Supplemental Materials

1. Implementation of Data Generating Process in Stan:

To facilitate the computation, it is convenient to represent the random effects in standardized form as multivariate standard normals. To standardize the random wave effects $\nu_{0,ij}$ and $\tau_{0,ij}$, we need to find $C$, a lower triangular matrix, and $\delta_{1,ij}$ and $\delta_{2,ij}$, independent bivariate standard normals, such that

$$
\begin{bmatrix}
\nu_{0,ij} \\
\tau_{0,ij}
\end{bmatrix} = C \begin{bmatrix}
\delta_{1,ij} \\
\delta_{2,ij}
\end{bmatrix} = \begin{bmatrix}
c_{11} & 0 \\
c_{21} & c_{22}
\end{bmatrix} \begin{bmatrix}
\delta_{1,ij} \\
\delta_{2,ij}
\end{bmatrix}
$$

(1)

It can easily be shown that $C$ is the Cholesky decomposition of the covariance matrix $\Sigma_{\nu_0,\tau_0}$.

$$
\text{COV} \begin{bmatrix}
\nu_{0,ij} \\
\tau_{0,ij}
\end{bmatrix} = \text{COV} \begin{bmatrix}
\delta_{1,ij} \\
\delta_{2,ij}
\end{bmatrix} = \text{CCOV} \begin{bmatrix}
\delta_{1,ij} \\
\delta_{2,ij}
\end{bmatrix} C^T = CC^T
$$

(2)

$$
\implies \Sigma_{\nu_0,\tau_0} = CC^T
$$

(3)

Similarly, we can standardize the random subject effects $\gamma_{0,i}, \gamma_{1,i}, \lambda_{0,i}, \lambda_{1,i}$

$$
\begin{bmatrix}
\gamma_{0,i} \\
\gamma_{1,i} \\
\lambda_{0,i} \\
\lambda_{1,i}
\end{bmatrix} = D \begin{bmatrix}
\theta_{1,i} \\
\theta_{2,i} \\
\theta_{3,i} \\
\theta_{4,i}
\end{bmatrix} = \begin{bmatrix}
d_{11} & 0 & 0 & 0 \\
d_{21} & d_{22} & 0 & 0 \\
d_{31} & d_{32} & d_{33} & 0 \\
d_{41} & d_{42} & d_{43} & d_{44}
\end{bmatrix} \begin{bmatrix}
\theta_{1,i} \\
\theta_{2,i} \\
\theta_{3,i} \\
\theta_{4,i}
\end{bmatrix}
$$

(4)

where $\begin{bmatrix}
\theta_{1,i} \\
\theta_{2,i} \\
\theta_{3,i} \\
\theta_{4,i}
\end{bmatrix} \sim \mathcal{N}_4(\mathbf{0}, \mathbb{I}_4)$, and $DD^T = \Sigma_{\gamma_{0,1},\gamma_{1,1},\lambda_{0,1},\lambda_{1,1}}$

Once the random effects are standardized, we can re-express our models in terms of the standard normals $\theta$ and $\delta$.

$$
Y_{ijk} = \beta_0 + \beta_1 \times \text{smk}_{ik} + \beta_2 \times \text{smk}_{ij} + \beta_3 \times \text{smk}_{ijk} + \beta_4 \times \text{wave}_{ij} +
$$

$$
\begin{array}{c}
d_{11} \times \theta_{1,i} + (d_{21} \times \theta_{1,i} + d_{22} \times \theta_{2,i}) \times \text{wave}_{ij} +

c_{11} \times \delta_{1,ij} +
\end{array}
$$

$$
\begin{array}{c}
\epsilon_{ijk}
\end{array}
$$

$$
\text{log}(\sigma_{\epsilon,ijk}^2) = \alpha_0 + \alpha_1 \times \text{smk}_{ik} + \alpha_2 \times \text{smk}_{ij} + \alpha_3 \times \text{smk}_{ijk} + \alpha_4 \times \text{wave}_{ij} +
$$

