GETTING STARTED WITH BAYESIAN GLMM IN
R, SAS, MPLUS, & WINBUGS

OUHSC SCUG Presentation on October 7, 2014

David Bard

Objectives

- Brief intro to generalized linear mixed models (GLMM)
- Even briefer intro to Bayesian estimation of random effect models
- Intro to the MCMCglmm package in R
- Comparison of MCMCglmm and WinBUGS/OpenBUGS, Mplus, and SAS proc mcmc

Unpacking GLMM

- GL~~M~~M = *GizedLM*
  - Traditional General Linear Model (GLM)
    - $y_i = x_i'\beta + \varepsilon_i\ ;\quad \varepsilon_i \sim N(0, \sigma^2)$
    - $\mu_i = x_i'\beta$
  - General-*ized* LM (McCullagh & Nelder, 1989)
    - $\eta_i = x_i'\beta$
    - $g(\mu_i) = x_i'\beta$
    - $E(y_i) = \mu_i = g^{-1}(x_i'\beta)\ ; y_i \sim exponential\ family$
    - $\text{var}(y_i) = \frac{\phi \mathrm{V}(\mu_i)}{w_i}$

Unpacking GLMM continued

- G-izedLM + random effects = GLMM
  - *Mixes* in some random effects with GizedLM fixed effects (Breslow & Clayton, 1993)
    - $\eta_{ij} = x'_{ij}\beta + z'_{ij}\gamma_j$
    - $g(\mu_{ij}) = x'_{ij}\beta + z'_{ij}\gamma_j$
    - $y_{ij} = g^{-1}(x'_{ij}\beta + z'_{ij}\gamma_j) + \varepsilon_{ij}\ ;\ \gamma \sim N(0, \Sigma_G), \varepsilon \sim N(0, \Sigma_R)$

Bayesian Inference for GLMM

- Inference for GLMM
  - Frequentist Likelihood Approach:
    - $\Pr(y|\beta,\gamma,\Sigma_G,\Sigma_R)$
  - Bayesian approach:
    - $\Pr(\beta,\gamma,\Sigma_G,\Sigma_R|y) \propto \Pr(y|\beta,\gamma,\Sigma_G,\Sigma_R)\Pr(\beta,\gamma,\Sigma_G,\Sigma_R)$
- Markov Chain Monte Carlo (MCMC)
  - Before MCMC, joint posterior distribution analytically intractable

GLMM Inference in R

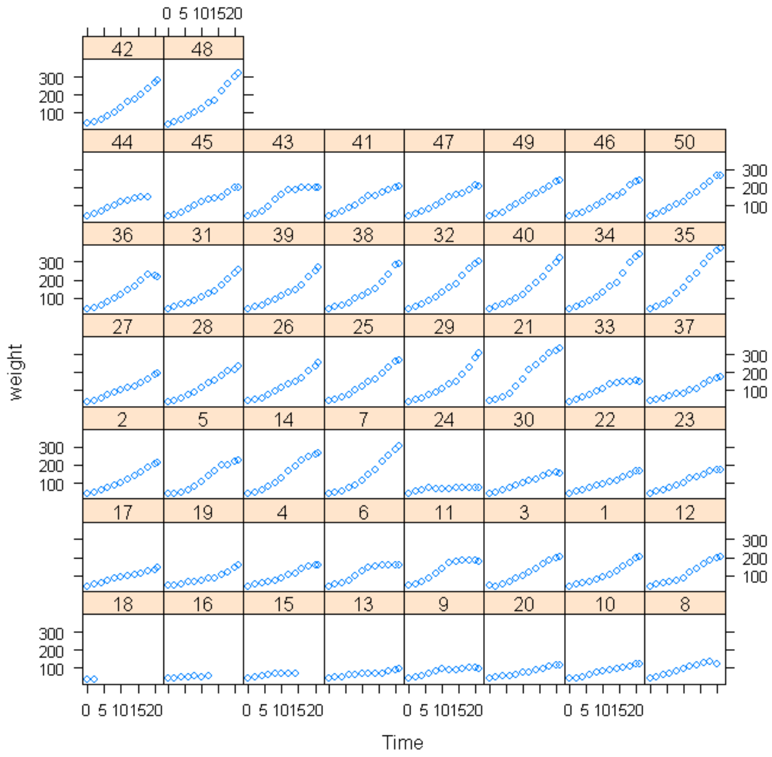
- Several self-contained packages available
  - See http://glmm.wikidot.com/faq
- I'll focus on one today, MCMCglmm
  - Markov chain Monte Carlo Sampler for Multivariate Generalised Linear Mixed Models with special emphasis on correlated random effects arising from pedigrees and phylogenies (Hadfield 2010).
  - http://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf

