Supplementary Materials 1: comparing parameterisations

Contents

NB this supplementary data is from: Richard M.A. Parker, George Leckie, Harvey Goldstein, Laura D. Howe, Jon Heron, Alun D. Hughes, David M. Phillippo, Kate Tilling. "Joint modelling of individual trajectories, within-individual variability and a later outcome: systolic blood pressure through childhood and left ventricular mass in early adulthood"

This document fits joint models, with a within-individual outcome and an individual-level outcome, and with shared random effects, to simulated data. It demonstrates the equivalence of (a) allowing the residual error in the individual-level outcome to freely-covary with the individual-level random effects from the within-individual outcome in a covariance matrix to (b) fitting the random effects from the within-individual outcome as exposures in the model for the individual-level outcome. It uses WinBUGS (Lunn et al. 2000), called from R via `R2WinBUGS` (Sturtz, Ligges, and Gelman 2005).

Loading libraries; choosing options

```
library(R2WinBUGS) # call WinBUGS
library(MASS) # ginv()

# Location of BUGS:
bugs.directory <- "C:\\WinBUGS14\\"
```

Simulating data

```
# NB: error in distal linear model is always the last column in u
set.seed(1)
# number of individuals
J <- 1000
# number of repeated measures per individual:
n <- 10
# individual-level indicator at level 1 (observation level):
Ind_L1 <- rep(1:J, each = n)
# total number of observations:
N <- J * n

# design matrix for fixed part of mean function for
# within-individual outcome, y1;
# contains constant of ones and a covariate (randomly-drawn
# from a standard Normal distribution):
X_y1_mu <- cbind(rep(1, times = N),
             rnorm(n = N, mean = 0, sd = 1))

# design matrix for random part of mean function for y1,
# and fixed part of within-individual-variance function for y1 (all same):
X_y1_wiv <- Z_y1_mu <- X_y1_mu

# design matrix for random part of within-individual variance function for y1
# (just constant):
Z_y1_wiv <- X_y1_mu[, 1]

# number of fixed effects (betas) in mean function for y1:
n_b <- ncol(X_y1_mu)
# coefficient values for fixed effects in mean function for y1:
beta <- c(1, 1)

# number of fixed effects (alphas) in mean function for y1:
n_a <- ncol(X_y1_wiv)
# coefficient values for fixed effects in within-individual variance function
# for y1:
alpha <- c(1, 1)

# number of fixed effects (gammas) in mean function for y2:
n_g <- 1
# design matrix for mean function for individual-level
# outcome, y2 (just a constant):
X_y2 <- rep(1, times = J)
# coefficient values for fixed effects in mean function for y2:
gamma <- 0

# total number of individual-level random effects
# (including error individual-level outcome):
n_u_total <- 4

# covariance matrix for individual-level random effects:
sigma2u <- matrix(c(0.5, 0.2, 0.08, 0.1,
```

```
                    0.2, 0.3, 0.07, 0.05,
                    0.08, 0.07, 0.12, 0.05,
                    0.1, 0.05, 0.05, 0.25),
                  nrow = n_u_total)

# generate random effects at the individual-level:
u <- MASS::mvrnorm(n = J,
                   mu = rep(0, times = n_u_total),
                   Sigma = sigma2u)
# expand out to level 1:
u_long <- u[Ind_L1, ]

# generate fixed and random part (at level 2) of model for mean of y1:
fixpart_y1_mu <- as.vector(X_y1_mu %*% beta)
randpart_y1_mu <- rowSums(Z_y1_mu * u_long[, 1:2])

# generate fixed and random part (at level 2) of model for within-individual
# variance of y1:
fixpart_y1_wiv <- as.vector(X_y1_wiv %*% alpha)
randpart_y1_wiv <- Z_y1_wiv * u_long[, 3]

# generate level 1 residuals:
log_sigma2_e <- fixpart_y1_wiv + randpart_y1_wiv
sigma_e <- sqrt(exp(log_sigma2_e))
e <- rnorm(n = N, mean = 0, sd = sigma_e)

# generate y1:
y1 <- fixpart_y1_mu + randpart_y1_mu + e

# generate y2:
fixpart_y2 <- X_y2 * gamma
y2 <- fixpart_y2 + u[, 4]

# Create z: matrix with JUST observed data (y2) in one column,
# leaving column of NAs, corresponding to random effects,
# to be estimated in model; this is passed in as data:
z <- matrix(, nrow = J, ncol = n_u_total)
z[, 4] <- y2
```

