# 8 Comparisons among more than two groups in a single classification: Normal one-way ANOVA (fixed effects)

#As a motivating example for this chapter we assume that we measured snout-vent length (SVL) in five populations of Smooth snakes (Fig. 8-1). We are interested in characterizing the populations in terms of SVL of their snakes and in possible differences between populations.

# 8.1Data generation

We assume five populations with 10 snakes measured in each with snout-vent length (SVL) averages of 50, 40, 45, 55 and 60. This corresponds to a baseline population mean of 50 and effects of populations 2–5 of -10, -5, 5, and 10. We choose a residual standard deviation of SVL of 3 and assemble everything.

**# Simulate a data set**

set.seed(82) # Initialize RNGs

nPops <- 5 # Number of populations

nSample <- 10 # Number of snakes in each

pop.means <- c(50, 40, 45, 55, 60) # Population mean SVL

sigma <- 3 # Residual sd

n <- nPops \* nSample # Total number of data points

eps <- rnorm(n, 0, sigma) # Residuals

x <- rep(1:5, rep(nSample, nPops)) # Indicator for population

means <- rep(pop.means, rep(nSample, nPops))

X <- as.matrix(model.matrix(~ as.factor(x)-1)) # Create design matrix

X # Inspect that

y <- as.numeric(X %\*% as.matrix(pop.means) + eps)

# %\*% denotes matrix multiplication

**# Save true values for later comparisons**

truth <- c(pop.means, sigma)

names(truth) <- c(paste0("pop",1:5), "sigma")

**# Make plot (Fig. 8–2)**

par(mfrow = c(1, 1), mar = c(6,6,6,3), cex.lab = 1.5, cex.axis = 1.5, cex.main = 2)

boxplot(y~x, col="grey", xlab="Population", ylab="SVL", main="", las = 1, frame = FALSE)

Fig. 8-2: Snout-vent length (SVL) of Smooth snakes in five populations simulated under a fixed-effects model.

# 8.2 Likelihood analysis using canned functions in R

**# Convert x to a factor to make the output tidier**

pop <- as.factor(x)

print(anova(lm(y~pop))) # produce the ANOVA table (not shown)

summary(out82.2 <- lm(y~pop-1))

**# Compare estimates with truth**

lm\_est <- c(coef(out82.2), sigma=sigma(out82.2))

cbind(truth=truth, lm=lm\_est)

# 8.3 Bayesian analysis with JAGS

# We fit a means parameterization of the model and obtain effects estimates (i.e., differences in the mean SVL among populations) as derived quantities. Note JAGS’ elegant double-indexing (alpha[x[i]]) to specify the expected SVL of snake  according to the -th value of the population index x. We also add two lines to show how custom hypotheses can easily be tested as derived quantities. Test 1 examines whether snakes in populations 2 and 3 have the same size as those in populations 4 and 5. Test 2 checks whether the size difference between snakes in populations 5 and 1 is twice that between populations 4 and 1. Both are fairly arbitrary of course and should serve for illustration only.

**# Bundle and summarize data**

str(dataList <- list(y = y, x = x, n = n, nPops = nPops))

**# Write JAGS model file**

cat(file="model82.3.txt", "

model {

**# Priors**

for (i in 1:nPops){ # Define alpha as a vector

alpha[i] ~ dnorm(0, 0.001)

}

tau <- pow(sigma, -2)

sigma ~ dunif(0, 100)

**# Likelihood**

for (i in 1:n) {

y[i] ~ dnorm(mean[i], tau)

mean[i] <- alpha[x[i]]

}

**# Derived quantities**

effe2 <- alpha[2] - alpha[1]

effe3 <- alpha[3] - alpha[1]

effe4 <- alpha[4] - alpha[1]

effe5 <- alpha[5] - alpha[1]

**# Custom hypothesis test / Define your own contrasts**

test1 <- (effe2+effe3) - (effe4+effe5) # Equals 0 when 2+3 = 4+5

test2 <- effe5 - 2 \* effe4 # Equals 0 when effe5 = 2\*effe4

}

")