MCMCglmm function

- MCMCglmm(fixed, random=NULL, rcov=~units, family="gaussian", mev=NULL, data,start=NULL, prior=NULL, tune=NULL, pedigree=NULL, nodes="ALL",scale=TRUE, nitt=13000, thin=10, burnin=3000, pr=FALSE,pl=FALSE, verbose=TRUE, DIC=TRUE, singular.ok=FALSE, saveX=TRUE, saveZ=TRUE, saveXL=TRUE, slice=FALSE, ginverse=NULL)

Example Dataset

- data(ChickWeight)
  - The ChickWeight data frame has 578 rows and 4 columns from an experiment on the effect of diet on early growth of chicks.
- xyplot(weight ~ Time | Chick, data = ChickWeight)



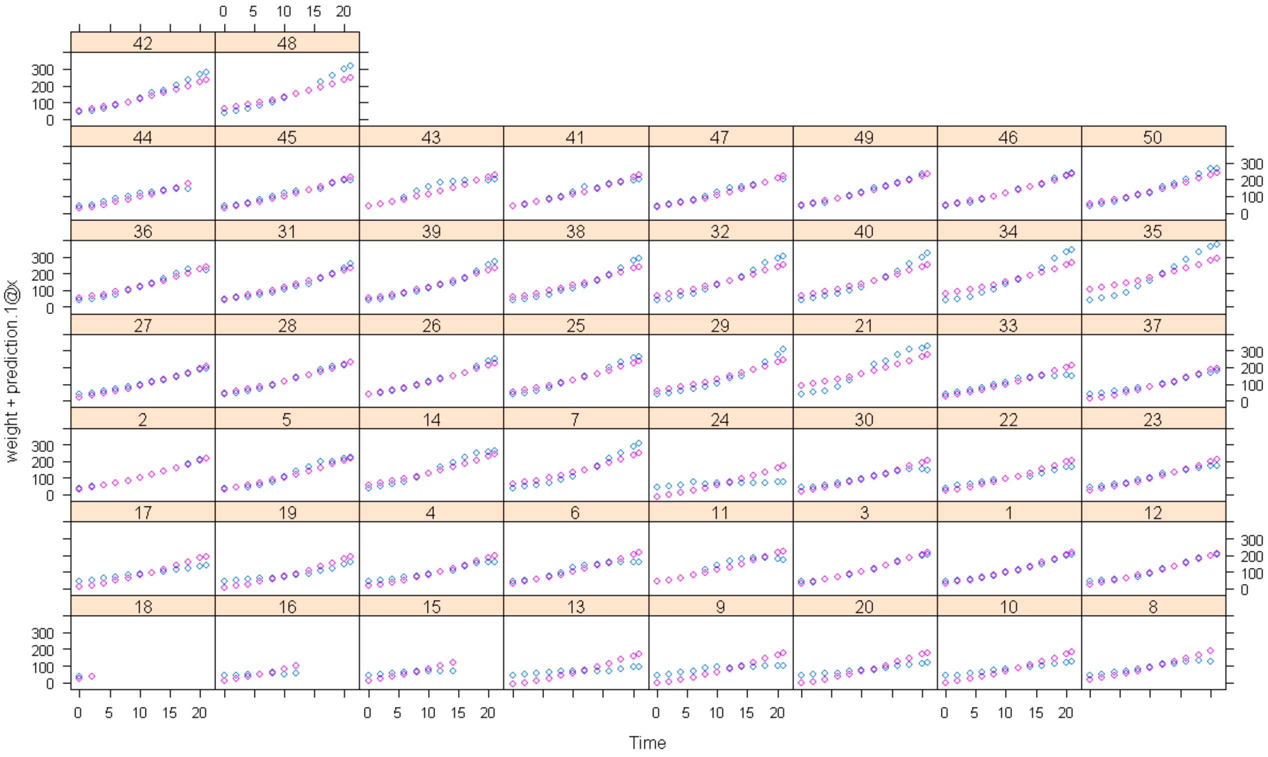
MCMCglmm fit to ChickWeight
(See Chpt 4 of Hadfield course notes)