$$
\begin{array}{c}
(d_{31} \times \theta_{1,i} + d_{32} \times \theta_{2,i} + d_{33} \times \theta_{3,i}) +
(d_{41} \times \theta_{1,i} + d_{42} \times \theta_{2,i} + d_{43} \times \theta_{3,i} + d_{44} \times \theta_{4,i}) \times \text{wave}_{ij} +
\end{array}
$$

$$
\begin{array}{c}
(c_{21} \times \delta_{1,ij} + c_{22} \times \delta_{2,ij})
\end{array}
$$

Below is sample code in R to simulate a three level data structure to be used below since the adolescent smoking data cannot be shared publicly at this time.

```r
DataGen <- function(nsubj, nwave, nob, beta0, beta1, beta2, beta3, alpha0, alpha1, alpha2, 
                    sdsubj_loc, sdsubj_scale, 
sdwave_loc, sdwave_scale, 
```
cor.subj_loc = matrix(c(1,-0.2,-0.2,1), nrow = 2, byrow = TRUE),
cor.subj_scale = matrix(c(1,-0.2,-0.2,1), nrow = 2, byrow = TRUE))

## Generate subject wave and time indicators
N <- nsubj * nwave * nobs
subject <- gl(n = nsubj, k = nwave * nobs)
wave <- rep(gl(n = nwave,k = nobs), nsubj)
obs <- rep(seq_len(nobs), nsubj * nwave)

## Generate subject, wave and obs level covariates (observed).
x.subj <- rep(rbinom(nsubj, 1, 0.5), each = nwave * nobs)
x.wave <- wave
x.obs <- rbinom(nobs, 1, 0.5)

## Generate subject, wave level random effects
chol.subj_loc <- t(chol(cor.subj_loc))
sigma.sd.subj_loc <- diag(sd.subj_loc) %*% chol.subj_loc
subj_loc <- mvrnorm(nsubj, mu = c(0, 0),
                   Sigma = sigma.sd.subj_loc %*% t(sigma.sd.subj_loc))
chol.subj_scale <- t(chol(cor.subj_scale))
sigma.sd.subj_scale <- diag(sd.subj_scale) %*% chol.subj_scale
subj_scale <- mvrnorm(nsubj, mu = c(0, 0),
                      Sigma = sigma.sd.subj_scale %*% t(sigma.sd.subj_scale))

## Generate the mean
y.mean <- beta0 + beta1 * x1 + beta2 * x2 + beta3 * x3 +
         subj1.loc + subj2.loc * x2 + wave.loc

## Generate the sd
y.sd <- sqrt(exp(alpha0 + alpha1 * x1 + alpha2 * x2 +
                 subj1.scale + subj2.scale * x2 + wave.scale))

## Generate response
y <- rnorm(N, mean = y.mean, sd = y.sd)

## Return a data frame
df.LSME <- data.frame(subject = subject, wave = wave, time = time,
                       gender = x.subj, wave = x.wave, smk = x.obs, y = y)
return(df.LSME)

Below is the code in R to prepare data to be used in Stan.

# Simulate data by calling DataGen function
df.LSME <- DataGen(nsubj = 100, nwave = 10, ntime = 10,
beta0 = 1, beta1 = 1, beta2 = 1, beta3 = 1,
alpha0 = 0.3, alpha1 = 0.2, alpha2 = 0.1,
sd.subj_loc = c(1, 0.5),
sd.subj_scale = c(0.5, 0.25),
sd.wave_loc = 1,
sd.wave_scale = 0.5,
cor.subj_loc = matrix(c(1, -0.2,
-0.2, 1),
nrow = 2,
byrow = TRUE),
cor.subj_scale = matrix(c(1, -0.2,
-0.2, 1),
nrow = 2,
byrow = TRUE))

