# 17 Binomial GLM with continuous and categorical explanatory variables

# We can specify a binomial ANCOVA by adding discrete and continuous covariates to the linear predictor of a binomial GLM. Once again, to stress the structural similarity with the normal linear model in chapter 11, we modify the Asp viper example just slightly. Instead of modeling a continuous measurement such as body mass in chapter 11, we will model a count governed by an underlying probability; specifically, we model the proportion of black individuals in Adder populations. The adder has an all-black and a zigzag morph, where females are brown and males grey.

It has been hypothesized that the black color confers a thermal advantage and therefore the proportion of black individuals should be greater in cooler or wetter habitats. We will simulate data that bear on this question and “study”, by simulation, 10 adder populations each in the Jura mountains, the Black Forest and the Alps. We will capture a number of snakes in these populations and record the proportion of black adders. Then, we relate these proportions to the mountain range as well as to a combined index of low temperature, wetness and northerliness of the site. Our expectation will of course be that there are relatively more black adders at cool and wet sites.

# 17.2 Data generation

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

set.seed(17)

nPops <- 3

nSample <- 10

n <- nPops \* nSample

x <- rep(1:nPops, rep(nSample, nPops))

pop <- factor(x, labels = c("Jura", "Black Forest", "Alps"))

# We construct a continuous wetness index: 0 denotes wet sites lacking sun and 1 is the converse.

# For ease of presentation, we sort this covariate; this has no effect on the analysis.

wetness.Jura <- sort(runif(nSample, 0, 1))

wetness.BlackF <- sort(runif(nSample, 0, 1))

wetness.Alps <- sort(runif(nSample, 0, 1))

wetness <- c(wetness.Jura, wetness.BlackF, wetness.Alps)

# We also need the number of adders examined in each population (N), i.e., the binomial totals, also called sample or trial size of the binomial distribution. We assume that the total number of snakes examined in each population is a random variable drawn from a uniform distribution, but this is not essential (and is not part of the model)

N <- round(runif(n, 10, 50) ) # Get discrete Uniform values

# We build the design matrix of an interactive combination of population and wetness.

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

print(Xmat, dig = 2)

# Select the parameter values and save them.

truth <- beta.vec <- c(-4, 1, 2, 6, 2, -5)

# We assemble the number of black adders captured in each population in the usual three steps:

# (1) we add up all components of the linear model to get the value of the linear predictor,

# (2) we apply the inverse logit transformation to get the expected proportion (p) of black adders in each population (Fig. 17-2; top) and finally,

# (3) we add binomial noise, i.e., use p and N to draw binomial random numbers representing the count of black adders in each sample of N snakes (Fig. 17-2; bottom).

# The value of the linear predictor is again obtained by matrix multiplication of the design matrix (Xmat) and the parameter vector (beta.vec).

lin.pred <- Xmat[,] %\*% beta.vec # Value of lin.predictor

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

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

hist(C) # Inspect simulated binomial counts

par(mfrow = c(1,2), mar = c(5,5,3,1))

matplot(cbind(wetness[1:10], wetness[11:20], wetness[21:30]), cbind(exp.p[1:10], exp.p[11:20], exp.p[21:30]), ylab = "Expected proportion black", xlab = "Wetness index", col = c("red","green","blue"), pch = c("J","B","A"), lty = "solid", type = "b", las = 1, cex = 1.2, main = "Expected proportion", lwd = 2, frame = FALSE)

matplot(cbind(wetness[1:10], wetness[11:20], wetness[21:30]), cbind(C[1:10]/N[1:10], C[11:20]/N[11:20], C[21:30]/N[21:30]), ylab = "Observed proportion black", xlab = "Wetness index", col = c("red","green","blue"), pch = c("J","B","A"), las = 1, cex = 1.2, main = "Realized proportion", frame = FALSE)

#17.3 Likelihood analysis with canned functions in R

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

summary(out17.3 <- glm(cbind(C, N-C) ~ pop \* wetness, family = binomial))

**# Compare estimates with truth**

glm\_est <- coef(out17.3)

tmp <- cbind(truth=beta.vec, glm=glm\_est)

print(tmp, 4)

# Owing to the small sample size, we observe only a moderate correspondence with the input values. If we were worried about bias, we could again run a quick simulation by repeating the data simulation/analysis cycle say, 100 times, and then checking that the distribution of the estimates were centered on the true values or not. Alternatively, we could greatly increase the sample size (e.g., setting nSample <- 10000) and then we should also observe much improved agreement between the estimates and the truth. Either would be a great exercise for you.

