Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 1

Table of Contents

- Listing 1.1: Summarizing a univariate posterior in R
- Listing 1.2: Summarizing a posterior predictive distribution (PPD) in R

Listing 1.1: Summarizing a univariate posterior in R.

```
# Load the data
  n <- 100
  Y <- 40
  a <- 1
  b <- 1
  A <- Y + a
  B <- n - Y + b

# Define a grid of points for plotting
  theta <- seq(0,1,.001)

# Evaluate the density at these points
  pdf <- dbeta(theta,A,B)

# Plot the posterior density
  plot(theta,pdf,type="l",ylab="Posterior",xlab=expression(theta))

# Posterior mean
  A/(A + B)

# Posterior median (0.5 quantile)
  qbeta(0.5,A,B)

# Posterior probability P(theta<0.5|Y)
  pbeta(0.5,A,B)

# Equal-tailed 95% credible interval
  qbeta(c(0.025,0.975),A,B)

# Monte Carlo approximation
  S        <- 100000
  samples <- rbeta(S,A,B)
  mean(samples)
  quantile(samples,c(0.025,0.975))
```

Listing 1.2: Summarizing a posterior predictive distribution (PPD) in R.

```
# Load the data
 n <- 5
 Y <- 1
 a <- 1
 b <- 1
 A <- Y + a
 B <- n - Y + b

# Plug-in estimator
 theta_hat   <- A/(A+B)
 y           <- 0:5
 PPD         <- dbinom(y,n,theta_hat)
 names(PPD) <- y
 round(PPD,2)

# Draws from the PPD, Y_star[i]~Binomial(n,theta_star[i])
 S            <- 100000
 theta_star   <- rbeta(S,A,B)
 Y_star       <- rbinom(S,n,theta_star)
 PPD          <- table(Y_star)/S
 round(PPD,2)
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 2

Table of Contents

- Listing 2.1: Monte Carlo analysis of the Boston Marathon data

---

Listing 2.1: Monte Carlo analysis of the Boston Marathon data.

```
# Hyperpriors

 nu        <- p+1
 R         <- diag(p)/(p+1)

# Process the data

 Ybar      <- colMeans(Y)
 SSE       <- sweep(Y,2,Ybar,"-")
 SSE       <- solve(t(SSE)%*%SSE)

# Monte Carlo settings from the posterior

 S         <- 10000
 Sigma_mn  <- Cor_mn  <- Omega_pos <- 0

# Monte Carlo sampling

 for(s in 1:S){
   Omega      <- rwish(n+nu,SSE+R)
   Sigma      <- solve(Omega)
   Sigma_mn   <- Sigma_mn + Sigma/S
   Cor_mn     <- Cor_mn + cov2cor(Sigma)/S
   Omega_pos <- Omega_pos + (Omega>0)/S
 }

# Evaluate significance of the precision matrix

 Omega_sig <- ifelse(Omega_pos<0.025  | Omega_pos>0.975, "gray","white")
 Omega_sig <- ifelse(Omega_pos<0.005 | Omega_pos>0.995, "black",Omega_sig)

# Plot some of the results

 library(fields)
 image.plot(1:p,1:p,Sigma_mn,
            xlab="Mile",ylab="Mile",
            main="Posterior mean covariance matrix")
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 3

Table of Contents

- Listing 3.1: Numerical optimization and integration to summarize the posterior
- Listing 3.2: Gibbs sampling for the Gaussian model with unknown mean and variance
- Listing 3.3: Gibbs sampling for the NFL concussions data
- Listing 3.4: Gibbs sampling for linear regression applied to the T-rex data
- Listing 3.5: Metropolis sampling for the NFL concussions data
- Listing 3.6: Metropolis-within-Gibbs sampling for simulated logistic regression data
- Listing 3.7: JAGS code for linear regression applied to the T-rex data
- Listing 3.8: JAGS code for the NFL concussions data
- Listing 3.9: Toy example with poor convergence
- Listing 3.10: Toy example with good convergence

