 = (&response2**A2_Lambda - 1) / A2_Lambda;
  end;
 else do;
  _boxcoxy2 = .;
  end;

/* calculate linear predictor for probability of consumption. */

 %if (&modeltype ^= ONEPART) %then %do;
  x_p1 = p1_intercept;
  %do i = 1 %to &nx_p1;
    %let xi = %scan(&covars_prob1,&i,%str( ));
    x_p1 = x_p1 + p1_&xi * &xi;
    %end;
  xu_p1 = x_p1 + u1;

/* logit or probit link. */
```

```
   %if (&link = LOGIT) %then %do;
     _p1 = 1 /(1 + exp(-xu_p1));
     _c1 = probit(1 - _p1);
     %end;
   %else %do;
     _p1 = probnorm(xu_p1);
     _c1 = -xu_p1;
     %end;
   %end;

/* calculate linear predictors for amount consumed. */

 x_a1 = a1_intercept;
 %do i = 1 %to &nx_a1;
   %let xi = %scan(&covars_amt1,&i,%str( ));
   x_a1 = x_a1 + a1_&xi * &xi;
   %end;
 xu_a1 = x_a1 + u2;

 x_a2 = a2_intercept;
 %do i = 1 %to &nx_a2;
   %let xi = %scan(&covars_amt2,&i,%str( ));
   x_a2 = x_a2 + a2_&xi * &xi;
   %end;
 xu_a2 = x_a2 + u3;

 if (_boxcoxy1 ^= .) then _z1 = (_boxcoxy1 - xu_a1) / sqrt(var_e2);
 else _z1 = .;

 if (_boxcoxy2 ^= .) then _z2 = (_boxcoxy2 - xu_a2) / sqrt(var_e3);
 else _z2 = .;

/* calculate likelihood for probability of consumption. */

 %if (&modeltype = ONEPART) %then %do;
   l_b = 1;
   %end;

 %else %do;
   %if (&threshold ^= Y) %then %do;
     if (_y1 = 0) then l_b = 1 - _p1;
     else if (_y1 = 1) then l_b = _p1;
     else if (_boxcoxy2 ^= .) then l_b = 1;
     else l_b = .;
     %end;

/* latent variable threshold model. */

   %else %do;
     if (_y1 = 0 & _boxcoxy2 ^= .) then do;
```

```
      _m = corr_e1e3 * _z2;
      _s = sqrt(1 - corr_e1e3**2);
      _w = (_c1 - _m) / _s;
      l_b = probnorm(_w);
      end;

    else if (_y1 = 1 & _boxcoxy2 ^= .) then do;
      _m = corr_e1e3 / (1 - corr_e2e3**2) * (_z2 - corr_e2e3 * _z1);
      _s = sqrt((1 - corr_e1e3**2 - corr_e2e3**2) / (1 - corr_e2e3**2));
      _w = (_c1 - _m) / _s;
      l_b = 1 - probnorm(_w);
      end;

    else if (_y1 = 0) then l_b = 1 - _p1;
    else if (_y1 = 1) then l_b = _p1;
    else if (_boxcoxy2 ^= .) then l_b = 1;
    else l_b = .;
    %end;
  %end;

 if (l_b ^= .) then ll_b = log(l_b);
 else ll_b = .;

/* calculate likelihood for amount consumed. */

 pi = arcos(-1);

 if (_z1 ^= . & _z2 ^= .) then
   ll_n = -log(2 * pi * sqrt(VAR_E2 * VAR_E3 * (1-CORR_E2E3**2))) -
          (_z1**2 - 2 * CORR_E2E3 * _z1 * _z2 +_z2**2) / (2 * (1-CORR_E2E3**2)) +
           (A1_Lambda - 1)*log(&response1) + (A2_Lambda - 1)*log(&response2);

 else if (_z1 ^= .) then
   ll_n = -log(sqrt(2 * pi * VAR_E2)) - _z1**2 / 2 +
          (A1_Lambda - 1)*log(&response1);

 else if (_z2 ^= .) then
   ll_n = -log(sqrt(2 * pi * VAR_E3)) - _z2**2 / 2 +
          (A2_Lambda - 1)*log(&response2);