Model A: freely-covarying covariance matrix

Equation

$$y_{1ij} = \beta_0 + \beta_1 \mathrm{x}_{1ij} + u_{0j} + + u_{1j}\mathrm{x}_{1ij} + e_{ij}$$

$$y_{2j} = \gamma_0 + u_{3j}$$

$$\begin{pmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \\ u_{3j} \end{pmatrix} \sim \mathrm{N}\left[\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_0^2 & & & \\ \sigma_{01} & \sigma_1^2 & & \\ \sigma_{02} & \sigma_{12} & \sigma_2^2 & \\ \sigma_{03} & \sigma_{13} & \sigma_{23} & \sigma_3^2 \end{pmatrix} \right]$$

$$e_{ij} \sim \mathrm{N}(0, \sigma^2_{eij}), \quad \ln(\sigma^2_{eij}) = \alpha_0 + \alpha_1 \mathrm{x}_{1ij} + u_{2j}$$

Specifying data

```
# covariate:
x1 <- X_y1_mu[, 2]
# for inverse Wishart prior for covariance matrix
# (specified with df equal to order of matrix, i.e. 'minimally informative'):
R2 <- 2 * sigma2u

bugsDat <- list(N = N,
                J = J,
                R2 = R2,
                z = z,
                Ind_L1 = Ind_L1,
                y1 = y1,
                x1 = x1
)
```

Initial values

```
# precision parameter for covariance matrix:
tau.u <- MASS::ginv(sigma2u)

# random effects:
z <- u
z[, 4] <- NA

# different initial values for each chain:
CV <- 0.1
inits_function <- function(){
  list(
    gamma = rnorm(n = length(gamma), mean = gamma, sd = abs(CV * gamma)),
    beta = rnorm(n = length(beta), mean = beta, sd = abs(CV * beta)),
    alpha = rnorm(n = length(alpha), mean = alpha, sd = abs(CV * alpha)),
    z = jitter(z, amount = 2),
    tau.u = jitter(tau.u)
  )
```

```
}

inits1 <- inits_function()
inits2 <- inits_function()
inits3 <- inits_function()

inits <- list(inits1,
              inits2,
              inits3)
```

Specifying the model

NB: Assuming this BUGS model is saved in its own file.

This model is adapted from BUGS output from a random slope model fitted in MLwiN (Browne 2018; Charlton et al. 2018).

```
writeLines(readLines("free_cov.bugs"))
```

```
## model {
##   # Repeated measure
##   for (i in 1:N) {
##     y1[i] ~ dnorm(mu[i], tau[i])
##     mu[i] <- beta[1] + beta[2] * x1[i] + z[Ind_L1[i], 1] +
##       z[Ind_L1[i], 2] * x1[i]
##     tau[i] <- 1 / exp(alpha[1] + alpha[2] * x1[i] + z[Ind_L1[i], 3])
##   }
##   # Higher level definitions
##   for (j in 1:J) {
##     z[j, 1:4] ~ dmnorm(mu.u[j, 1:4], tau.u[1:4, 1:4])
##   }
##   # Priors for fixed effects
##   for (k in 1:2) {
##     beta[k] ~ dflat()
##     alpha[k] ~ dflat()
##   }
##   gamma ~ dflat()
##   # Priors for random terms
##   for (j in 1:J){
##     for (k in 1:3){
##       mu.u[j, k] <- 0
##     }
##     mu.u[j, 4] <- gamma
##   }
##   tau.u[1:4, 1:4] ~ dwish(R2[1:4, 1:4], 4)
##   sigma2.u[1:4, 1:4] <- inverse(tau.u[,])
## }
```

Fitting model

```
parameters.to.save <- c("beta",
                        "alpha",
```

```
                        "gamma",
                        "sigma2.u")
n.chains <- 3
n.thin <- 1
n.iter <- 10000
n.burnin <- 5000
seed = 1

free_cov_model_fit <-
  bugs(
    data = bugsDat,
    inits = inits,
    parameters.to.save = parameters.to.save,
    model.file = "free_cov.bugs",
    n.chains = n.chains,
    n.iter = n.iter,
    n.burnin = n.burnin,
    n.thin = n.thin,
    DIC = FALSE,
    bugs.directory = bugs.directory,
    program = "WinBUGS",
    working.directory = NULL,
    clearWD = TRUE,
    bugs.seed = seed,
    save.history = FALSE
  )
```