Listing 3.1: Numerical optimization and integration to summarize the posterior.

```
library(cubature)
Y <- c(2.68,  1.18, -0.97, -0.98, -1.03) # Data

# Evaluate the density on the grid for plotting
 m      <- 50
 mu     <- seq(-4,6,length=m)
 sigma  <- seq(0,10,length=m)
 theta  <- as.matrix(expand.grid(mu,sigma))
 D      <- dnorm(theta[,1],0,100)*dunif(theta[,2],0,10) # Prior
 for(i in 1:length(Y)){ # Likelihood
   D <- D * dnorm(Y[i],theta[,1],theta[,2])
 }
 W      <- matrix(D/sum(D),m,m)

# MAP estimation
 neg_log_post <- function(theta,Y){
   log_like   <- sum(dnorm(Y,theta[1],theta[2],log=TRUE))
   log_prior  <- dnorm(theta[1],0,100,log=TRUE)+
                 dunif(theta[2],0,10,log=TRUE)
 return(-log_like-log_prior)}

 inits <- c(mean(Y),sd(Y))
 MAP   <- optim(inits,neg_log_post,Y=Y,
                method = "L-BFGS-B", # Since the prior is bounded
                lower = c(-Inf,0), upper = c(Inf,10))$par

# Compute the posterior mean
 post <- function(theta,Y){
   like  <- prod(dnorm(Y,theta[1],theta[2]))
   prior <- dnorm(theta[1],0,100)*dunif(theta[2],0,10)
 return(like*prior)}

 g0 <- function(theta,Y){post(theta,Y)}
 g1 <- function(theta,Y){theta[1]*post(theta,Y)}
 g2 <- function(theta,Y){theta[2]*post(theta,Y)}
 m0 <- adaptIntegrate(g0,c(-5,0.01),c(5,5),Y=Y)$int #constant m(Y)
 m1 <- adaptIntegrate(g1,c(-5,0.01),c(5,5),Y=Y)$int
 m2 <- adaptIntegrate(g2,c(-5,0.01),c(5,5),Y=Y)$int
 pm <- c(m1,m2)/m0

# Make the plot
 image(mu,sigma,W,col=gray.colors(10,1,0),
       xlab=expression(mu),ylab=expression(sigma))
 points(theta,cex=0.1,pch=19)
 points(pm[1],pm[2],pch=19,cex=1.5)
 points(MAP[1],MAP[2],col="white",cex=1.5,pch=19)
 box()
```

Listing 3.2: Gibbs sampling for the Gaussian model with unknown mean and variance.

```
 # Load the data

 Y <- c(2.68,1.18,-0.97,-0.98,-1.03)
 n <- length(Y)

# Create an empty matrix for the MCMC samples

 S                  <- 25000
 samples            <- matrix(NA,S,2)
 colnames(samples)  <- c("mu","sigma")

# Initial values

 mu    <- mean(Y)
 sig2  <- var(Y)

# priors: mu ~ N(gamma,tau), sig2 ~ InvG(a,b)

 gamma  <- 0
 tau    <- 100^2
 a      <- 0.1
 b      <- 0.1

# Gibbs sampling

 for(s in 1:S){
   P     <- n/sig2 + 1/tau
   M     <- sum(Y)/sig2 + gamma/tau
   mu    <- rnorm(1,M/P,1/sqrt(P))

   A     <- n/2 + a
   B     <- sum((Y-mu)^2)/2 + b
   sig2  <- 1/rgamma(1,A,B)

   samples[s,]<-c(mu,sqrt(sig2))
 }

 # Plot the joint posterior and marginal of mu
 plot(samples,xlab=expression(mu),ylab=expression(sigma))
 hist(samples[,1],xlab=expression(mu))

 # Posterior mean, median and credible intervals
 apply(samples,2,mean)
 apply(samples,2,quantile,c(0.025,0.500,0.975))
```

Listing 3.3: Gibbs sampling for the NFL concussions data.

```
# Load data

 Y <- c(171, 152, 123, 199)
 n <- 4
 N <- 256

# Create an empty matrix for the MCMC samples

 S                 <- 25000
 samples           <- matrix(NA,S,5)
 colnames(samples) <- c("lam1","lam2","lam2","lam4","gamma")

# Initial values

 lambda <- log(Y/N)
 gamma  <- 1/mean(lambda)

# priors: lambda[i]|gamma ~ Gamma(1,gamma), gamma ~ InvG(a,b)

 a <- 0.1
 b <- 0.1

# Gibbs sampling

 for(s in 1:S){
   for(i in 1:n){
     lambda[i] <- rgamma(1,Y[i]+1,N+gamma)
   }
   gamma        <- rgamma(1,4+a,sum(lambda)+b)
   samples[s,]  <- c(lambda,gamma)
 }

 boxplot(samples[,1:4],outline=FALSE,
         ylab=expression(lambda),names=2012:2015)
 plot(samples[,5],type="l",xlab="Iteration",
      ylab=expression(gamma))

 # Posterior mean, median and credible interval
 apply(samples,2,mean)
 apply(samples,2,quantile,c(0.025,0.500,0.975))
```

Listing 3.4: Gibbs sampling for linear regression applied to the T-rex data.

```
library(mvtnorm)

# Load T-Rex data

 mass <- c(29.9, 1761, 1807, 2984, 3230, 5040, 5654)
 age  <- c(2, 15, 14, 16, 18, 22, 28)
 n    <- length(age)
 X    <- cbind(1,age)
 Y    <- mass

# Create an empty matrix for the MCMC samples

 S                 <- 10000
 samples           <- matrix(NA,S,3)
 colnames(samples) <- c("Beta1","Beta2","Sigma")

# Initial values

 beta <- lm(mass~age)$coef
 sig2 <- var(lm(mass~age)$residuals)

# priors: beta ~ N(0,tau I_2), sigma^2 ~ InvG(a,b)

 tau    <- 10000^2
 a      <- 0.1
 b      <- 0.1

# Blocked Gibbs sampling

 V      <- diag(2)/tau
 tXX    <- t(X)%*%X
 tXY    <- t(X)%*%Y

 for(s in 1:S){
   P    <- tXX/sig2 + V
   W    <- tXY/sig2
   beta <- rmvnorm(1,solve(P)%*%W,solve(P))
   beta <- as.vector(beta)

   A    <- n/2 + a
   B    <- sum((Y-X%*%beta)^2)/2 + b
   sig2 <- 1/rgamma(1,A,B)

   samples[s,]<-c(beta,sqrt(sig2))
 }

 pairs(samples)
```