 %if (&modeltype ^= ONEPART) %then %do;
   else if (_y1 = 0) then ll_n = 0;
   %end;

 else ll_n = .;

 ll = ll_b + ll_n;

 if (&response1 > .Z) then _response = &response1;
 else _response = &response2;
```

```
 model _response ~ general(ll);

 /* specify random effects. */

%if (&dim_u > 0) %then %do;
 random %do i = 1 %to &dim_u;
        %let ii = %scan(&index_u,&i,%str( ));
        u&ii
        %end;
       ~ normal([%do i = 1 %to &dim_u;
                  0
                  %if (&i < &dim_u) %then ,;
                  %end;
                ],
                [%do i = 1 %to &dim_u;
                  %let ii = %scan(&index_u,&i,%str( ));
                  %do j = 1 %to %eval(&i-1);
                    %let jj = %scan(&index_u,&j,%str( ));
                     cov_u&jj.u&ii,
                     %end;
                  var_u&ii
                  %if (&i < &dim_u) %then ,;
                  %end;
                ])
      subject=&subject;
 %end;

 /* estimate additional parameters. */

 %do i = 1 %to &dim_u;
  %let ii = %scan(&index_u,&i,%str( ));
   estimate "VAR_U&ii" VAR_U&ii;
   %end;

 %do i = 1 %to &dim_u;
  %let ii = %scan(&index_u,&i,%str( ));
  %do j = %eval(&i+1) %to &dim_u;
    %let jj = %scan(&index_u,&j,%str( ));
     estimate "COV_U&ii.U&jj" COV_U&ii.U&jj;
     %end;
   %end;

 %do i = 1 %to &dim_u;
  %let ii = %scan(&index_u,&i,%str( ));
  %do j = %eval(&i+1) %to &dim_u;
    %let jj = %scan(&index_u,&j,%str( ));
     estimate "CORR_U&ii.U&jj" CORR_U&ii.U&jj;
     %end;
   %end;
```

```
  estimate "VAR_E2" VAR_E2;
  estimate "VAR_E3" VAR_E3;

  *%if (&modeltype ^= ONEPART & &threshold = Y) %then estimate "GAMMA" GAMMA
%str(;);
  *%if (&modeltype ^= ONEPART & &threshold = Y) %then estimate "THETA" THETA
%str(;);

  %if (&modeltype ^= ONEPART & &threshold = Y) %then estimate "COV_E1E3" COV_E1E3
%str(;);
  estimate "COV_E2E3" COV_E2E3;

  %if (&modeltype ^= ONEPART & &threshold = Y) %then estimate "CORR_E1E3"
CORR_E1E3 %str(;);
  estimate "CORR_E2E3" CORR_E2E3;

  %if (&modeltype ^= ONEPART) %then predict x_p1 out=pred_x_p1%str(;);
  predict x_a1 out=pred_x_a1;
  predict x_a2 out=pred_x_a2;

  ods output ParameterEstimates=_parms_b;
  ods output AdditionalEstimates=_adparms_b;
  ods output FitStatistics=fit_b;
  ods output ConvergenceStatus=conv_b;
  run;

title%eval(&ntitle+1);

%if (&print = N) %then ods exclude none %str(;);

data _null_;
  set conv_b;
  call symput("status",trim(left(put(status, 6.))));
  run;

%if (&status ^= 0) %then %goto exit;

 /*** save parameter estimates and predicted values. ***/

proc transpose data=_parms_b out=_parms_b(drop=_name_);
  id parameter;
  var estimate;
  run;

proc transpose data=_adparms_b out=_adparms_b(drop=_name_);
   id label;
   var estimate;
   run;
```