# Prepare data for stan
nsubj <- 100
nwave <- 10
N <- nrow(df.LSME)
ni <- 10
subject <- df.LSME$subject
wave <- df.LSME$wave
# Decompose x3 into subject, wave and observation level
x3 <- df.LSME$x3
x3.i <- as.vector(tapply(df.LSME$x3, df.LSME$subject, mean))
x3.subj <- rep(x3.i, each = 100)
x3.ij <- t(tapply(df.LSME$x3, list(df.LSME$subject, df.LSME$wave), mean))
x3.wave <- rep(x3.ij, each = 10) - x3.subj
x3.obs <- x3 - x3.wave
# Other variables
x1 <- df.LSME$x1
x2 <- df.LSME$x2
wave <- df.LSME$wave
X_mean <- unname(model.matrix(˜ 1 + x1 + x2 + x3.subj + x3.wave + x3.obs))
X_var <- unname(model.matrix(˜ 1 + x1 + x2 + x3.subj + x3.wave))
Z_subj <- unname(model.matrix(˜ 1 + x2))
y <- df.LSME$y
# Incorporate into a list
StanDat <- list(
  N = N,
  nsubj = nsubj,
  nwave = nwave,
  subject = subject,
  wave = wave,
  X_mean = X_mean,
  X_var = X_var,
  Z_subj = Z_subj,
  y = y)
# Get initial values for stan
lme_fit <- lmer(y ˜ x1 + x2 + x3.subj + x3.wave + x3.obs +
                (1 + x2 | subject) +
                (1 | subject:wave))
sum <- summary(lme_fit)

MLS_init <- function(){
  list(
    beta = sum$coefficients[, "Estimate"],
    alpha = c(log(sum$sigma), 0, 0, 0, 0),
    C_subject_loc = diag(2),
    C_subject_scale = diag(2),
    sigma_subj_loc = sqrt(diag(VarCorr(lme_fit)$subject)),
    sigma_subj_scale = c(0.1, 0.1),
    sigma_wave_loc = sqrt(diag(VarCorr(lme_fit)$"subject:wave")),}
Below is sample code in Stan to estimate the model presented in this manuscript. Save this stan file in the same working directory as LSME_model.stan.

data {
  int<lower=1> N; //number of data points
  int<lower=1> nsubj; //number of subjects
  int<lower=1> nwave; //number of waves
  int<lower=1, upper=nsubj> subject[N]; //indicator for subjects
  int<lower=1, upper=nwave> wave[N]; //indicator for waves
  row_vector[6] X_mean[N]; //design matrix for fixed effect
  row_vector[5] X_var[N]; //design matrix for fixed effect
  row_vector[2] Z_subj[N]; //design matrix for subject level RE
  real y[N]; //outcome
}

parameters {
  vector[6] beta; //fixed effect for mean
  vector[5] alpha; //fixed effect for variance
  cholesky_factor_corr[2] C_subj_loc; //cholesky components for RE at subject level
  cholesky_factor_corr[2] C_subj_scale; //cholesky components for RE at subject level
  vector<lower=0>[2] sigma_subj_loc; //se for RE at subject level
  vector<lower=0>[2] sigma_subj_scale; //se for RE at subject level
  real<lower=0> sigma_wave_loc; //se for RE at wave level
  real<lower=0> sigma_wave_scale; //se for RE at wave level
  vector[2] z_subj_loc[nsubj]; //RE at subject level
  vector[2] z_subj_scale[nsubj]; //RE at subject level
  real z_wave_loc[nwave]; //RE at wave level
  real z_wave_scale[nwave]; //RE at wave level
}