Listing 3.5: Metropolis sampling for the NFL concussions data.

```
# Load data

 Y <- c(171, 152, 123, 199)
 t <- 1:4
 n <- 4
 N <- 256

# Create an empty matrix for the MCMC samples

 S                   <- 25000
 samples             <- matrix(NA,S,2)
 colnames(samples)   <- c("beta1","beta2")
 fitted              <- matrix(NA,S,4)

# Initial values
 beta   <- c(log(mean(Y/N)),0)

# priors: beta[j] ~ N(0,tau^2)
 tau    <- 10

# Prep for Metropolis sampling

 log_post <- function(Y,N,t,beta,tau){
    mn     <- N*exp(beta[1]+beta[2]*t)
    like   <- sum(dpois(Y,mn,log=TRUE))
    prior  <- sum(dnorm(beta,0,tau,log=TRUE))
    post   <- like + prior
 return(post)}

 can_sd <- rep(0.1,2)

# Metropolis sampling

 for(s in 1:S){
   for(j in 1:2){
     can     <- beta
     can[j]  <- rnorm(1,beta[j],can_sd[j])
     logR    <- log_post(Y,N,t,can,tau)-log_post(Y,N,t,beta,tau)
     if(log(runif(1))<logR){
       beta <- can
     }
   }
   samples[s,] <- beta
   fitted[s,]  <- N*exp(beta[1]+beta[2]*t)
 }

 boxplot(fitted,outline=FALSE,ylim=range(Y),
         xlab="Year",ylab="Fitted values",names=2012:2015)
 points(Y,pch=19)
```

Listing 3.6: Metropolis-within-Gibbs sampling for simulated logistic regression data.

```
# Simulate data
 n          <- 100
 p          <- 20
 X          <- cbind(1,matrix(rnorm(n*(p-1)),n,p-1))
 beta_true  <- rnorm(p,0,.5)
 prob       <- 1/(1+exp(-X%*%beta_true))
 Y          <- rbinom(n,1,prob)

# Function to compute the log posterior
 log_post_beta <- function(Y,X,beta,sigma){
     prob  <- 1/(1+exp(-X%*%beta))
     like  <- sum(dbinom(Y,1,prob,log=TRUE))
     prior <- dnorm(beta[1],0,10,log=TRUE) +             # Intercept
              sum(dnorm(beta[-1],0,sigma,log=TRUE)) # Slopes
 return(like+prior)}

# Create empty matrix for the MCMC samples
 S                 <- 10000
 samples           <- matrix(NA,S,p+1)

# Initial values and priors
 beta   <- rep(0,p)
 sigma  <- 1
 a      <- 0.1
 b      <- 0.1
 can_sd <- 0.1

 for(s in 1:S){

   # Metropolis for beta
   for(j in 1:p){
     can    <- beta
     can[j] <- rnorm(1,beta[j],can_sd)
     logR   <- log_post_beta(Y,X,can,sigma)-
               log_post_beta(Y,X,beta,sigma)
     if(log(runif(1))<logR){
       beta <- can
     }
   }

   # Gibbs for sigma
   sigma <- 1/sqrt(rgamma(1,(p-1)/2+a,sum(beta[-1]^2)/2+b))

   samples[s,] <- c(beta,sigma)

  }

  boxplot(samples[,1:p],outline=FALSE,
          xlab="Index, j",ylab=expression(beta[j]))
  points(beta_true,pch=19,cex=1.25)
```

Listing 3.7: JAGS code for linear regression applied to the T-rex data.

```
library(rjags)
# Load T-Rex data
mass <- c(29.9, 1761, 1807, 2984, 3230, 5040, 5654)
age  <- c(2, 15, 14, 16, 18, 22, 28)
n    <- length(age)
data <- list(mass=mass,age=age,n=n)

# (1) Define the model as a string
model_string <- textConnection("model{
  # Likelihood (dnorm uses a precision, not variance)
   for(i in 1:n){
     mass[i] ~ dnorm(beta1 + beta2*age[i],tau)
   }
  # Priors
   tau   ~  dgamma(0.1, 0.1)
   sigma <- 1/sqrt(tau)
   beta1 ~  dnorm(0, 0.001)
   beta2 ~  dnorm(0, 0.001)
}")

# (2) Load the data, specify initial values and compile the MCMC code
inits <- list(beta1=rnorm(1),beta2=rnorm(1),tau=rgamma(1,1))
model <- jags.model(model_string,data = data, inits=inits, n.chains=2)

# (3) Burn-in for 10000 samples
update(model, 10000, progress.bar="none")

# (4) Generate 20000 post-burn-in samples
params  <- c("beta1","beta2","sigma")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=20000, progress.bar="none")

# (5) Summarize the output
summary(samples)
plot(samples)
```