```
data parms_b;
  merge _parms_b _adparms_b;

 /* add fixed parameters. */

  %if (&lambda1 ^= %str()) %then A1_Lambda = &lambda1%str(;);
  %if (&lambda2 ^= %str()) %then A2_Lambda = &lambda2%str(;);

  %if (&modeltype ^= ONEPART & &var_u1 = 0) %then VAR_U1 = 0%str(;);
  %if (&var_u2 = 0) %then VAR_U2 = 0%str(;);
  %if (&var_u3 = 0) %then VAR_U3 = 0%str(;);

  %if (&modeltype ^= ONEPART) %then %do;
    %if (&var_u1 = 0 | &var_u2 = 0) %then COV_U1U2 = 0%str(;);
    %if (&var_u1 = 0 | &var_u3 = 0) %then COV_U1U3 = 0%str(;);
    %end;
  %if (&var_u2 = 0 | &var_u3 = 0) %then COV_U2U3 = 0%str(;);

  %if (&modeltype ^= ONEPART) %then %do;
    %if (&var_u1 = 0 | &var_u2 = 0) %then CORR_U1U2 = 0%str(;);
    %if (&var_u1 = 0 | &var_u3 = 0) %then CORR_U1U3 = 0%str(;);
    %end;
  %if (&var_u2 = 0 | &var_u3 = 0) %then CORR_U2U3 = 0%str(;);
  run;

data pred_x_b;
  merge &data
       %if (&modeltype ^= ONEPART) %then pred_x_p1(keep=&subject &repeat pred
rename=(pred=pred_x_p1));
                                    pred_x_a1(keep=&subject &repeat pred rename=(pred=pred_x_a1))
                                    pred_x_a2(keep=&subject &repeat pred rename=(pred=pred_x_a2));
    by &subject &repeat;

  label %if (&modeltype ^= ONEPART) %then pred_x_p1 = " ";
       pred_x_a1 = " " pred_x_a2 = " ";
  run;

 /*** delete unneeded data sets. */

proc datasets lib=work nolist;
  delete _initcov _init_parms _parms_b _adparms_b
       %if (&modeltype ^= ONEPART) %then pred_x_p1;
       pred_x_a1 pred_x_a2;
  run;
  quit;

%exit:

title%eval(&ntitle+1);
```

```
%if (&replicate_var ^= %str()) %then footnote %str(;);

%mend Food_Bivariate;

 /*** end of macro Food_Bivariate ***************************************/



/*****************************************************************************
*
 * Macro Numint_Density calculates predicted values for the bivariate         *
 * measurement error model.                                                   *

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

%macro NumInt_Density (data       = ,
                       subject    = ,
                       param      = ,
                       modeltype  = TWOPART,
                       link       = PROBIT,
                       covars     = ,
                       constant   = 1000,
                       lambda_t1  = 1,
                       lambda_t2  = 1,
                       lambda_td  = 1,
                       shift1     = 0,
                       shiftd     = 0,
                       epi        = N,
                       print      = N,
                       ntitle     = 1);

%let modeltype = %upcase(&modeltype);
%let link      = %upcase(&link);
%let print     = %upcase(%substr(&print,1,1));
%let epi       = %upcase(%substr(&epi,1,1));

%if (&link      ^= LOGIT)  %then %let link = PROBIT;
%if (&modeltype ^= ONEPART) %then %let modeltype = TWOPART;

%let p1_names = ;
%let a1_names = ;
%let a2_names = ;
%let ncov     = 0;

%do %while (%scan(&covars,%eval(&ncov+1),%str( )) ^= %str());
  %let ncov = %eval(&ncov+1);
  %let xi = %scan(&covars,&ncov,%str( ));
  %let p1_names = &p1_names p1_&xi;
  %let a1_names = &a1_names a1_&xi;
```

```
  %let a2_names = &a2_names a2_&xi;
  %end;