transformed parameters {
  matrix[2,2] Sigma_subj_loc;
  matrix[2,2] Sigma_subj_scale;
  vector[2] var_subj_loc;
  vector[2] var_subj_scale;
  real var_wave_loc;
  real var_wave_scale;
  vector[2] nu_subj_loc[nsubj];
  vector[2] nu_subj_scale[nsubj];
  real nu_wave_loc[nwave];
  real nu_wave_scale[nwave];
  {
    Sigma_subj_loc = diag_pre_multiply(sigma_subj_loc, C_subj_loc);
    Sigma_subj_scale = diag_pre_multiply(sigma_subj_scale, C_subj_scale);
    var_subj_loc = sigmasubj_loc .* sigma_subj_loc;
    var_subj_scale = sigmasubj_scale .* sigma_subj_scale;
    var_wave_loc = sigma_wave_loc * sigma_wave_loc;
    var_wave_scale = sigma_wave_scale * sigma_wave_scale;
  }
}
for (j in 1 : nsubj) {
    nu_subj_loc[j] = Sigma_subj_loc * z_subj_loc[j];
    nu_subj_scale[j] = Sigma_subj_scale * z_subj_scale[j];
}
for (k in 1 : nwave) {
    nu_wave_loc[k] = sigma_wave_loc * z_wave_loc[k];
    nu_wave_scale[k] = sigma_wave_scale * z_wave_scale[k];
}

model {
    //priors
    C_subj_loc ~ lkj_corr_cholesky(1);
    C_subj_scale ~ lkj_corr_cholesky(1);
    //REs
    for (j in 1 : nsubj) {
        z_subj_loc[j] ~ normal(0, 1);
        z_subj_scale[j] ~ normal(0, 1);
    }
    for (k in 1 : nwave) {
        z_wave_loc[k] ~ normal(0, 1);
        z_wave_scale[k] ~ normal(0, 1);
    }
    //likelihood
    for (i in 1 : N) {
        y[i] ~ normal(X_mean[i] * beta +
                        Z_subj[i] * nu_subj_loc[subject[i]] +
                        nu_wave_loc[wave[i]],
                        sqrt(exp(X_var[i] * alpha +
                                Z_subj[i] * nu_subj_scale[subject[i]] +
                                nu_wave_scale[wave[i]])));
    }
}

Below is sample code in R to run the stan model and extract the MCMC samples.

# Run the stan model
MLS.fit <- stan(data = StanDat,
                file = "LSME_model.stan",
                init = MLS_init,
                iter = 1000,
                warmup = 500,
                chains = 4,
                cores = 10)
# Examine traceplot
traceplot(MLS.fit, pars = "beta")
stan_dens(MLS.fit, pars = "beta", inc_warmup = TRUE)
# Extract MCMC samples, get point estimate and CI
MCMC.beta <- extract(MLS.fit, pars = "beta")$beta
beta.est <- colMeans(MCMC.beta)
beta.sd <- apply(MCMC.beta, 2, sd)
beta.lower <- apply(MCMC.beta, 2, quantile, probs = 0.025)
beta.upper <- apply(MCMC.beta, 2, quantile, probs = 0.975)
2. Metropolis-Hastings-Gibbs Sampling Algorithm

For a full Bayesian approach, parameters (β, α) and random effects (γ0, γ1, λ0, λ1, ν0, τ0) are both regarded as random quantities while the data Y are regarded as fixed. To simplify the notation, denote θ = (β, α) as the model parameter vector, (γ1, λ1) = (γ0, i, γ1, i, λ0, i, λ1, i)_{i=1}^{n_i} as the random subject location/scale effects, (νi,j, τi,j) = (ν0, i, j, τ0, i, j)_{i,j=1}^{n_{i,j}} as the random wave location/scale effects, and Yi,j as the data vector.