**# Function to generate starting values**

inits <- function(){ list(alpha = rnorm(nPops, mean = mean(y)), sigma = rlnorm(1) )}

**# Parameters to estimate**

params <- c("alpha", "sigma", "effe2", "effe3", "effe4", "effe5", "test1", "test2")

**# MCMC settings**

na <- 1000 ; ni <- 3000 ; nb <- 1000 ; nc <- 3 ; nt <- 1

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

out82.3 <- jags(dataList, inits, params, "model82.3.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(out82.3) # not shown

print(out82.3, 3)

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

jags\_est <- out82.3$summary[1:6,1]

cbind(truth=truth, lm=lm\_est, JAGS=jags\_est)

# 8.5 Bayesian analysis with Stan

**# Load Stan R package**

library(rstan)

**# Bundle and summarize data (same)**

str(dataList <- list(y = y, x =x, n = n, nPops = nPops))

**# Write text file with model description in BUGS language**

cat(file = "model82\_5.stan", # This line is R code

"

data { // This is the first line of Stan code

int<lower=1> n; // Declare all data

int<lower=1> nPops;

vector [n] y;

int<lower=1> x[n];

}

parameters { // Define format for all parameters

vector [nPops] alpha;

real<lower=0> sigma;

}

model {

// Priors

alpha ~ normal(0,1000);

sigma ~ cauchy(0, 10);

// Likelihood

for(i in 1:n){

y[i] ~ normal(alpha[x[i]], sigma);

}

}

generated quantities {

// Derived quantities

real effe2 = alpha[2] - alpha[1];

real effe3 = alpha[3] - alpha[1];

real effe4 = alpha[4] - alpha[1];

real effe5 = alpha[5] - alpha[1];

// Custom hypothesis test / Define your own contrasts

real test1 = (effe2+effe3) - (effe4+effe5); // Equals 0 when 2+3 = 4+5

real test2 = effe5 - 2 \* effe4;

} // This is the last line of Stan code

" )

**# HMC settings**

ni <- 3000 ; nb <- 1000 ; nc <- 3 ; nt <- 1

**# Call STAN (ART 31/4 sec)**

system.time(

out82.5 <- stan(file = "model82\_5.stan", data=dataList,

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

rstan::traceplot(out82.5) # not shown

print(out82.5, dig = 2) # not shown

**# Compare estimates with truth**

stan\_est <- summary(out82.5)$summary[1:6,1]

tmp <- cbind(truth=truth, lm=lm\_est, JAGS=jags\_est, NIMBLE=nimble\_est, Stan=stan\_est)

print(tmp, 4)

# 8.6 Do-it-yourself MLEs

# The likelihood to be maximized is the joint likelihood over all units in the data set, where the contribution from each datum comes from a Normal density. Note that argument X in the NLL function we define is the design matrix for a model without intercept.

**# Definition of NLL for a one-way ANOVA with Gaussian errors**

NLL <- function(param, y, X) {

alpha <- param[1:5]

sigma <- param[6]

mu <- X %\*% alpha

L <- dnorm(y, mu, sigma) # Likelihood contr. for 1 datum

LL <- log(L)

NLL <- -sum(LL) # NLL for all data points

return(NLL)

}

**# Get desired design matrix**

X <- model.matrix(~pop-1)

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

inits <- c('mu1' = 50, 'mu2' = 50, 'mu3' = 50, 'mu4' = 50, 'mu5' = 50, 'sigma' = 10)

out82.6 <- optim(inits, NLL, y = y, X = X, hessian=TRUE)

getMLE(out82.6, 4)

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

diy\_est <- out82.6$par

tmp <- cbind(truth=truth, lm=lm\_est, JAGS=jags\_est,  
 Stan=stan\_est, DIY=diy\_est)

print(tmp, 4)