Listing 3.8: JAGS code for the NFL concussions data.

```
library(rjags)
# Load the NFL concussions data
 Y <- c(171, 152, 123, 199)
 n <- 4
 N <- 256

# (1) Define the model as a string
 model_string <- textConnection("model{
   # Likelihood
    for(i in 1:n){
      Y[i] ~ dpois(N*lambda[i])
    }
   # Priors
    for(i in 1:n){
      lambda[i] ~ dgamma(1,gamma)
    }
    gamma   ~  dgamma(a, b)
 }")

# (2) Load the data and compile the MCMC code
inits <- list(lambda=rgamma(n,1,1),gamma=1)
data  <- list(Y=Y,N=N,n=n,a=0.1,b=0.1)
model <- jags.model(model_string,data = data, inits=inits, n.chains=2)

# (3) Burn-in for 10000 samples
update(model, 10000, progress.bar="none")

# (4) Generate 20000 post-burn-in samples
params  <- c("lambda")
samples <- coda.samples(model,
           variable.names=params,
           n.iter=20000, progress.bar="none")

# (5) Compute 90% credible intervals
samps <- rbind(samples[[1]],samples[[2]]) #2S x 4 matrix of samples
apply(samps,2,quantile,c(0.05,0.95))
```

Listing 3.9: Toy example with poor convergence.

```
# Define the model as a string
model_string <- textConnection("model{
  Y     ~ dpois(exp(mu[1]+mu[2]))
  mu[1] ~ dnorm(0,0.001)
  mu[2] ~ dnorm(0,0.001)
}")

# Generate MCMC samples
inits <- list(mu=rnorm(2,0,5))
data  <- list(Y=1)
model <- jags.model(model_string,data = data,
                    inits=inits, n.chains=3)

update(model, 1000, progress.bar="none")
samples <- coda.samples(model,
           variable.names=c("mu"),
           n.iter=5000, progress.bar="none")

# Apply convergence diagnostics

# Plots
plot(samples)
autocorr.plot(samples)

# Statistics
autocorr(samples[[1]],lag=1)
effectiveSize(samples)
gelman.diag(samples)
geweke.diag(samples[[1]])
```

Listing 3.10: Toy example with good convergence.

```
# Define the model as a string
model_string <- textConnection("model{
  Y1    ~ dpois(exp(mu[1]))
  Y2    ~ dpois(exp(mu[2]))
  mu[1] ~ dnorm(0,0.001)
  mu[2] ~ dnorm(0,0.001)
}")

# Generate MCMC samples
inits <- list(mu=rnorm(2,0,5))
data  <- list(Y1=1,Y2=10)
model <- jags.model(model_string,data = data,
                    inits=inits, n.chains=3)

update(model, 1000, progress.bar="none")
samples <- coda.samples(model,
           variable.names=c("mu"),
           n.iter=5000, progress.bar="none")

# Apply convergence diagnostics

# Plots
plot(samples)
autocorr.plot(samples)

# Statistics
autocorr(samples[[1]],lag=1)

effectiveSize(samples)
gelman.diag(samples)
geweke.diag(samples[[1]])
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 4

Table of Contents

- Listing 4.1: R code for comparing two normal means with the Jeffreys' prior
- Listing 4.2: R code for Bayesian linear regression under the Jeffreys' prior
- Listing 4.3: JAGS code for multiple linear regression with Gaussian priors
- Listing 4.4: JAGS code for the Bayesian LASSO
- Listing 4.5: JAGS code for linear regression predictions
- Listing 4.6: R code to use JAGS MCMC samples for linear regression predictions
- Listing 4.7: Model statements for several GLMs in JAGS
- Listing 4.8: The one-way random effects model in JAGS
- Listing 4:9: The one-way random effects model with half-Cauchy priors
- Listing 4.10: Random slopes model in JAGS
- Listing 4.11: Model statement for heteroskedastic Gaussian regression
- Listing 4.12: Model statement for Gaussian regression with non-normal errors
- Listing 4.13: Random slopes model with autoregressive dependence in JAGS

Listing 4.1: R code for comparing two normal means with the Jeffreys' prior.

