# 6. Comparing two groups in a normal (Gaussian) population

# We first simulate data under this model and for a motivating example return to peregrine falcons. We imagine that we had measured the wingspan of a number of male and female birds and are interested in a sex difference in this measure of size. For Western Europe, Monneret (2006) gives the range of male wingspan as 70–85 cm and that for females as 95–115 cm. Assuming normal distributions for wingspan, this implies means and standard deviations of about 77.5 and 2.5 cm for males, and of 105 and 3 cm for females.

**# 6.1 Generate a data set**

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

set.seed(61)

n1 <- 60 # Number of females

n2 <- 40 # Number of males

mu1 <- 105 # Population mean of females

mu2 <- 77.5 # Population mean of males

sigma <- 2.75 # Average population SD of both

n <- n1+n2 # Total sample size

y1 <- rnorm(n1, mu1, sigma) # Data for females

y2 <- rnorm(n2, mu2, sigma) # Date for males

y <- c(y1, y2) # Aggregate both data sets

x <- rep(c(0,1), c(n1, n2)) # Indicator variable indexing a male

**# Make a plot (Fig. 6-1)**

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 = "Male", ylab = "Wingspan (cm)", las = 1, frame = FALSE)

# Altermative way to simulate the same kind of data. It lets one see perhaps more clearly the principle of an effects parameterization of the linear model:

set.seed(61)

n <- n1+n2 # Total sample size

alpha <- mu1 # Mean for females serves as the intercept

beta <- mu2-mu1 # beta is the difference male-female

E.y <- alpha + beta\*x # Expectation

y.obs <- rnorm(n = n, mean = E.y, sd = sigma) # Add random variation

boxplot(y.obs ~ x, col = "grey", xlab = "Male", ylab = "Wingspan (cm)", las = 1, frame = FALSE)

**# Save true values for later comparisons**

truth <- c(mu1=mu1, delta=mu2-mu1, sigma=sigma)

# 6.2 Likelihood analysis with canned functions in R

summary(out62.1 <- lm(y ~ x)) # Analysis of first data set

summary(out62.2 <- lm(y.obs ~ x)) # Analysis of second data set

anova(out62.1)

**# Compare estimates with truth**

lm\_est <- c(coef(out62.2), sigma=sigma(out62.2))

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

print(tmp, 4)

# 6.3 Bayesian analysis with JAGS

**# Bundle and summarize data**

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

**# Write JAGS model file**

cat(file="model6.3.txt", "

model {

**# Priors**

mu1 ~ dnorm(0,0.001) # Precision = 1/variance

delta ~ dnorm(0,0.001) # Large variance = Small precision

tau <- pow(sigma, -2)

sigma ~ dunif(0, 10)

**# Likelihood**

for (i in 1:n) {

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

mu[i] <- mu1 + delta \*x[i]

}

**# Derived quantities: one of the greatest things about a Bayesian analysis**

for (i in 1:n) {

residual[i] <- y[i] - mu[i] # Define residuals

}

mu2 <- mu1 + delta # Difference in wingspan

}

")

**# Function to generate starting values**

inits <- function(){list(mu1=rnorm(1), delta=rnorm(1), sigma = rlnorm(1))}

**# Parameters to estimate**

params <- c("mu1","mu2", "delta", "sigma", "residual")

**# MCMC settings**

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

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

out6.3 <- jags(data = dataList, inits = inits, parameters.to.save = params, model.file = "model6.3.txt", n.iter = ni, n.burnin = nb, n.chains = nc, n.thin = nt, n.adapt = na, parallel = FALSE)

par(mfrow=c(2, 2)); jagsUI::traceplot(out6.3) # not shown

print(out6.3, 3)

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

jags\_est <- out6.3$summary[c(1,3,4),1]

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

print(tmp, 4)

# 6.5 Bayesian analysis with Stan

**# Load Stan R package**

library(rstan)

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

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

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

cat(file = "model6\_5.stan", # This line is R code

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

int<lower=0> n; // Define the format of all data

vector[n] y; // ... including the dimension of vectors

vector[n] x; //

}

parameters { // Define format for all parameters

real alpha;

real beta;

real<lower=0> sigma;

}

model {

// Priors

alpha ~ normal(0,1000);

beta ~ normal(0,1000);

sigma ~ cauchy(0, 10);

// 'Likelihood'

y ~ normal(alpha + beta \* x, sigma);

} // This is the last line of Stan code

" )

**# HMC settings**

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

**# Call STAN (ART 34/3 sec), check convergence and summarize posteriors**

system.time(

out6.5 <- stan(file = "model6\_5.stan", data=dataList,

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

rstan::traceplot(out6.5) # not shown

print(out6.5, dig = 2) # not shown

**# Compare estimates with truth**

stan\_est <- summary(out6.5)$summary[1:3,1]

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

print(tmp, 4)

6.6 Do-it-yourself MLEs

#The likelihood to be maximized is again the joint likelihood given by the joint probability of all units in the data set, where the contribution from each datum comes from a normal density, but from one which differs between the two samples in terms of the mean, though not in terms of the variance. We could define the likelihood function in term of the logarithm of the variance or standard deviation, as we will often do later, but it turns out that this is not necessary here.

**# Definition of NLL**

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

alpha <- param[1]

beta <- param[2]

sigma <- param[3]

mu <- alpha + beta \* x

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

LL <- log(L) # loglikelihood contr. for 1 obs.

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

return(NLL)

}

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

inits <- c('alpha' = 50, 'beta' = 10, 'sigma' = 2)

out6.6 <- optim(inits, NLL, y = y, x = x, hessian=TRUE)

getMLE(out6.6, 5)

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

diy\_est <- out6.6$par

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

print(tmp, 4)