%let nt1 = 0;
%do %while (%quote(%scan(&lambda_t1,%eval(&nt1+1),%str( ))) ^= %str());
  %let nt1 = %eval(&nt1+1);
  %end;

%let nt2 = 0;
%do %while (%quote(%scan(&lambda_t2,%eval(&nt2+1),%str( ))) ^= %str());
  %let nt2 = %eval(&nt2+1);
  %end;

%let ntd = 0;
%do %while (%quote(%scan(&lambda_td,%eval(&ntd+1),%str( ))) ^= %str());
  %let ntd = %eval(&ntd+1);
  %end;

proc sort data=&data;  by &subject;  run;

 /* calculate predicted values. */

data _predicted (keep=&subject pred_x_a1 pred_x_a2 %if (&modeltype = TWOPART) %then
pred_x_p1;);
  set &data (keep=&subject &covars);
  if (_n_ = 1) then set &param (keep=a1_intercept a2_intercept &a1_names &a2_names
                          %if (&modeltype = TWOPART) %then p1_intercept &p1_names;);

  if (nmiss(of &covars) > 0) then delete;

  array x  (*) &covars;
  array a1 (*) &a1_names;
  array a2 (*) &a2_names;

  pred_x_a1 = a1_intercept;
  pred_x_a2 = a2_intercept;
  do i = 1 to dim(x);
    pred_x_a1 = pred_x_a1 + a1(i) * x(i);
    pred_x_a2 = pred_x_a2 + a2(i) * x(i);
    end;

  %if (&modeltype = TWOPART) %then %do;
    array p1 (*) &p1_names;

    pred_x_p1 = p1_intercept;
    do i = 1 to dim(x);
      pred_x_p1 = pred_x_p1 + p1(i) * x(i);
      end;
    %end;
  run;
```

```
 /* determine number and order of random effects u. */

%let dim_u   = 0;
%let index_u = ;
%let name_u  = ;

%if (&modeltype ^= ONEPART) %then %do;
  %let dim_u   = %eval(&dim_u + 1);
  %let index_u = &index_u 1;
  %let name_u  = &name_u u1;
  %end;

%let dim_u   = %eval(&dim_u + 1);
%let index_u = &index_u 2;
%let name_u  = &name_u u2;

%let dim_u   = %eval(&dim_u + 1);
%let index_u = &index_u 3;
%let name_u  = &name_u u3;

 /* calculate the cholesky decomposition of the covariance matrix of u. */

proc iml;

  %if (&dim_u > 0) %then %do;
    use &param;
      read all var{%do i = 1 %to &dim_u;
                     %let ii = %scan(&index_u,&i,%str( ));
                     %do j = 1 %to %eval(&i-1);
                       %let jj = %scan(&index_u,&j,%str( ));
                       COV_U&jj.U&ii
                       %end;
                     VAR_U&ii
                     %end;
                   } into var_u;
      close &param;

    var_u     = sqrsym(var_u);
    u_zero    = (vecdiag(var_u) = 0);
    u_nonzero = (vecdiag(var_u) > 0);
    var_u     = var_u # (u_nonzero * u_nonzero`);
    var_u     = var_u + diag(u_zero);

    root_u = root(var_u)`;
    root_u = root_u # (u_nonzero * u_nonzero`);

  %if (&modeltype = ONEPART) %then root_u = block({0}, root_u) %str(;);
    %end;  /* %if (&dim_u > 0) %then %do */
```

```
 %else %do;
  root_u = repeat(0,3,3);
  %end;

 root_u = symsqr(root_u)`;

/* create sas data set. */

 names = {"b11" "b21" "b22" "b31" "b32" "b33"};
 create _rootu from root_u [colname=names];
 append from root_u;
 close _rootu;
 quit;