```
# Y1 is the n1-vector of data for group 1
# Y2 is the n2-vector of data for group 2

# Statistics from group 1
 Ybar1  <- mean(Y1)
 s21    <- mean((Y1-Ybar1)^2)
 n1     <- length(Y1)

# Statistics from group 2
 Ybar2  <- mean(Y2)
 s22    <- mean((Y2-Ybar2)^2)
 n2     <- length(Y2)

# Posterior of the difference assuming equal variance
 delta_hat  <- Ybar2-Ybar1
 s2         <- (n1*s21 + n2*s22)/(n1+n2)
 scale      <- sqrt(s2)*sqrt(1/n1+1/n2)
 df         <- n1+n2
 cred_int   <- delta_hat + scale*qt(c(0.025,0.975),df=df)

# Posterior of delta assuming unequal variance using MC sampling
 mu1        <- Ybar1 + sqrt(s21/n1)*rt(1000000,df=n1)
 mu2        <- Ybar2 + sqrt(s22/n2)*rt(1000000,df=n2)
 delta      <- mu2-mu1

 hist(delta,main="Posterior distribution of the difference in means")
 quantile(delta,c(0.025,0.975)) # 95% credible set
```

Listing 4.2: R code for Bayesian linear regression under the Jeffreys' prior.

```
# This code assumes:
#   Y is the n-vector of observations
#   X us the n x p matrix of covariates
#   The first column of X is all ones for the intercept

# Compute posterior mean and 95% interval
 beta_mean   <- solve(t(X)%*%X)%*%t(X)%*%Y
 sigma2      <- mean((Y-X%*%beta_mean)^2)
 beta_cov    <- sigma2*solve(t(X)%*%X)
 beta_scale  <- sqrt(diag(beta_cov))
 df          <- length(Y)
 beta_025    <- beta_mean + beta_scale*qt(0.025,df=df)
 beta_975    <- beta_mean + beta_scale*qt(0.975,df=df)

# Package the output
 out             <- cbind(beta_mean,beta_025,beta_975)
 rownames(out) <- colnames(X)
 colnames(out) <- c("Mean","Q 0.025","Q 0.975")
```

Listing 4.3: JAGS code for multiple linear regression with Gaussian priors.

```
# Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]),taue)
 }
# Priors
 beta[1] ~ dnorm(0,0.001) #X[i,1]=1 for the intercept
 for(j in 2:p){
    beta[j] ~ dnorm(0,taub*taue)
 }
 taue   ~  dgamma(0.1, 0.1)
 taub   ~  dgamma(0.1, 0.1)
```

Listing 4.4: JAGS code for the Bayesian LASSO.

```
# Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]),taue)
 }
# Priors
 beta[1] ~ dnorm(0,0.001)
 for(j in 2:p){
    beta[j] ~ ddexp(0,taub*taue)
 }
 taue   ~  dgamma(0.1, 0.1)
 taub   ~  dgamma(0.1, 0.1)
```

Listing 4.5: JAGS code for linear regression predictions.

```
# Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]),taue)
 }
# Priors
 beta[1] ~ dnorm(0,0.001)
 for(j in 2:p){
    beta[j] ~ dnorm(0,taub*taue)
 }
 taue   ~  dgamma(0.1, 0.1)
 taub   ~  dgamma(0.1, 0.1)

# Predictions
 for(i in 1:n_pred){
    Y_pred[i] ~ dnorm(inprod(X_pred[i,],beta[]),taue)
 }
# User must pass JAGS the covariates X_pred and integer n_pred
# JAGS returns PPD samples of Y_pred
```

Listing 4.6: R code to use JAGS MCMC samples for linear regression predictions.

```
# INPUTS
#   beta_samples := S x p matrix of MCMC samples (from JAGS)
#   taue_samples := S x 1 matrix of MCMC samples (from JAGS)
#   X_pred       := n_pred x p matrix of prediction covariates

S       <- nrow(beta_samples)
n_pred <- nrow(X_pred)
Y_pred <- matrix(NA,S,n_pred)
sigma   <- 1/sqrt(taue_samples)

for(s in 1:S){
  Y_pred[s,] ~ X_pred%*%beta_samples[s,]+rnorm(n_pred,0,sigma[s])
}

# OUTPUT
#   Y_pred        := S x n_pred matrix of PPD samples
```

Listing 4.7: Model statements for several GLMs in JAGS.

```
# (a) Logistic regression
  for(i in 1:n){
    Y[i]           ~ dbern(q[i])
    logit(q[i]) <- inprod(X[i,],beta[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}

# (b) Probit regression
  for(i in 1:n){
    Y[i]            ~ dbern(q[i])
    probit(q[i]) <- inprod(X[i,],beta[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}

# (c) Poisson regression
  for(i in 1:n){
    Y[i]              ~ dpois(lambda[i])
    log(lambda[i]) <- inprod(X[i,],beta[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}

# (d) Negative binomial regression
  for(i in 1:n){
    Y[i]              ~ dnegbin(q[i],m)
    q[i]             <- m/(m + lambda[i])
    log(lambda[i]) <- inprod(X[i,],beta[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
  m ~ dgamma(0.1,0.1)

# (e) Zero-inflated Poisson
  for(i in 1:n){
    Y[i]              ~ dpois(q[i])
    q[i]             <- Z[i]*lambda[i]
    Z[i]              ~ dbern(p[i])
    log(lambda[i])  <- inprod(X[i,],beta[])
    logit(p[i])      <- inprod(X[i,],alpha[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
  for(j in 1:p){alpha[j] ~ dnorm(0,0.01)}

# (f) Beta regression
  for(i in 1:n){
    Y[i]           ~ dbeta(r*q[i],r*(1-q[i]))
    logit(q[i]) <- inprod(X[i,],beta[])
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
  r ~ dgamma(0.1,0.1)
```

Listing 4.8: The one-way random effects model in JAGS.