- Fit simple 2nd order polynomial with a random intercept
- Priors
  - prior.m4a.1 <- list(R = list(V = 1e-07, n = -2),G = list(G1 = list(V = 1, n = 1)))
  - Prior for Σ_R is Wishart(V=0,nu=-2)
    - "The inverse gamma is a special case of the inverse Wishart, although it is parametrised using shape and scale, where nu = 2 $*$ shape and V = scale/shape (or shape = nu/2 and scale = nu$*$V/2)."
  - Prior for Σ_G is Wishart(1,1)

MCMCglmm fit to ChickWeight

- Model statement
  - m4a.1 <- MCMCglmm(weight ~ Diet + poly(Time, 2,raw = TRUE), random = ~Chick, data = ChickWeight,verbose = FALSE, pr = TRUE, prior = prior.m4a.1,saveX = TRUE, saveZ = TRUE)
- Visualize model predictions
  - W.1<-cBind(m4a.1$X, m4a.1$Z) # note X and Z are sparse so use cBind intstead of cbind
  - prediction.1<-W.1%*%posterior.mode(m4a.1$Sol)
  - xyplot(weight+prediction.1@x~Time|Chick, data=ChickWeight)

MCMCglmm fit to ChickWeight



MCMCglmm fit to ChickWeight

- prior.m4a.3 <- list(R = list(V = 1, n = 0.002),G = list(G1 = list(V = diag(3), n = 3)))
- m4a.3 <- MCMCglmm(weight ~ Diet + poly(Time, 2,raw = TRUE), random = ~us(1 + poly(Time, 2,raw = TRUE)):Chick,data = ChickWeight, verbose = FALSE,pr = TRUE, prior = prior.m4a.3, saveX = TRUE,saveZ = TRUE)

MCMCglmm Output for Quadratic Random Effect Model

Iterations = 3001:12991 Thinning interval = 10 Sample size = 1000

DIC: 3932.687

G-structure: ~us(1 + poly(Time, 2, raw = TRUE)):Chick

	post.mean	l-95% CI	u-95% CI	eff.samp
(Intercept):(Intercept).Chick	28.6006	10.80540	48.4596	591.4
poly(Time, 2)1:(Intercept).Chick	-17.8679	-28.46534	-9.0170	897.1
poly(Time, 2)2:(Intercept).Chick	0.7339	0.08795	1.3298	1000.0
(Intercept):poly(Time, 2)1.Chick	-17.8679	-28.46534	-9.0170	897.1
poly(Time, 2)1:poly(Time, 2)1.Chick	12.0861	6.97558	17.5365	1286.4
poly(Time, 2)2:poly(Time, 2)1.Chick	-0.5198	-0.91008	-0.1562	1000.0
(Intercept):poly(Time, 2)2.Chick	0.7339	0.08795	1.3298	1000.0
poly(Time, 2)1:poly(Time, 2)2.Chick	-0.5198	-0.91008	-0.1562	1000.0
poly(Time, 2)2:poly(Time, 2)2.Chick	0.1209	0.07670	0.1707	891.2