Inspecting results

```
print(free_cov_model_fit, digits.summary = 3)
# Inference for Bugs model at "free_cov.bugs", fit using WinBUGS,
#  3 chains, each with 10000 iterations (first 5000 discarded)
#  n.sims = 15000 iterations saved
#               mean    sd   2.5%   25%   50%   75% 97.5%  Rhat n.eff
# beta[1]      1.015 0.029  0.959 0.996 1.015 1.034 1.071 1.002  3000
# beta[2]      1.014 0.025  0.964 0.997 1.014 1.031 1.063 1.005   460
# alpha[1]     0.999 0.020  0.960 0.985 0.999 1.013 1.037 1.001 15000
# alpha[2]     1.001 0.017  0.968 0.990 1.001 1.013 1.034 1.003  1100
# gamma        0.016 0.016 -0.016 0.005 0.016 0.027 0.047 1.001  6600
# sigma2.u[1,1] 0.455 0.037  0.387 0.430 0.454 0.480 0.529 1.004   580
# sigma2.u[1,2] 0.202 0.025  0.155 0.185 0.201 0.218 0.253 1.002  1900
# sigma2.u[1,3] 0.072 0.018  0.038 0.060 0.072 0.084 0.106 1.003  1400
# sigma2.u[1,4] 0.068 0.015  0.039 0.057 0.067 0.078 0.096 1.001  8300
# sigma2.u[2,1] 0.202 0.025  0.155 0.185 0.201 0.218 0.253 1.002  1900
# sigma2.u[2,2] 0.316 0.025  0.270 0.299 0.315 0.332 0.367 1.002  2400
# sigma2.u[2,3] 0.063 0.015  0.033 0.052 0.063 0.073 0.093 1.001  5200
# sigma2.u[2,4] 0.033 0.013  0.009 0.024 0.033 0.041 0.058 1.002  2400
# sigma2.u[3,1] 0.072 0.018  0.038 0.060 0.072 0.084 0.106 1.003  1400
# sigma2.u[3,2] 0.063 0.015  0.033 0.052 0.063 0.073 0.093 1.001  5200
# sigma2.u[3,3] 0.117 0.017  0.086 0.105 0.116 0.128 0.154 1.006  2000
# sigma2.u[3,4] 0.054 0.010  0.035 0.048 0.054 0.060 0.073 1.002  1600
# sigma2.u[4,1] 0.068 0.015  0.039 0.057 0.067 0.078 0.096 1.001  8300
```

```
# sigma2.u[4,2] 0.033 0.013  0.009 0.024 0.033 0.041 0.058 1.002  2400
# sigma2.u[4,3] 0.054 0.010  0.035 0.048 0.054 0.060 0.073 1.002  1600
# sigma2.u[4,4] 0.253 0.011  0.231 0.245 0.252 0.260 0.276 1.001  7400
#
# For each parameter, n.eff is a crude measure of effective sample size,
# and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

row1 <- c(
  free_cov_model_fit$summary["sigma2.u[1,1]", "mean"],
  free_cov_model_fit$summary["sigma2.u[1,2]", "mean"],
  free_cov_model_fit$summary["sigma2.u[1,3]", "mean"],
  free_cov_model_fit$summary["sigma2.u[1,4]", "mean"]
)
row2 <- c(
  free_cov_model_fit$summary["sigma2.u[2,1]", "mean"],
  free_cov_model_fit$summary["sigma2.u[2,2]", "mean"],
  free_cov_model_fit$summary["sigma2.u[2,3]", "mean"],
  free_cov_model_fit$summary["sigma2.u[2,4]", "mean"]
)
row3 <- c(
  free_cov_model_fit$summary["sigma2.u[3,1]", "mean"],
  free_cov_model_fit$summary["sigma2.u[3,2]", "mean"],
  free_cov_model_fit$summary["sigma2.u[3,3]", "mean"],
  free_cov_model_fit$summary["sigma2.u[3,4]", "mean"]
)
row4 <- c(
  free_cov_model_fit$summary["sigma2.u[4,1]", "mean"],
  free_cov_model_fit$summary["sigma2.u[4,2]", "mean"],
  free_cov_model_fit$summary["sigma2.u[4,3]", "mean"],
  free_cov_model_fit$summary["sigma2.u[4,4]", "mean"]
)
free_covmatrix <- matrix(c(row1, row2, row3, row4), nrow = 4)
```

Calculate regression coefficients

Calculate values for regression coefficients, in the model for `y2`, for the random effects for the within-individual outcome (i.e. the estimated change in `y2` per one-unit change in the corresponding random effect) from the freely-covarying individual-level covariance matrix fitted in the model above. See Supplemental Material in Macdonald-Wallis et al. (2012) for discussion of equations used below.