```
# Likelihood
 for(i in 1:n){for(j in 1:m){
   Y[i,j] ~ dnorm(alpha[i],sig2_inv)
 }}

# Random effects
 for(i in 1:n){alpha[i] ~ dnorm(mu,tau2_inv)}

# Priors
 mu        ~ dnorm(0,0.0001)
 sig2_inv ~ dgamma(0.1,0.1)
 tau2_inv ~ dgamma(0.1,0.1)
```

Listing 4:9: The one-way random effects model with half-Cauchy priors.

```
# Likelihood
 for(i in 1:n){for(j in 1:m){
   Y[i,j] ~ dnorm(alpha[i],sig2_inv)
 }}

# Random effects
 for(i in 1:n){alpha[i] ~ dnorm(mu,tau2_inv)}

# Priors
 mu         ~ dnorm(0,0.0001)
 sig2_inv  <- pow(sigma1,-2)
 tau2_inv  <- pow(sigma2,-2)
 sigma1     ~ dt(0, 1, 1)T(0,) # Half-Cauchy priors with
 sigma2     ~ dt(0, 1, 1)T(0,) # location 0 and scale 1
```

Listing 4.10: Random slopes model in JAGS.

```
# Likelihood
  for(i in 1:n){for(j in 1:m){
    Y[i,j] ~ dnorm(alpha[i,1]+alpha[i,2]*age[j],tau)
  }}

# Random effects
  for(i in 1:n){
     alpha[i,1:2] ~ dmnorm(beta[1:2],Omega_inv[1:2,1:2])
  }

# Priors
  tau ~ dgamma(0.1,0.1)
  for(j in 1:2){beta[j] ~ dnorm(0,0.0001)}
  Omega_inv[1:2,1:2] ~ dwish(R[,],2.1)

  R[1,1]<-1/2.1
  R[1,2]<-0
  R[2,1]<-0
  R[2,2]<-1/2.1
```

Listing 4.11: Model statement for heteroskedastic Gaussian regression.

```
for(i in 1:n){
  Y[i]      ~ dnorm(mu[i],prec[i])
  mu[i]    <- inprod(x[i,],beta[])
  prec[i] <- 1/sig2[i]
  sig2[i]   <- exp(inprod(x[i,],alpha[]))
}
for(j in 1:p){beta[j] ~ dnorm(0,taub)}
for(j in 1:p){alpha[j] ~ dnorm(0,taua)}
taub ~ dgamma(0.1,0.1)
taua ~ dgamma(0.1,0.1)
```

Listing 4.12: Model statement for Gaussian regression with non-normal errors.

```
# (a) Regression with student-t errors
for(i in 1:n){
  Y[i]   ~ dt(mu[i],tau,df)
  mu[i] <- inprod(X[i,],beta[])
}
for(j in 1:p){beta[j] ~ dnorm(0,taub)}
tau ~ dgamma(0.1,0.1)
df  ~ dgamma(0.1,0.1)


# (b) Regression with mixture-of-normals errors
for(i in 1:n){
  Y[i]   ~ dnorm(mu[i]+theta[g[i]],tau1)
  mu[i] <- inprod(X[i,],beta[])
  g[i]   ~ dcat(pi[])
}
for(k in 1:K){theta[k] ~ dnorm(0,tau2)}
for(j in 1:p){beta[j]  ~ dnorm(0,tau3)}
tau1     ~ dgamma(0.1,0.1)
tau2     ~ dgamma(0.1,0.1)
tau3     ~ dgamma(0.1,0.1)
pi[1:K]  ~ ddirch(alpha[1:K])
```

Listing 4.13: Random slopes model with autoregressive dependence in JAGS.