R-structure: ~units

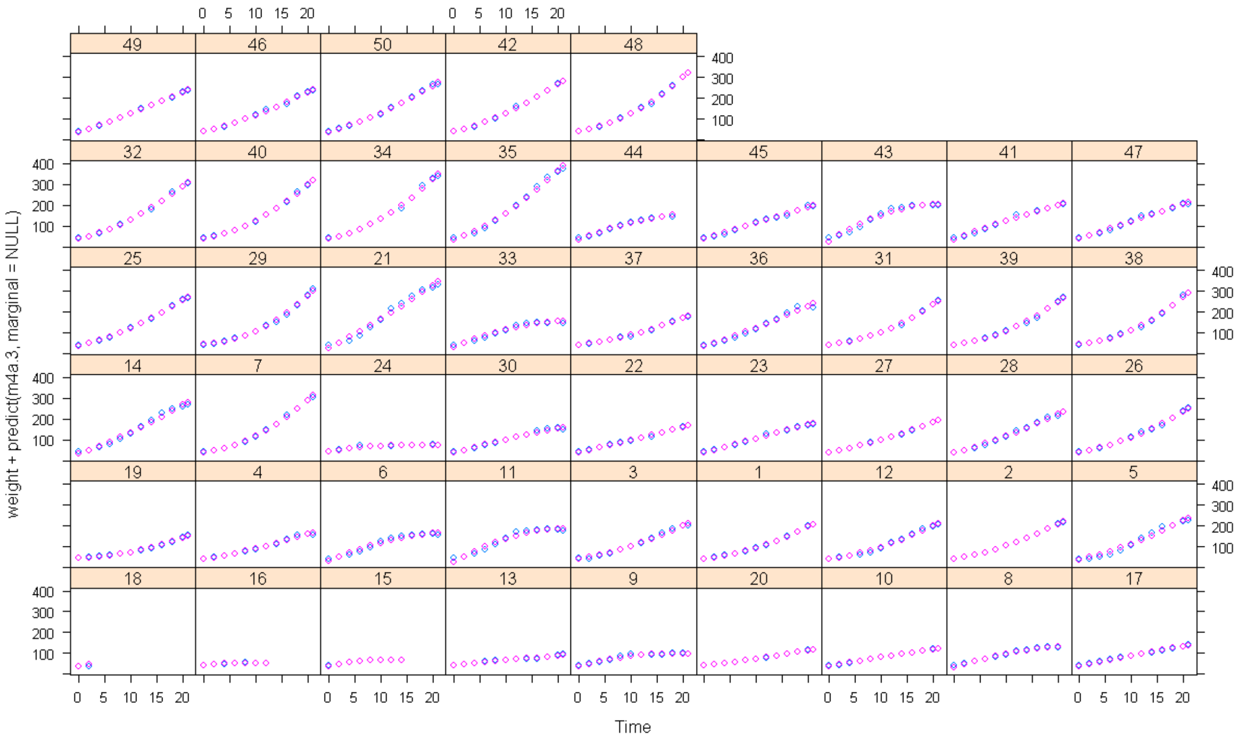
	post.mean	l-95% CI	u-95% CI	eff.samp
units	43.9	39.1	49.95	1000

Location effects: weight ~ Diet + poly(Time, 2, raw = TRUE)

	post.mean	l-95% CI	u-95% CI	eff.samp	pMCMC	
(Intercept)	36.08622	33.58255	38.33466	1000.0	<0.001	***
Diet2	1.42484	-1.63380	4.33774	1110.6	0.350	
Diet3	1.38532	-1.76403	4.12476	1000.0	0.378	
Diet4	3.94954	0.93236	7.19609	521.6	0.012	*
poly(Time, 2)1	5.92471	4.96057	6.93158	1000.0	<0.001	***
poly(Time, 2)2	0.11233	0.01541	0.21158	1000.0	0.034	*

---Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

MCMCglmm fit to ChickWeight




```
> m4a.1$DIC
[1] 5525.262
> m4a.3$DIC
[1] 3932.774
```

REML fit to ChickWeight

m5a.3.REML <- lmer(weight ~ Diet + poly(Time, 2, raw = TRUE) + (poly(Time, 2, raw = TRUE) | Chick), data = ChickWeight)
summary(m5a.3.REML)

Linear mixed model fit by REML ['lmerMod']
Random effects:

Groups	Name		Variance	Std.Dev.	Corr
Chick	(Intercept)		31.15433	5.5816	
	poly(Time, 2, raw = TRUE)1	12.47343	3.5318	-1.00	
	poly(Time, 2, raw = TRUE)2	0.05408	0.2325	0.64	-0.64
Residual			43.73500	6.6132	

Number of obs: 578, groups: Chick, 50

Fixed effects:

	Estimate	Std. Error	t value	
(Intercept)	36.07142	1.23192	29.281	
Diet2	1.44948	1.40532	1.031	
Diet3	1.36360	1.40532	0.970	
Diet4	4.16271	1.40546	2.962	
poly(Time, 2, raw = TRUE)1		5.90475	0.52667	11.212
poly(Time, 2, raw = TRUE)2		0.11580	0.03406	3.399

