# 18 Binomial mixed-effects model (Binomial GLMM)

# -------------------------------------------------------------

# As in a Poisson GLMM, we can also add into a binomial GLM random variation beyond what is stipulated by the binomial distribution. We illustrate this for a slight modification of the Red-backed shrike example from chapter 15. Instead of counting the number of pairs, which naturally leads to the adoption of a Poisson model, we now imagine that we study the reproductive success (success or failure) of its much rarer cousin, the glorious Woodchat shrike. We examine the relationship between precipitation during the breeding season and reproductive success; wet springs are likely to depress the proportion of successful nests. We assemble data from 16 populations studied over 10 years.

# 18.2 Data generation

# ----------------------

# Simulate two independent sets of random effects, i.e., no correlation.

set.seed(18)

nPops <- 16

nYears <- 10

n <- nPops \* nYears

pop <- gl(n = nPops, k = nYears)

# Create a uniform covariate as an index to spring precipitation: -1 denotes little rain and 1 much. This implicit centering of the continuous covariate leads to the desirable interpretation of the intercept as the expected value of the response at the average of the covariate.

prec <- runif(n, -1, 1)

# N is the binomial total, i.e., the number of nesting attempts surveyed in year i.

N <- round(runif(n, 10, 50) )

# We build the design matrix as before and again look at the top 91 rows.

Xmat <- model.matrix(~pop\*prec-1-prec)

print(Xmat[1:91,], dig = 2) # Print top 91 rows

# Next, we choose the parameter values from their respective normal distributions, but first need to pick the values of the associated hyperparameters.

intercept.mean <- 0 # Select hyperparams

slope.mean <- -2

intercept.sd <- 1

slope.sd <- 1

intercept.effects <- rnorm(n = nPops, mean = intercept.mean, sd = intercept.sd)

slope.effects <- rnorm(n = nPops, mean = slope.mean, sd = slope.sd)

all.effects <- c(intercept.effects, slope.effects) # All together

**# Save vector of true parameter values**

truth <- c(intercept.mean=intercept.mean, slope.mean=slope.mean,

intercept.sd=intercept.sd, slope.sd=slope.sd)

# We assemble the counts C by first computing the value of the linear predictor, then applying the inverse-logit transformation and finally integrating binomial noise (where we need N).

lin.pred <- Xmat %\*% all.effects # Value of lin.predictor

exp.p <- exp(lin.pred) / (1 + exp(lin.pred)) # Expected proportion

# For each population, we plot the expected and of the observed, or realized, breeding success against standardized spring precipitation

# The differemce between the two is due to Binomial random variation.

xyplot(exp.p ~ prec | pop, ylab = "Expected woodchat shrike breeding success ", xlab = "Spring precipitation index", main = "Expected breeding success", pch = 16, cex = 1.2, col = rgb(0, 0, 0, 0.4))

C <- rbinom(n = n, size = N, prob = exp.p) # Add binomial variation

xyplot(C/N ~ prec | pop, ylab = "Realized woodchat shrike breeding success ", xlab = "Spring precipitation index", main = "Realized breeding success", pch = 16, cex = 1.2, col = rgb(0, 0, 0, 0.4))

# 18.3 Likelihood analysis with canned functions in R

# -------------------------------------------------------------

library('lme4')

out18.3 <- glmer(cbind(C, N-C) ~ prec + (prec || pop), family = binomial)

summary(out18.3)

**# Compare estimates with truth**

glmer\_est <- c(fixef(out18.3),

sqrt(as.numeric(summary(out18.3)$varcor)))

tmp <- cbind(truth=truth, glmer=glmer\_est)

print(tmp, 4)

# 18.4 Bayesian analysis with JAGS

# -----------------------------------------------------

**# Bundle and summarize data**

str(dataList <- list(C = C, N = N, pop = as.numeric(pop), prec = prec, nPops = nPops, n = n) )