/* numerically integrate predicted intake. */

data pred_t (keep=&subject
              %do i = 1 %to &nt1; pred_t1_&i %end;
              %do i = 1 %to &nt2; pred_t2_&i %end;
              %do i = 1 %to &ntd; pred_td_&i %end;);
 if (_n_ = 1) then set &param(keep=a1_lambda a2_lambda var_e2 var_e3 min_a1 min_a2);
 if (_n_ = 1) then set _rootu;
 set _predicted;

 stddev_e2 = sqrt(var_e2);
 stddev_e3 = sqrt(var_e3);

 if (a1_lambda ^= 0) then a1_lambda_inv = 1 / a1_lambda;
 if (a2_lambda ^= 0) then a2_lambda_inv = 1 / a2_lambda;

/* quadrature points and weights. */

 c1 = -2.1;  w1 = 0.063345;
 c2 = -1.3;  w2 = 0.080255;
 c3 = -0.8;  w3 = 0.070458;
 c4 = -0.5;  w4 = 0.159698;
 c5 =  0.0;  w5 = 0.252489;
 c6 =  0.5;  w6 = 0.159698;
 c7 =  0.8;  w7 = 0.070458;
 c8 =  1.3;  w8 = 0.080255;
 c9 =  2.1;  w9 = 0.063345;

 array c (*) c1-c9;
 array w (*) w1-w9;

 sumw = 0;
 do i = 1 to dim(c);  sumw = sumw + w(i);  end;
 do i = 1 to dim(c);  w(i) = w(i) / sumw;  end;
```

```
  if (nmiss(of pred_x_a1 pred_x_a2 %if (&modeltype ^= ONEPART) %then pred_x_p1;) = 0)
then do;

  %do i = 1 %to &nt1;
    pred_t1_&i = 0;
    %end;

  %do i = 1 %to &nt2;
    pred_t2_&i = 0;
    %end;

  %do i = 1 %to &ntd;
    pred_td_&i = 0;
    %end;

 /* calculate p1 = probability to consume (probit or logit link). */

  %if (&modeltype = ONEPART) %then %do;
    p1 = 1;
    z1 = 0;
    weight1 = 1;
    %end;

  %else %do;
    do i1 = 1 to dim(c);
      z1 = c(i1);
      u1 = b11 * z1;
      weight1 = w(i1);

      pred_p1 = pred_x_p1 + u1;
      %if (&link = PROBIT) %then p1 = probnorm(pred_p1) %str(;);
      %else p1 = 1 / (1 + exp(-pred_p1)) %str(;);
    %end;

 /* calculate a1 = amount on consumption day. */

    do i2 = 1 to dim(c);
      z2 = c(i2);
      u2 = b21 * z1 + b22 * z2;
      weight12 = weight1 * w(i2);

      bc_a1 = pred_x_a1 + u2;

 /* back-transform amount to original scale. */

 /* surveillance assumption: 9-point approximation of integral (exact for log transformation). */

      %if (&epi ^= Y) %then %do;

        a1 = 0;
```

```
      if (a1_lambda = 0) then a1 = exp(bc_a1 + var_e2 / 2);
      else do i = 1 to dim(c);
        bc_ai = bc_a1 + stddev_e2 * c(i);
        temp = max(0, a1_lambda * bc_ai + 1);
        ai = max(min_a1, temp**a1_lambda_inv);
        a1 = a1 + w(i) * ai;
        end;

      %end;   /* %if (&epi ^= Y) %then %do */

/* epi assumption. */

     %if (&epi = Y) %then %do;

      if (a1_lambda = 0) then a1 = exp(bc_a1);
      else do;
        temp = max(0, a1_lambda * bc_a1 + 1);
        a1 = max(min_a1, temp**a1_lambda_inv);
        end;

      %end;   /* %if (&epi = Y) %then %do */

/* calculate t1 = p1 * a1. */

     t1 = p1 * a1 + &shift1;