WinBUGS code

```
require(R2WinBUGS)
model <- function(){
for (i in 1:n){
y[i] ~ dnorm (y.hat[i], tau.y)
y.hat[i] <- inprod(B[county[i],],Z[i,]) + inprod(beta[],X[i,])
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)

for (l in 1:3){beta[l]~dnorm(0,1.0E-6)}

for (j in 1:J){
for (k in 1:K){
B[j,k] <- xi[k]*B.raw[j,k]
}
B.raw[j,1:K] ~ dmnorm (mu.raw[], Tau.B.raw[,])
}
```

```
for (k in 1:K){
mu[k] <- xi[k]*mu.raw[k]
mu.raw[k] ~ dnorm (0, .0001)
xi[k] ~ dunif (0, 100)
}
Tau.B.raw[1:K,1:K] ~ dwish(W[,], df)
df <- K+1
Sigma.B.raw[1:K,1:K] <- inverse(Tau.B.raw[,])
for (k in 1:K){
for (k.prime in 1:K){
rho.B[k,k.prime] <- Sigma.B.raw[k,k.prime]/
sqrt(Sigma.B.raw[k,k]*Sigma.B.raw[k.prime,k.prime])
}
sigma.B[k] <- abs(xi[k])*sqrt(Sigma.B.raw[k,k])
}

}
```

```
model.file <-
file.path(tempdir(),"model.txt")
write.model(model, model.file)
```

WinBUGS code

```
missObs <- !apply(is.na(ChickWeight[
,c("Diet","Time","Chick")]),1,base::any)
bugDat <- ChickWeight[missObs,]
y <- bugDat$weight
dsgnMat <- as.matrix(model.matrix(weight ~
Diet+poly(Time, 2,raw = TRUE),data=bugDat))
#cbind(bugDat[,c("Diet","Time")],bugDat$Time
^2))
Z <- dsgnMat[,-c(2:4)]
X <- dsgnMat[,c(2:4)]
county <- sapply(bugDat$Chick,function(x)
which(unique(bugDat$Chick) %in% x))
J <- length(unique(county)) #nrow(X)
K <- ncol(X)
```

```
n <- nrow(bugDat)
W <- diag (3)
bugs.data <- list ("n", "J", "K", "Z",
"X", "y", "county", "W")

bugs.inits <- function (){
list (B.raw=array(rnorm(J*K), c(J,K)),
mu.raw=rnorm(K), sigma.y=runif(1),
Tau.B.raw=rwish(K+1,diag(K)),
xi=runif(K), beta=rnorm(3))
}

bugs.parameters <- c ("B", "mu",
"beta", "sigma.y", "sigma.B", "rho.B")
```

```
bugsMod <- R2WinBUGS:::bugs (bugs.data, bugs.inits, bugs.parameters, model.file,n.chains=3, n.iter=2000, n.thin=10,
#n.burnin=1000,
bugs.directory="F:\\Program Files\\WinBUGS14", clearWD=TRUE, debug=TRUE )
```

WinBUGS Output

- > bugsMod$summary[!grepl("^B\\[",row.names(bugsMod$summary)),]

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mu[1]	34.7888000	2.63619542	29.55408630	33.96499890	35.05500	36.1349990	38.1757471	1.232519	90
mu[2]	5.9312100	0.55340567	4.93427295	5.52424985	5.92450	6.3149991	7.0219244	1.013821	130
mu[3]	0.1170932	0.03387847	0.05346756	0.09427499	0.11895	0.1401245	0.1827151	1.006368	300
beta[1]	3.0466125	4.11195347	-1.03070000	1.24025000	2.45350	3.9942500	12.1322500	1.233420	73
beta[2]	4.1255903	4.67322899	-0.04247700	1.99250000	3.44450	5.0857500	12.5910000	1.232081	120
beta[3]	5.6563383	3.55086906	0.39663750	3.93650000	5.44800	6.8557500	12.1310000	1.176267	300
sigma.y	6.6822533	0.21928991	6.25432402	6.53349856	6.69550	6.8352500	7.0731989	1.009611	180
sigma.B[1]	6.8031400	1.59958897	4.78300732	5.91849994	6.63600	7.3562497	10.7334123	1.200988	17
sigma.B[2]	3.4715533	0.34714864	2.83274560	3.21224974	3.44800	3.7060000	4.2070000	1.004215	260
sigma.B[3]	0.2243653	0.02367226	0.18379497	0.20857489	0.22220	0.2381500	0.2793959	1.004949	300
rho.B[1,1]	1.0000000	0.00000000	1.00000000	1.00000000	1.00000	1.0000000	1.0000000	1.000000	1
rho.B[1,2]	-0.8113147	0.10318901	-0.94616000	-0.88775000	-0.82275	-0.7583750	-0.5609825	1.746399	6
rho.B[1,3]	0.2960798	0.17634359	-0.05806150	0.18640000	0.31335	0.4164000	0.5962300	1.037619	59
rho.B[2,1]	-0.8113147	0.10318901	-0.94616000	-0.88775000	-0.82275	-0.7583750	-0.5609825	1.746399	6
rho.B[2,2]	1.0000000	0.00000000	1.00000000	1.00000000	1.00000	1.0000000	1.0000000	1.000000	1
rho.B[2,3]	-0.5923480	0.10052565	-0.76591000	-0.66230000	-0.59870	-0.5385250	-0.3749425	1.011282	250
rho.B[3,1]	0.2960798	0.17634359	-0.05806150	0.18640000	0.31335	0.4164000	0.5962300	1.037619	59
rho.B[3,2]	-0.5923480	0.10052565	-0.76591000	-0.66230000	-0.59870	-0.5385250	-0.3749425	1.011282	250
rho.B[3,3]	1.0000000	0.00000000	1.00000000	1.00000000	1.00000	1.0000000	1.0000000	1.000000	1
deviance	3832.3566667	17.44449743	3801.00000000	3821.00000000	3831.00000	3844.0000000	3868.5249678	1.051326	40