Here `I` = intercept, `S` = slope, `W` = within-individual variance.

```
cov_IS <- free_covmatrix[1, 2]
cov_IW <- free_covmatrix[1, 3]
cov_ID <- free_covmatrix[1, 4]

cov_SW <- free_covmatrix[2, 3]

cov_SD <- free_covmatrix[2, 4]
cov_WD <- free_covmatrix[3, 4]

var_I <- free_covmatrix[1, 1]
var_S <- free_covmatrix[2, 2]
```

```r
var_W <- free_covmatrix[3, 3]
var_D <- free_covmatrix[4, 4]

D3 <- (var_I * var_S * var_W) +
  (2 * cov_IS * cov_IW * cov_SW) -
  (var_I * ((cov_SW)^2)) -
  (var_S * ((cov_IW)^2)) -
  (var_W * ((cov_IS)^2))

alpha_11 <- (var_S * var_W) - ((cov_SW)^2)
alpha_12 <- (cov_IW * cov_SW) - (var_W * cov_IS)
alpha_13 <- (cov_IS * cov_SW) - (var_S * cov_IW)

alpha_21 <- (cov_IW * cov_SW) - (var_W * cov_IS)
alpha_22 <- (var_I * var_W) - ((cov_IW)^2)
alpha_23 <- (cov_IS * cov_IW) - (var_I * cov_SW)

alpha_31 <- (cov_IS * cov_SW) - (var_S * cov_IW)
alpha_32 <- (cov_IS * cov_IW) - (var_I * cov_SW)
alpha_33 <- (var_I * var_S) - ((cov_IS)^2)

gamma_1 <- ((alpha_11 * cov_ID) + (alpha_12 * cov_SD) + (alpha_13 * cov_WD)) / D3
gamma_1
# [1] 0.101114

gamma_2 <- ((alpha_21 * cov_ID) + (alpha_22 * cov_SD) + (alpha_23 * cov_WD)) / D3
gamma_2
# [1] -0.04431129

gamma_3 <- ((alpha_31 * cov_ID) + (alpha_32 * cov_SD) + (alpha_33 * cov_WD)) / D3
gamma_3
# [1] 0.4232283
```

Model B: random effects as exposures

Here we fit a model to the same data, but with the random effects from the within-individual outcome fitted as exposures for the individual-level outcome.

Equation

$$y_{1ij} = \beta_0 + \beta_1 \mathrm{x}_{1ij} + u_{0j} + u_{1j}\mathrm{x}_{1ij} + e_{ij}$$

$$y_{2j} = \gamma_0 + \gamma_1 u_{0j} + \gamma_2 u_{1j} + \gamma_3 u_{2j} + u_{3j}$$

$$\begin{pmatrix} u_{0j} \\ u_{1j} \\ u_{2j} \\ u_{3j} \end{pmatrix} \sim \mathrm{N}\left[\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_0^2 & & & \\ \sigma_{01} & \sigma_1^2 & & \\ \sigma_{02} & \sigma_{12} & \sigma_2^2 & \\ 0 & 0 & 0 & \sigma_3^2 \end{pmatrix} \right]$$

$$e_{ij} \sim \mathrm{N}(0, \sigma^2_{eij}), \quad \ln(\sigma^2_{eij}) = \alpha_0 + \alpha_1 \mathrm{x}_{1ij} + u_{2j}$$

Specifying data

```
sigma2u_cond <- sigma2u[1:3, 1:3]

R2_cond <- 2 * sigma2u_cond

bugsDat <- list(
  N = N,
  J = J,
  R2_cond = R2_cond,
  Ind_L1 = Ind_L1,
  y1 = y1,
  x1 = x1,
  y2 = y2
  )
```