Since θ, (γ, λ) and (ν, τ) are all random, they each follow some prior distribution before we get to observe the data Y, which we denote as π(θ), π(γ, λ), and π(ν, τ) respectively. Since individuals are assumed to be independent and correlations exist due to repeated measurements within the same individual, π(γ, λ) can be written as \( \prod_{i=1}^{n_i} \pi_i(γ_i, λ_i) \), the product of the prior for each individual (same applies to π(ν, τ)). A natural choice for π(γ, λ) is that it is difficult to sample from the conditional posteriors if they are not of recognized forms. One solution is to use the Gaussian priors that it is difficult to sample from the conditional posteriors if they are not of recognized forms. One solution is to use the Gaussian priors. However, in the absence of conjugacy, Metropolis-Hastings should be used to sample from each full conditional posterior with some probability (acceptance ratio)\(^2\). This is not of concern if we choose the priors as multivariate Gaussian, as normal likelihood will lead to conjugate posterior for multivariate Gaussian priors\(^7\). However, in the absence of conjugacy, Metropolis-Hastings should be used to sample from each full conditional.

The detailed Markov Chain Monte Carlo algorithm where (component wise) Metropolis-Hastings is nested in Gibbs sampling is listed below, with q being the corresponding (user defined) proposal distribution. After enough runs, the chains will ultimately converge to the joint posterior and one can summarize the posterior samples to get the parameter estimates as well as credible intervals.

1. Initialize at \((θ, γ, λ, ν, τ) = (θ^0, γ^0, λ^0, ν^0, τ^0)\)
2. Sample a single random value iteratively from each full conditional posterior by Metropolis-Hastings algorithm below, for \(t = 1, 2, \ldots\)
   a) Given the current value of \(θ^t\) = \(θ\), generate a proposed new value \(θ'\) according to \(q_θ(θ → θ')\), and accept \(θ'\) with probability \(A_θ = \min\left(1, \frac{P(θ' | γ^t, λ^t, ν^t, τ^t, Y)q_θ(θ → θ')}{P(θ | γ^t, λ^t, ν^t, τ^t, Y)q_θ(θ' → θ)}\right)\), and accept \(θ'\) with probability \(A_θ = \min\left(1, \frac{P(θ' | γ^t, λ^t, ν^t, τ^t, Y)q_θ(θ → θ')}{P(θ | γ^t, λ^t, ν^t, τ^t, Y)q_θ(θ' → θ)}\right)\).
   b) Given the current value of \((γ^t, λ^t) = (γ, λ)\), generate a proposed new pair \((γ', λ')\) according to \(q_{γ, λ}(γ, λ) → (γ', λ')\), and accept \((γ', λ')\) with probability \(A_{γ, λ} = \min\left(1, \frac{P(γ', λ' | θ^t, ν^t, τ^t, Y)q_{γ, λ}(γ, λ) → (γ', λ')} {P(γ, λ | θ^t, ν^t, τ^t, Y)q_{γ, λ}(γ', λ') → (γ, λ)}\right)\), and accept \((γ', λ')\) with probability \(A_{γ, λ} = \min\left(1, \frac{P(γ, λ | θ^t, ν^t, τ^t, Y)q_{γ, λ}(γ, λ) → (γ', λ')} {P(γ, λ | θ^t, ν^t, τ^t, Y)q_{γ, λ}(γ', λ') → (γ, λ)}\right)\).
   c) Given the current value of \((ν^t, τ^t) = (ν, τ)\), generate a proposed new value \((ν', τ')\) according to \(q_{ν, τ}(ν, τ) → (ν', τ')\), and accept \((ν', τ')\) with probability \(A_{ν, τ} = \min\left(1, \frac{P(ν', τ' | θ^t, γ^t, λ^t, Y)q_{ν, τ}(ν, τ) → (ν', τ')} {P(ν, τ | θ^t, γ^t, λ^t, Y)q_{ν, τ}(ν', τ') → (ν, τ)}\right)\), and accept \((ν', τ')\) with probability \(A_{ν, τ} = \min\left(1, \frac{P(ν, τ | θ^t, γ^t, λ^t, Y)q_{ν, τ}(ν, τ) → (ν', τ')} {P(ν, τ | θ^t, γ^t, λ^t, Y)q_{ν, τ}(ν', τ') → (ν, τ)}\right)\).
3. Repeat step a, b, c in (2) until convergence.