# 17.4 Bayesian analysis with JAGS

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

**# Bundle and summarize data**

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

wetness=wetness, n=n) )

**# Write JAGS model file**

cat(file="model17.4.txt", "

model {

**# Priors**

for (i in 1:nPops){

alpha[i] ~ dnorm(0, 0.01) # Intercepts

beta[i] ~ dnorm(0, 0.01) # Slopes

}

**# Likelihood**

for (i in 1:n) {

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

logit(p[i]) <- alpha[pop[i]] + beta[pop[i]]\* wetness[i] # Jura is baseline

}

**# Derived quantities**

**# Recover the effects relative to baseline level (no. 1)**

a.effe2 <- alpha[2] - alpha[1] # Intercept Black Forest vs. Jura

a.effe3 <- alpha[3] - alpha[1] # Intercept Alps vs. Jura

b.effe2 <- beta[2] - beta[1] # Slope Black Forest vs. Jura

b.effe3 <- beta[3] - beta[1] # Slope Alps vs. Jura

**# Custom comparison**

test1 <- beta[3] - beta[2] # Difference slope Alps -Black Forest

}

")

**# Function to generate starting values**

inits <- function(){ list(alpha = rnorm(nPops, 3, 1),   
 beta = rnorm(nPops, 2, 1))}

**# Parameters to estimate**

params <- c("alpha", "beta", "a.effe2", "a.effe3", "b.effe2", "b.effe3", "test1")

# Preliminary runs of the model with shorter chains suggest considerable autocorrelation and hence, to accumulate a more "information-dense" sample from the joint posterior, we run longer chains and thin.

**# MCMC settings**

na <- 5000 ; ni <- 30000 ; nb <- 10000 ; nc <- 3 ; nt <- 20

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

out17.4 <- jags(dataList, inits, params, "model17.4.txt", n.iter = ni, n.burnin = nb, n.chains = nc, n.thin = nt, n.adapt = na, parallel = TRUE)

par(mfrow=c(2, 2)); traceplot(out17.4) # not shown

print(out17.4, 3) # not shown

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

jags\_est <- unlist(out17.4$mean)[c(1,7,8,4,9,10)]

tmp <- cbind(truth=beta.vec, glm=glm\_est, JAGS=jags\_est)

print(tmp, 4)

# We get rather similar estimates, but both sets of estimates somewhat off the truth, as we would expect for a small sample size.

#17.6 Bayesian analysis with Stan

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

library(rstan)

**# Bundle and summarize data**

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

wetness=wetness, n=n) )

**# Write Stan model**

cat(file="model17\_6.stan", "

data{

int n; //Number of samples

int nPops; //Number of populations

int C[n]; //Counts for each sample

int N[n]; //Sizes of each sample

int pop[n]; //Population indices

vector[n] wetness; //Covariate

}

parameters{

real alpha[nPops]; //Intercepts for each pop

real beta[nPops]; //Slopes for each pop

}

transformed parameters{

vector[n] p; //Estimated probabilities

for (i in 1:n){

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

}

}

model{

for (i in 1:n){

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

}

}

generated quantities{

real a\_effe2 = alpha[2] - alpha[1]; // Intercept Black Forest vs. Jura

real a\_effe3 = alpha[3] - alpha[1]; // Intercept Alps vs. Jura

real b\_effe2 = beta[2] - beta[1]; // Slope Black Forest vs. Jura

real b\_effe3 = beta[3] - beta[1]; // Slope Alps vs. Jura

real test1 = beta[3] - beta[2]; // Difference slope Alps -Black Forest

}

")

**# HMC settings**

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

**# Call STAN (ART 31/4 sec), assess convergence and print results table**

system.time(

out17.6 <- stan(file = "model17\_6.stan", data = dataList,

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

rstan::traceplot(out17.6) # not shown

print(out17.6, dig = 3) # not shown

**# Compare results with truth and previous estimates**

stan\_est <- summary(out17.6)$summary[c(1,37,38,4,39,40),1]

tmp <- cbind(truth=beta.vec, glm=glm\_est, JAGS=jags\_est, Stan=stan\_est)

print(tmp, 4)

# 17.7 Do-it-yourself MLEs

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

**# Define NLL for general logistic regression with Binomial response**

NLL <- function(beta, y, N, Xmat) {

p <- plogis(Xmat %\*% beta)

L <- dbinom(y, N, p) # Likelihood contribution for 1 observation

LL <- log(L) # Log-likelihood contribution for 1 observation

NLL <- -sum(LL) # NLL for all observations in data set

return(NLL)

}

**# Minimize that NLL to find MLEs and also get SEs**

inits <- rep(0, 6)

names(inits) <- names(coef(out17.3))

out17.7 <- optim(inits, NLL, y = C, N = N, Xmat = Xmat,

hessian=TRUE, method = "BFGS")

getMLE(out17.7, 4)

**# Compare with truth and previous estimates**

diy\_est <- out17.7$par

tmp <- cbind(truth=beta.vec, glm=glm\_est, JAGS=jags\_est,

Stan=stan\_est, DIY=diy\_est)

print(tmp, 4)