/* transform t1. */

     %do i = 1 %to &nt1;
       %let lambdai = %scan(&lambda_t1,&i,%str( ));
       %if (%quote(&lambdai) = 0) %then boxcox_t1 = log(t1) %str(;);
       %else boxcox_t1 = t1**&lambdai %str(;);
       pred_t1_&i = pred_t1_&i + weight12 * boxcox_t1;
       %end;

/* calculate a2 = energy intake. */

     do i3 = 1 to dim(c);
       z3 = c(i3);
       u3 = b31 * z1 + b32 * z2 + b33 * z3;
       weight123 = weight12 * w(i3);

       bc_a2 = pred_x_a2 + u3;

/* back-transform energy intake to original scale. */

/* surveillance assumption: 9-point approximation of integral (exact for log transformation). */

       %if (&epi ^= Y) %then %do;
```

```
        a2 = 0;
        if (a2_lambda = 0) then a2 = exp(bc_a2 + var_e3 / 2);
        else do i = 1 to dim(c);
          bc_ai = bc_a2 + stddev_e3 * c(i);
          temp = max(0, a2_lambda * bc_ai + 1);
          ai = max(min_a2, temp**a2_lambda_inv);
          a2 = a2 + w(i) * ai;
          end;

        %end;   /* %if (&epi ^= Y) %then %do */

/* epi assumption. */

       %if (&epi = Y) %then %do;

        if (a2_lambda = 0) then a2 = exp(bc_a2);
        else do;
          temp = max(0, a2_lambda * bc_a2 + 1);
          a2 = max(min_a2, temp**a2_lambda_inv);
          end;

        %end;   /* %if (&epi = Y) %then %do */

/* calculate density. */

       t2 = a2;
       td = &constant * (t1 / t2) + &shiftd;

/* transform t2 and td. */

     %do i = 1 %to &nt2;
       %let lambdai = %scan(&lambda_t2,&i,%str( ));
       %if (%quote(&lambdai) = 0) %then boxcox_t2 = log(t2) %str(;);
       %else boxcox_t2 = t2**&lambdai %str(;);
       pred_t2_&i = pred_t2_&i + weight123 * boxcox_t2;
       %end;

     %do i = 1 %to &ntd;
       %let lambdai = %scan(&lambda_td,&i,%str( ));
       %if (%quote(&lambdai) = 0) %then boxcox_td = log(td) %str(;);
       %else boxcox_td = td**&lambdai %str(;);
       pred_td_&i = pred_td_&i + weight123 * boxcox_td;
       %end;

       end;   /* do i3 = 1 to dim(c) */
     end;   /* do i2 = 1 to dim(c) */

  %if (&modeltype ^= ONEPART) %then %do;
    end;   /* do i1 = 1 to dim(c) */
    %end;
```

```
  %do i = 1 %to &nt1;
    %let lambdai = %scan(&lambda_t1,&i,%str( ));
    %if (%quote(&lambdai) ^= 0) %then pred_t1_&i = (pred_t1_&i - 1) / &lambdai %str(;);
    %end;

  %do i = 1 %to &nt2;
    %let lambdai = %scan(&lambda_t2,&i,%str( ));
    %if (%quote(&lambdai) ^= 0) %then pred_t2_&i = (pred_t2_&i - 1) / &lambdai %str(;);
    %end;

  %do i = 1 %to &ntd;
    %let lambdai = %scan(&lambda_td,&i,%str( ));
    %if (%quote(&lambdai) ^= 0) %then pred_td_&i = (pred_td_&i - 1) / &lambdai %str(;);
    %end;

  end;   /* if (nmiss(of pred_x_p1 pred_x_a1 pred_x_a2) = 0) then do */

  run;

%if (&print = Y) %then %do;
  title%eval(&ntitle+1) "Distribution of Predicted Intake";

  proc means data=pred_t(drop=&subject) n mean var std stderr min max;
    run;

  title%eval(&ntitle+1);
  %end;

%mend NumInt_Density;


/*************************************************************************************
*
 * End of macro Numint_Density                                    *

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