BUGS:inits

```
sigma2.y2 <- sigma2u[4, 4]

gamma_cond <- c(gamma, rep(0, times = 3))

tau.u_cond <- MASS::ginv(sigma2u_cond)

tau.y2 <- 1 / sigma2.y2

u_cond <- u[, 1:3]

CV <- 0.1

inits_function <- function(){
  list(
    gamma_cond = rnorm(n = length(gamma_cond),
                       mean = gamma_cond,
                       sd = abs(CV * gamma_cond)),
    beta = rnorm(n = length(beta), mean = beta, sd = abs(CV * beta)),
    alpha = rnorm(n = length(alpha), mean = alpha, sd = abs(CV * alpha)),
    u_cond = jitter(u_cond, amount = 2),
    tau.u_cond = jitter(tau.u_cond),
    tau.y2 = jitter(tau.y2)
  )
}

inits1 <- inits_function()
inits2 <- inits_function()
inits3 <- inits_function()

inits <- list(inits1,
              inits2,
              inits3)
```

Specifying the model

NB: Assuming this BUGS model is saved in its own file.

This model is adapted from BUGS output from a random slope model fitted in MLwiN (Browne 2018; Charlton et al. 2018).

```
writeLines(readLines("conditional.bugs"))
```

```
## Warning in readLines("conditional.bugs"): incomplete final line found on
## 'conditional.bugs'
```

```
## model {
##   # Repeated measure
##   for (i in 1:N) {
##     y1[i] ~ dnorm(mu[i], tau[i])
##     mu[i] <- beta[1] + beta[2] * x1[i] + u_cond[Ind_L1[i], 1] +
##       u_cond[Ind_L1[i], 2] * x1[i]
##     tau[i] <- 1 / exp(alpha[1] + alpha[2] * x1[i] + u_cond[Ind_L1[i], 3])
##   }
##   # Individual-level outcome
##   for (j in 1:J) {
##     y2[j] ~ dnorm(mu.y2[j], tau.y2)
##     mu.y2[j] <- gamma_cond[1] + gamma_cond[2] * u_cond[j, 1] +
##       gamma_cond[3] * u_cond[j, 2] + gamma_cond[4] * u_cond[j, 3]
##   }
##   ## Higher level definitions
##   for (j in 1:J) {
##     u_cond[j, 1:3] ~ dmnorm(zero2[1:3], tau.u_cond[1:3, 1:3])
##   }
##   # Priors for fixed effects
##   for (k in 1:2) {
##     beta[k] ~ dflat()
##     alpha[k] ~ dflat()
##   }
##   for (k in 1:4) {
##     gamma_cond[k] ~ dflat()
##   }
##   # Priors for random terms
##   for (i in 1:3) {
##     zero2[i] <- 0
##   }
##   tau.u_cond[1:3,1:3] ~ dwish(R2_cond[1:3, 1:3],3)
##   sigma2.u[1:3,1:3] <- inverse(tau.u_cond[,])
##   tau.y2 ~ dgamma(0.001, 0.001)
##   sigma2.y2 <- 1 / tau.y2
##
## }
```

BUGS: chains, monitoring, saving

```
parameters.to.save <- c("beta", "alpha", "gamma_cond", "sigma2.u", "sigma2.y2")
n.chains <- 3
n.thin <- 1
```

```
n.iter <- 10000
n.burnin <- 5000
seed = 1

# BUGS: run model

cond_model_fit <-
  bugs(
    data = bugsDat,
    inits = inits,
    parameters.to.save = parameters.to.save,
    model.file = "conditional.bugs",
    n.chains = n.chains,
    n.iter = n.iter,
    n.burnin = n.burnin,
    n.thin = n.thin,
    DIC = FALSE,
    bugs.directory = bugs.directory,
    program = "WinBUGS",
    working.directory = NULL,
    clearWD = TRUE,
    bugs.seed = seed,
    save.history = FALSE
  )
```