Mplus code

```
require(MplusAutomation)
mpDat <- as.data.frame(cbind(county,dsgnMat[,-1],y))
head(dsgnMat[,-1])
colnames(mpDat) <-
c(colnames(mpDat)[1:4],"time","time2","weight")

modelStem <- "mpQuad"
mpFiles1 <- mplusObject(
TITLE  = "ChickWeight Quadratic Random Effect;",
VARIABLE = "CLUSTER = county;
WITHIN = Diet2 Diet3 Diet4 time time2;",
ANALYSIS = "Type = twolevel random; Estimator = Bayes;
proc = 4; fbiter = 13000; thin = 10;",
MODEL = "%WITHIN%
s1 | weight on time;
s2 | weight on time2;
weight on Diet2 Diet3 Diet4;
%BETWEEN%
weight with s1 s2;
s1 with s2;",
```

```
OUTPUT = "sampstat; tech1; TECH8;",
PLOT = "TYPE = PLOT2;",
rdata=mpDat,
usevariables =
c("county","Diet2","Diet3","Diet4","time","time2","weight"))

base <- tempdir()
#cat(base,"\n")
mpInput <- paste0(modelStem,".inp")
mpData <- "chkwgtDat"
cd(base,pre="chickwgt",num="Q")
mpModel <-
mplusModeler(mpFiles1,dataout=mpData,modelout=mpIn
put,run=1)
mpModel$results$summaries
mpModel$results$parameters$unstandardized
```

Mplus Output

Mplus.version	Title	AnalysisType	DataType	Estimator	Observations	Parameters	DIC	pD	Filename
7.3	ChickWeight Quadratic Random Effect;	twolevel random	INDIVIDUAL	BAYES	578	13	3925.949	105.498	mpQuad.out

	paramHeader	param	est	posterior_sd	pval	lower_2.5ci	upper_2.5ci	sig	BetweenWithin
1	WEIGHT.ON	DIET2	2.501	1.485	0.048	-0.468	5.407	FALSE	Within
2	WEIGHT.ON	DIET3	3.618	1.640	0.014	0.372	6.809	TRUE	Within
3	WEIGHT.ON	DIET4	5.466	1.575	0.000	2.394	8.544	TRUE	Within
4	Residual	WEIGHT	43.542	2.862	0.000	38.394	49.647	TRUE	Within
5	WEIGHT.WITH	S1	-24.181	7.313	0.000	-42.050	-14.033	TRUE	Between
6	WEIGHT.WITH	S2	0.579	0.372	0.029	-0.018	1.443	FALSE	Between
7	S1.WITH	S2	-0.576	0.199	0.000	-1.058	-0.291	TRUE	Between
8	Means	WEIGHT	35.165	1.399	0.000	32.362	37.906	TRUE	Between
9	Means	S1	5.895	0.581	0.000	4.756	7.036	TRUE	Between
10	Means	S2	0.118	0.037	0.001	0.045	0.190	TRUE	Between
11	Variances	WEIGHT	44.484	15.677	0.000	22.926	83.258	TRUE	Between
12	Variances	S1	14.789	3.781	0.000	9.637	24.082	TRUE	Between
13	Variances	S2	0.061	0.015	0.000	0.040	0.098	TRUE	Between