```
# Likelihood
  for(i in 1:n){
    Y[i,1:m] ~ dmnorm(mn[i,1:m],SigmaInv)
    for(j in 1:m){mn[i,j] <- alpha[i,1]+alpha[i,2]*age[j]}
  }
  SigmaInv[1:m,1:m] <- inverse(Sigma[1:m,1:m])
  for(j in 1:m){for(k in 1:m){
     Sigma[j,k] <- pow(rho,abs(k-j))/tau
  }}

# Random effects
  for(i in 1:n){alpha[i,1:2] ~ dmnorm(beta[1:2],Omega[1:2,1:2])}

# Priors
  tau ~ dgamma(0.1,0.1)
  for(j in 1:2){beta[j] ~ dnorm(0,0.0001)}
  rho  ~ dunif(0,1)
  Omega[1:2,1:2] ~ dwish(R[,],2.1)

  R[1,1] <- 1/2.1
  R[1,2] <- 0
  R[2,1] <- 0
  R[2,2] <- 1/2.1
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 5

Table of Contents

- Listing 5.1: JAGS code for SSVS
- Listing 5.2: JAGS code to compute WAIC and DIC for the random effects model
- Listing 5.3: JAGS code to compute DIC for the constant slopes model
- Listing 5.4: JAGS code for the over-dispersed Poisson regression model

---

Listing 5.1: JAGS code for SSVS.

```
for(i in 1:n){
  Y[i]          ~ dbern(pi[i])
  logit(pi[i]) <- alpha              + X[i,1]*beta[1] +
                    X[i,2]*beta[2] + X[i,3]*beta[3] +
                    X[i,4]*beta[4] + X[i,5]*beta[5]
}
for(j in 1:5){
  beta[j] <- gamma[j]*delta[j]
  gamma[j] ~ dbern(0.5)
  delta[j] ~ dnorm(0,tau)
}
alpha ~ dnorm(0,0.01)
tau   ~ dgamma(0.1,0.1)
```

Listing 5.2: JAGS code to compute WAIC and DIC for the random effects model.

```
mod <- textConnection("model{
  for(i in 1:n){
    Y[i]          ~ dbern(pi[i])
    logit(pi[i]) <- beta[1] + X[i,1]*beta[2]
                    + X[i,2]*beta[3] + X[i,3]*beta[4]
                    + X[i,4]*beta[5]  + X[i,5]*beta[6]
                    + theta[village[i]]
    like[i]      <- dbin(Y[i],pi[i],1) # For WAIC computation
  }
  for(j in 1:6){beta[j] ~ dnorm(0,0.01)}
  for(j in 1:65){theta[j] ~ ddexp(0,tau)}
  tau   ~ dgamma(0.1,0.1)
}")

data   <- list(Y=Y,X=X,n=n,village=village)
model  <- jags.model(mod,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
samps  <- coda.samples(model, variable.names=c("like"),
                       n.iter=50000, progress.bar="none")

# Compute DIC
DIC    <- dic.samples(model,n.iter=50000,progress.bar="none")

# Compute WAIC
like   <- rbind(samps[[1]],samps[[2]]) # Combine the two chains
fbar   <- colMeans(like)
Pw     <- sum(apply(log(like),2,var))
WAIC   <- -2*sum(log(fbar))+2*Pw
```

Listing 5.3: JAGS code to compute DIC for the constant slopes model.

```
model_string <- "model{
  for(i in 1:n){
     Y[i]     ~ dnorm(Xb[i],taue)
     Xb[i]    ~ inprod(X[i,],beta[])
     like[i] <- dnorm(Y[i],Xb[i],taue) # For WAIC
  }
  for(j in 1:p){beta[j] ~ dnorm(0,0.01)}
  tau ~ dgamma(0.1,0.1)
}"

# Put the data and model statement JAGS format
dat   <- list(Y=Y,n=n,X=X,p=p)
init  <- list(beta=rep(0,p),tau=1)
model <- jags.model(textConnection(model_string),n.chains=2,
                    inits=init,data = dat)

# Burn-in samples
update(model, 10000, progress.bar="none")

# Compute DIC
dic <- dic.samples(model1,n.iter=50000)

# Compute WAIC
samp <- coda.samples(model, variable.names=c("like"),
        n.iter=50000)
like <- rbind(samps[[1]],samps[[2]]) # Combine the two chains
fbar <- colMeans(like)
Pw   <- sum(apply(log(like),2,var))
WAIC <- -2*sum(log(fbar)) + 2*Pw
```

Listing 5.4: JAGS code for the over-dispersed Poisson regression model.

```
# Likelihood
for(i in 1:n){
  Y[i]             ~ dnegbin(q[i],m)
  q[i]            <- m/(m+N[i]*lambda[i])
  log(lambda[i]) <- alpha + inprod(Z[i,],beta[1:5]) + X[i]*beta[6]
}

#Priors
 for(j in 1:6){
    beta[j] ~ dnorm(0,0.1)
 }
 alpha ~ dnorm(0,0.1)
 m     ~ dgamma(0.1,0.1)

# Posterior predictive checks
for(i in 1:n){
  Y2[i]    ~ dnegbin(q[i],m)
  rate[i] <- Y2[i]/N[i]
}

D[1] <- min(Yp[])
D[2] <- max(Yp[])
D[3] <- max(Yp[])-min(Yp[])
D[4] <- min(rate[])
D[5] <- max(rate[])
D[6] <- max(rate[])-min(rate[])
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 6

Table of Contents

- Listing 6.1: Spatial data fusion model for BHNU abundance
- Listing 6.2: JAGS code for hierarchical growth curve modeling
- Listing 6.3: JAGS model statement for the missing data analysis of the marathon data

---

Listing 6.1: Spatial data fusion model for BHNU abundance.

```
# Data layer
   for(i in 1:n){
     Y1[i]   ~ dbin(phi[i],N1[i]) # BBS
     phi[i] <- 1-exp(-lam[i])

     Y2[i]   ~ dnegbin(q[i],m)      # eBird
     q[i]   <- m/(m+N2[i]*(theta1*lam[i]+theta2))
   }

# Process layer
   for(j in 1:p){beta[j]~dnorm(beta0,tau)}
   for(i in 1:n){
     log(lam[i])   <- inprod(X[i,],beta[])
   }

# Prior layer
   theta1  ~ dgamma(0.1,0.1)
   theta2  ~ dgamma(0.1,0.1)
   m       ~ dgamma(0.1,0.1)
   tau     ~ dgamma(0.1,0.1)
   beta0   ~ dnorm(0,1)
```

Listing 6.2: JAGS code for hierarchical growth curve modeling.

```
# n is the total number of observations for all species
# x[i] is the log age of individual i
# y[i] is the log mass of individual i
# sp[i] is the species number (1, 2, 3, or 4) of individual i

# Data layer
for(i in 1:n){
  y[i]      ~ dnorm(muY[i],taue)
  muY[i]   <- log(a[sp[i]] + b[sp[i]]/(1+exp(-part[i])) -   0.5/taue
  part[i] <- (x[i]-c[sp[i]])/d[sp[i]]
}

# Process layer
for(j in 1:N){
  a[j]   <- exp(alpha[j,1])
  b[j]   <- exp(alpha[j,2])
  c[j]   <- alpha[j,3]
  d[j]   <- exp(alpha[j,4])

  for(k in 1:4){alpha[j,k] ~ dnorm(mu[k],tau[k])}
}

# Prior layer
for(k in 1:4){
  mu[k]   ~ dnorm(0,0.1)
  tau[k] ~ dgamma(0.1,0.1)
}
taue ~ dgamma(0.1,0.1)
```

Listing 6.3: JAGS model statement for the missing data analysis of the marathon data.

```
# Likelihood
   for(i in 1:n){
     Y[i] ~ dnorm(alpha + inprod(X[i,],beta[]),taue)
   }

  # Missing-data model
   for(i in 1:n){
     X[i,1]  ~ dnorm(0,tau1)
     for(j in 2:p){
       X[i,j] ~ dnorm(rho*X[i,j-1],tau2)
     }
   }

  # Priors
   alpha ~ dnorm(0,0.01)
   for(j in 1:p){
     beta[j] ~ dnorm(0,0.01)
   }
   taue  ~  dgamma(0.1, 0.1)
   tau1  ~  dgamma(0.1, 0.1)
   tau2  ~  dgamma(0.1, 0.1)
   rho   ~  dnorm(0, 0.01)
```

Code for the listings in Bayesian Statistical Methods Code by Reich and Ghosh

Chapter 7

Table of Contents

- Listing 7.1: R simulation study code

Listing 7.1: R simulation study code.

```
# Set up the simulation
 library(BLR)
 n       <- 25       # Sample size
 p_null <- 15        # Number of null covariates
 p_act   <- 5        # Number of active covariates
 nsims   <- 100      # Number of simulated datasets
 sigma   <- 1        # True value of sigma
 beta    <- c(rep(0,p_null),rep(1,p_act))  # True beta

# Define matrices to store the results
 p    <- p_null + p_act
 EST1 <- VAR1 <- matrix(0,nsims,p)
 EST2 <- VAR2 <- matrix(0,nsims,p)

# Start the simulation
 for(sim in 1:nsims){
  set.seed(sim*1234)
  # Generate a dataset
    X <- matrix(rnorm(n*p),n,p)
    Y <- X%*%beta+rnorm(n,0,sigma)

  # Fit ordinary least squares
    ols         <- summary(lm(Y~X))$coef[-1,]
    EST1[sim,] <- ols[,1]
    VAR1[sim,] <- ols[,2]^2

  # Fit the Bayesian LASSO
    blr         <- BLR(y=Y,XL=X)
    EST2[sim,] <- blr$bL
    VAR2[sim,] <- blr$SD.bL^2
  }

# Compute the results
 E    <- sweep(EST1,2,beta,"-")
 MSE  <- mean(E^2)
 BIAS <- mean(E)
 VAR  <- mean(VAR1)
 COV  <- mean(abs(E/sqrt(VAR1))<2)

 E    <- sweep(EST2,2,beta,"-")
 MSE  <- c(MSE,mean(E^2))
 BIAS <- c(BIAS,mean(E))
 VAR  <- c(VAR,mean(VAR2))
 COV  <- c(COV,mean(abs(E/sqrt(VAR2))<2))

 out  <- cbind(BIAS,VAR,MSE,COV)
```