Inspecting results

```
 print(cond_model_fit, digits.summary = 3)
# Inference for Bugs model at "conditional.bugs", fit using WinBUGS,
#  3 chains, each with 10000 iterations (first 5000 discarded)
#  n.sims = 15000 iterations saved
#                  mean    sd   2.5%    25%    50%    75% 97.5%  Rhat n.eff
# beta[1]        1.016 0.029  0.959  0.996  1.016  1.035 1.070 1.003  1000
# beta[2]        1.015 0.025  0.966  0.998  1.015  1.032 1.062 1.016   140
# alpha[1]       0.999 0.020  0.960  0.986  0.999  1.013 1.038 1.002  1800
# alpha[2]       1.001 0.018  0.966  0.989  1.001  1.013 1.036 1.002  5000
# gamma_cond[1]  0.016 0.016 -0.016  0.006  0.016  0.027 0.048 1.001  3800
# gamma_cond[2]  0.101 0.040  0.020  0.074  0.102  0.128 0.179 1.003  1100
# gamma_cond[3] -0.044 0.051 -0.146 -0.079 -0.043 -0.010 0.053 1.001  3800
# gamma_cond[4]  0.426 0.106  0.236  0.351  0.419  0.493 0.651 1.008   490
# sigma2.u[1,1]  0.455 0.037  0.385  0.430  0.454  0.479 0.530 1.002  1800
# sigma2.u[1,2]  0.202 0.025  0.153  0.185  0.201  0.219 0.254 1.006   440
# sigma2.u[1,3]  0.071 0.018  0.036  0.059  0.071  0.083 0.106 1.008   280
# sigma2.u[2,1]  0.202 0.025  0.153  0.185  0.201  0.219 0.254 1.006   440
# sigma2.u[2,2]  0.316 0.026  0.267  0.298  0.315  0.333 0.369 1.003  1000
# sigma2.u[2,3]  0.062 0.015  0.033  0.052  0.062  0.072 0.092 1.002  1800
# sigma2.u[3,1]  0.071 0.018  0.036  0.059  0.071  0.083 0.106 1.008   280
# sigma2.u[3,2]  0.062 0.015  0.033  0.052  0.062  0.072 0.092 1.002  1800
# sigma2.u[3,3]  0.117 0.016  0.086  0.106  0.117  0.128 0.151 1.006   650
# sigma2.y2      0.224 0.012  0.201  0.216  0.224  0.232 0.248 1.002  1500
#
# For each parameter, n.eff is a crude measure of effective sample size,
```

```
# and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
```

Compare estimates of regression coefficients from the two models

Comparing estimates of regression coefficients from the two parameterisations illustrates their equivalence:

```
# Select regression coefficients for random effects fitted as exposures in model for
# individual-level outcome in the second 'conditional' model:
gamma_estimates <- c(
  cond_model_fit$summary["gamma_cond[1]", "mean"],
  cond_model_fit$summary["gamma_cond[2]", "mean"],
  cond_model_fit$summary["gamma_cond[3]", "mean"],
  cond_model_fit$summary["gamma_cond[4]", "mean"]
)
gamma_estimates[2:4]
# [1]  0.10114596 -0.04445182  0.42582335

# Comparing these to gamma estimates as earlier calculated from
# 'freely-covarying' model:
comparison_table <- matrix(c(gamma_1, gamma_2, gamma_3, gamma_estimates[2:4]), ncol = 2)
colnames(comparison_table) <- c("freely-covarying model", "conditional model")
rownames(comparison_table) <- c("intercept", "slope", "wiv")
round(comparison_table, digits = 2)
#           freely-covarying model conditional model
# intercept                   0.10              0.10
# slope                      -0.04             -0.04
# wiv                         0.42              0.43
```


Browne, W.J. 2018. *MCMC Estimation in Mlwin V3.02*. Centre for Multilevel Modelling, University of Bristol.

Charlton, C., J. Rasbash, W.J. Browne, M. Healy, and B. Cameron. 2018. "MLwiN Version 3.02." Centre for Multilevel Modelling, University of Bristol.

Lunn, D.J., A. Thomas, N. Best, and D. Spiegelhalter. 2000. "WinBUGS — a Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing* 10: 325–37.

Macdonald-Wallis, C., D.A. Lawlor, T. Palmer, and K. Tilling. 2012. "Multivariate Multilevel Spline Models for Parallel Growth Processes: Application to Weight and Mean Arterial Pressure in Pregnancy." *Statistics in Medicine* 31 (26): 3147–64. https://doi.org/10.1002/sim.5385.

Sturtz, Sibylle, Uwe Ligges, and Andrew Gelman. 2005. "R2WinBUGS: A Package for Running Winbugs from R." *Journal of Statistical Software* 12 (3): 1–16. http://www.jstatsoft.org.