SAS proc mcmc

```
proc mcmc data=chkwgt nmc=10000 thin=10
outpost=postout
   seed=17 init=random;
   *ods select Parameters REParameters
PostSummaries;
   array theta[3] alpha beta1 beta2;
   array theta_c[3];
   array Sig_c[3,3];
   array mu0[3] (0 0 0);
   array Sig0[3,3] (1000 0 0
                    0 1000 0
                    0 0 1000);
   array S[3,3] (1 0 0
                 0 1 0
                 0 0 1);
parms theta_c {36 5.9 .12} Sig_c {30 0 0 0 15 0 0 0
0.10} var_y { 44 };
   parms d2 d3 d4;
   prior theta_c ~ mvn(mu0, Sig0);
   prior Sig_c ~ iwish(3, S);
   prior var_y ~ igamma(0.001, scale=0.001);
   prior d2 d3 d4 ~ normal(mean = 0, var = 1e6);
   random theta ~ mvn(theta_c, Sig_c)
subject=county;*monitor=(alpha_9 alpha_25 );
   mu = alpha + d2 * diet2 + d3 * diet3 + d4 * diet4 +
beta1 * time + beta2 * time2;
   model weight ~ normal(mu, var=var_y);
run;
```

SAS Output

Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
theta_c1	1000	36.2189	1.0712	35.4513	36.2101	36.9453
theta_c2	1000	5.9098	0.5058	5.5666	5.9097	6.2500
theta_c3	1000	0.1141	0.0390	0.0868	0.1143	0.1397
Sig_c1	1000	22.8357	8.4000	16.4569	21.6119	27.7463
Sig_c2	1000	-15.9086	4.5804	-18.5958	-15.3239	-12.4258
Sig_c3	1000	0.6524	0.2557	0.4721	0.6286	0.8105
Sig_c4	1000	-15.9086	4.5804	-18.5958	-15.3239	-12.4258
Sig_c5	1000	11.7026	2.6942	9.7744	11.3285	13.0261
Sig_c6	1000	-0.4965	0.1676	-0.5916	-0.4830	-0.3784
Sig_c7	1000	0.6524	0.2557	0.4721	0.6286	0.8105
Sig_c8	1000	-0.4965	0.1676	-0.5916	-0.4830	-0.3784
Sig_c9	1000	0.0755	0.0164	0.0638	0.0731	0.0852
var_y	1000	44.1617	3.0411	41.9688	43.9400	46.2062
d2	1000	1.4699	1.3118	0.5995	1.5305	2.4082
d3	1000	1.4207	1.4776	0.3806	1.4395	2.4819
d4	1000	3.2638	1.5504	2.2698	3.1260	4.2531

Posterior Intervals

Parameter	Alpha	Equal-Tail Interval		HPD Interval	
theta_c1	0.050	34.3190	38.3687	34.3133	38.3615
theta_c2	0.050	4.9527	6.8830	4.9186	6.8195
theta_c3	0.050	0.0368	0.1901	0.0348	0.1863
Sig_c1	0.050	9.9307	41.9441	8.6948	39.4294
Sig_c2	0.050	-26.6860	-8.7971	-25.0472	-8.0377
Sig_c3	0.050	0.2342	1.2554	0.1791	1.1490
Sig_c4	0.050	-26.6860	-8.7971	-25.0472	-8.0377
Sig_c5	0.050	7.5476	18.2496	7.0956	17.3317
Sig_c6	0.050	-0.8862	-0.2171	-0.8644	-0.2102
Sig_c7	0.050	0.2342	1.2554	0.1791	1.1490
Sig_c8	0.050	-0.8862	-0.2171	-0.8644	-0.2102
Sig_c9	0.050	0.0492	0.1126	0.0480	0.1096
var_y	0.050	38.8878	50.3790	38.7140	49.9304
d2	0.050	-1.1540	3.9506	-1.1086	3.9761
d3	0.050	-1.4888	4.0987	-1.4468	4.1170
d4	0.050	0.1898	6.5590	0.0331	6.3151