**# Write JAGS model file**

cat(file="model18.4.txt", "

model {

**# Priors**

for (i in 1:nPops){

alpha[i] ~ dnorm(intercept.mean, intercept.tau) # Intercepts

beta[i] ~ dnorm(slope.mean, slope.tau) # Slopes

}

intercept.mean ~ dnorm(0, 0.001) # Hyperparameter for random intercepts

intercept.tau <- pow(intercept.sd, -2)

intercept.sd ~ dunif(0, 10)

slope.mean ~ dnorm(0, 0.001) # Hyperparameter for random slopes

slope.tau <- pow(slope.sd, -2)

slope.sd ~ dunif(0, 10)

**# 'Likelihood'**

for (i in 1:n) {

C[i] ~ dbin(p[i], N[i])

logit(p[i]) <- alpha[pop[i]] + beta[pop[i]]\* prec[i]

}

}

")

**# Function to generate starting values**

inits <- function(){ list(intercept.mean = rnorm(1, 0, 1), slope.mean = rnorm(1, 0, 1))}

**# Parameters to estimate**

params <- c("intercept.mean", "slope.mean", "intercept.sd", "slope.sd",

"alpha", "beta")

**# MCMC settings**

na <- 1000 ; ni <- 12000 ; nb <- 2000 ; nc <- 3 ; nt <- 10

**# Call JAGS (ART <1 min), check convergence and summarize posteriors**

out18.4 <- jags(dataList, inits, params, "model18.4.txt", n.iter = ni, n.burnin = nb, n.chains = nc, n.thin = nt, n.adapt = na, parallel = TRUE)

par(mfrow=c(2, 2)); jagsUI::traceplot(out18.4)

print(out18.4, 3)

**# Compare likelihood with Bayesian estimates and with truth**

jags\_est <- out18.4$summary[1:4,1]

tmp <- cbind(truth=truth, glmer=glmer\_est, JAGS=jags\_est)

print(tmp, 4)

# 18.6 Bayesian analysis with Stan

# --------------------------------

library(rstan)

**# Bundle and summarize data**

str(dataList <- list(C = C, N = N, pop = as.numeric(pop), prec = prec, nPops = nPops, n = n) )

**# Write Stan model**

cat(file="model18\_6.stan", "

data{

int n; //Number of samples

int nPops; //Number of populations

int N[n]; //Number of trials in each sample

int C[n]; //Successes in each sample

vector[n] prec; //covariate

int pop[n]; //Population index

}

parameters{

real intercept\_mean;

real slope\_mean;

real<lower=0> intercept\_sd;

real<lower=0> slope\_sd;

vector[nPops] alpha;

vector[nPops] beta;

}

model{

vector[n] p; //Estimated success probability

intercept\_mean ~ normal(0, 1000);

slope\_mean ~ normal(0, 1000);

for (i in 1:nPops){

alpha[i] ~ normal(intercept\_mean, intercept\_sd);

beta[i] ~ normal(slope\_mean, slope\_sd);

}

for(i in 1:n){

p[i] = inv\_logit(alpha[pop[i]] + beta[pop[i]] \* prec[i]);

C[i] ~ binomial(N[i], p[i]);

}

}

")

# Parameters to estimate

params <- c("intercept\_mean", "slope\_mean", "intercept\_sd", "slope\_sd",

"alpha", "beta")

**# HMC settings**

ni <- 2000 ; nb <- 1000 ; nc <- 3 ; nt <- 1

**# Call STAN (ART 38/6 sec), assess convergence and print results table**

system.time(

out18.6 <- stan(file = "model18\_6.stan", data = dataList,

warmup = nb, iter = ni, chains = nc, thin = nt) )

rstan::traceplot(out18.6) # not shown

print(out18.6, dig = 3) # not shown

**# Compare estimates with truth**

stan\_est <- summary(out18.6)$summary[1:4,1]

tmp <- cbind(truth=truth, glmer=glmer\_est, JAGS=jags\_est, Stan=stan\_est)

print(tmp, 4)