Comparison of parameter estimates

Fixed Effects: Posterior Mean Estimates

	MCMCglmm	SAS	Mplus	WinBUGS	lmer
Intercept	36.09	36.22	35.17	34.79	36.07
Diet2	1.42	1.47	2.50	3.05	1.45
Diet3	1.39	1.42	3.62	4.13	1.36
Diet4	3.95	3.26	5.47	5.66	4.16
linear	5.92	5.91	5.90	5.93	5.90
quadratic	0.11	0.11	0.12	0.12	0.12

Random Effect Variances: Posterior Mean Estimates

	MCMCglmm	SAS	Mplus	WinBUGS	lmer
Intercept	28.60	22.84	44.48	46.24	31.15
Linear	12.09	11.70	14.79	12.04	12.47
Quadratic	0.12	0.08	0.06	0.05	0.05
Residual	43.9	44.16	43.54	44.65	43.74

Why the differences?

- Posterior-mean, median, mode?
  - Not in this instance, but worth paying attention to across software/packages
- Equivalence of convergence achieved?
  - Hard to know for certain, but seems to be equivalent
- Differences in default/recommended prior distributions?
  - Yes. MCMCglmm/SAS, Mplus, and WinBUGS were all different in prior examples
  - Note: MCMCglmm prior below approx. reproduces Mplus output
    - prior_mg1_mplus <- list(R = list(V = 1e-16, n = -2),G = list(G1 = list(V = diag(1e-16,3), n = -4)))

Which prior should I use?

- If your lucky, won’t matter
- If your unlucky, explore the shape of the default distributions before deciding which feels most comfortable to you
  - VisCov Package in R can be helpful
  - MCMCglmm course notes provide overview of prior choice implications; see also Gelman & Hill (2006) chpt 16-17.

Examining MCMC Diagnostics

- MCMCglmm:plot(m4a.3$Sol)
- autocorr(m4a.3$Sol[,1])

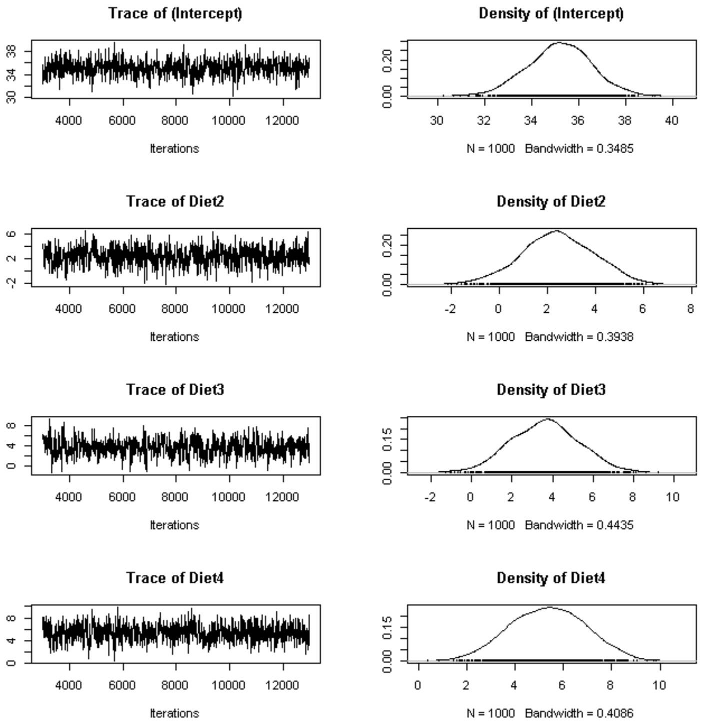
  Lag 0   1.0000000000

  Lag 10 -0.0018215519

  Lag 50   0.0379767609

  Lag 100 0.0122456031

  Lag 500 -0.0002609533
- Uses coda package
  - bcoda <- as.mcmc.list(m4a.3$Sol)



Pros and Cons MCMCglmm

- Pro
  - Straightforward extensions exist for common distributions like the logistic, Poisson, gamma, …
  - Highly efficient estimation time
  - Coda package handy
  - Handles highly complex pedigree and phylogeny data structures
- Con
  - Prediction code underdeveloped
  - Can only run single chain at a time

More MCMCglmm info

- http://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf