LaplacesDemon Examples

Byron Hall
STATISTICAT, LLC

Abstract

The **LaplacesDemon** package in R enables Bayesian inference with any Bayesian model, provided the user specifies the likelihood. This vignette is a compendium of examples of how to specify different model forms.

Keywords:~Bayesian, Bayesian Inference, Laplace's Demon, LaplacesDemon, R, STATISTICAT.

LaplacesDemon (Hall 2011), usually referred to as Laplace's Demon, is an R package that is available on CRAN (R Development Core Team 2010). A formal introduction to Laplace's Demon is provided in an accompanying vignette entitled "**LaplacesDemon** Tutorial", and an introduction to Bayesian inference is provided in the "Bayesian Inference" vignette.

The purpose of this document is to provide users of the **LaplacesDemon** package with examples of a variety of Bayesian methods. To conserve space, the examples are not worked out in detail, and only the minimum of necessary materials is provided for using the various methodologies. Necessary materials include the form expressed in notation, data (which is often simulated), initial values, and the `Model` function. This vignette will grow over time as examples of more methods become included. Contributed examples are welcome. Please send contributed examples in a similar format in an email to `statisticat@gmail.com` for review and testing. All accepted contributions are, of course, credited.

Contents

1. Autoregression, AR(1)

1.1. Form

$$y_t \sim N(\mu_{t-1}, \tau^{-1}), \quad t = 2, \ldots, (T-1)$$
$$y^{new}_T \sim N(\mu_T, \tau^{-1})$$
$$\mu_t = \alpha + \phi y_t, \quad t = 1, \ldots, T$$
$$\alpha \sim N(0, 1000)$$
$$\phi \sim N(0, 1000)$$
$$\tau \sim \Gamma(0.001, 0.001)$$

1.2. Data

```
T <- 100
y <- rep(0,T)
y[1] <- 0
for (t in 2:T) {y[t] <- y[t-1] + rnorm(1,0,0.1)}
mon.names <- c("LP", "tau", paste("mu[",T,"]", sep=""))
parm.names <- c("alpha","phi","log.tau")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

1.3. Initial Values

```
Initial.Values <- c(rep(0,2), log(1))
```

1.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     alpha.mu <- 0; alpha.tau <- 1.0E-3
     phi.mu <- 0; phi.tau <- 1.0E-3
```

```
tau.alpha <- 1.0E-3; tau.beta <- 1.0E-3
### Parameters
alpha <- parm[1]; phi <- parm[2]; tau <- exp(parm[3])
### Log(Prior Densities)
alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
phi.prior <- dnorm(phi, phi.mu, 1/sqrt(phi.tau), log=TRUE)
tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
### Log-Likelihood
mu <- alpha + phi*Data$y
LL <- sum(dnorm(Data$y[2:(Data$T-1)], mu[1:(Data$T-2)],
     1/sqrt(tau), log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + phi.prior + tau.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,tau,mu[Data$T]),
     yhat=mu, parm=parm)
return(Modelout)
}
```

2. Binary Logit

2.1. Form

$$y \sim Bern(\eta)$$

$$\eta = \log[1 + \exp(\mu)]$$

$$\mu = \mathbf{X}\beta$$

$$\beta_j \sim N(0, 1000), \quad j = 1, \dots, J$$

2.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
J <- 3
y <- ifelse(demonsnacks$Calories <= 137, 0, 1)
X <- cbind(1, as.matrix(demonsnacks[,c(7,8)]))
for (j in 2:J) {X[,j] <- CenterScale(X[,j])}
mon.names <- "LP"
parm.names <- rep(NA,J)
for (j in 1:J) {parm.names[j] <- paste("beta[",j,"]",sep="")}
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

2.3. Initial Values

```
Initial.Values <- rep(0,J)
```

2.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     beta.mu <- rep(0,Data$J)
     beta.tau <- rep(1.0E-3,Data$J)
     ### Parameters
     beta <- parm[1:Data$J]
     ### Log(Prior Densities)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     ### Log-Posterior
     mu <- beta %*% t(Data$X)
     eta <- invlogit(mu)
     ### Log-Likelihood
     LL <- sum(dbern(Data$y, eta, log=TRUE))
     ### Log-Posterior
     LP <- LL + sum(beta.prior)
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
          yhat=eta, parm=parm)
     return(Modelout)
     }
```

3. Binomial Probit

3.1. Form

$$y \sim Bin(p, n)$$
$$p = \phi(\mu)$$
$$\mu = \beta_1 + \beta_2 x$$
$$\beta_j \sim N(0, 1000), \quad j = 1, \dots, J$$

where ϕ is the inverse CDF, and J=2.

3.2. Data

```
#10 Trials
exposed <- c(100,100,100,100,100,100,100,100,100,100)
deaths <- c(10,20,30,40,50,60,70,80,90,100)
dose <- c(1,2,3,4,5,6,7,8,9,10)
```

```
J <- 2 #Number of parameters
mon.names <- "LP"
parm.names <- c("beta[1]","beta[2]")
MyData <- list(J=J, n=exposed, mon.names=mon.names, parm.names=parm.names,
    x=dose, y=deaths)
```

3.3. Initial Values

```
Initial.Values <- rep(0,J)
```

3.4. Model

```
Model <- function(parm, Data)
    {
    ### Prior Parameters
    beta.mu <- rep(0,Data$J)
    beta.tau <- rep(1.0E-3,Data$J)
    ### Parameters
    beta <- parm
    ### Log of Prior Densities
    beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
    ### Log-Likelihood
    mu <- beta[1] + beta[2]*Data$x
    mu <- ifelse(mu < -10, -10, mu); mu <- ifelse(mu > 10, 10, mu)
    p <- pnorm(mu)
    LL <- sum(dbinom(Data$y, Data$n, p, log=TRUE))
    ### Log-Posterior
    LP <- LL + sum(beta.prior)
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
        yhat=p, parm=parm)
    return(Modelout)
    }
```

4. Dynamic Linear Model (DLM)

The data is presented so that the time-series is subdivided into three sections: modeled ($t = 1, \dots, T_m$), one-step ahead forecast ($t = T_m+1$), and future forecast [$t = (T_m+2), \dots, T$].

4.1. Form

$$y_t \sim N(\mu_t, \tau_V^{-1}), \quad t = 1, \dots, T_m$$
$$y_t^{new} \sim N(\mu_t, \tau_V^{-1}), \quad t = (T_m + 1), \dots, T$$
$$\mu_t = \alpha + x_t \beta_t, \quad t = 1, \dots, T$$
$$\alpha \sim N(0, 1000)$$

$$\beta_1 \sim N(0, 1000)$$
$$\beta_t \sim N(\beta_{t-1}, \tau_W^{-1}), \quad t = 2, \dots, T$$
$$\tau_V \sim \Gamma(0.001, 0.001)$$
$$\tau_W \sim \Gamma(0.001, 0.001)$$

4.2. Data

```
T <- 20
T.m <- 14
beta.orig <- x <- rep(0,T)
for (t in 2:T) {
beta.orig[t] <- beta.orig[t-1] + rnorm(1,0,0.1)
x[t] <- x[t-1] + rnorm(1,0,0.1)}
y <- 10 + beta.orig*x + rnorm(T,0,0.1)
y[(T.m+2):T] <- NA
mon.names <- rep(NA, (T-T.m))
for (i in 1:(T-T.m)) mon.names[i] <- paste("mu[",(T.m+i),"]", sep="")
parm.names <- rep(NA, T+3)
parm.names[1] <- "alpha"
for (i in 1:T) {parm.names[i+1] <- paste("beta[", i, "]", sep="")}
parm.names[(T+2):(T+3)] <- c("log.beta.w.tau","log.v.tau")
MyData <- list(T=T, T.m=T.m, mon.names=mon.names, parm.names=parm.names,
     x=x, y=y)
```

4.3. Initial Values

```
Initial.Values <- rep(0,T+3)
```

4.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]
     beta <- parm[2:(Data$T+1)]
     beta.w.tau <- exp(parm[Data$T+2])
     v.tau <- exp(parm[Data$T+3])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, 1/sqrt(1.0E-3), log=TRUE)
     beta.prior <- rep(0,Data$T)
     beta.prior[1] <- dnorm(beta[1], 0, 1/sqrt(1.0E-3), log=TRUE)
     beta.prior[2:Data$T] <- dnorm(beta[2:Data$T], beta[1:(Data$T-1)],
          1/sqrt(beta.w.tau), log=TRUE)
     beta.w.tau.prior <- dgamma(beta.w.tau, 0.001, 0.001, log=TRUE)
     v.tau.prior <- dgamma(v.tau, 1.0E-3, 1.0E-3, log=TRUE)
```

```
### Log-Likelihood
mu <- alpha + beta*Data$x
LL <- sum(dnorm(Data$y[1:Data$T.m], mu[1:Data$T.m], 1/sqrt(v.tau),
     log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + sum(beta.prior) + beta.w.tau.prior +
     v.tau.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=mu[(Data$T.m+1):Data$T],
     yhat=mu, parm=parm)
return(Modelout)
}
```

5. Laplace Regression

This linear regression specifies that y is Laplace-distributed, where it is usually Gaussian or normally-distributed. It has been claimed that it should be surprising that the normal distribution became the standard, when the Laplace distribution usually fits better and has wider tails (Kotz, Kozubowski, and Podgorski 2001). Another popular alternative is to use the t-distribution, though it is more computationally expensive to estimate, because it has three parameters. The Laplace distribution has only two parameters, location and scale like the normal distribution, and is computationally easier to fit. This example could be taken one step further, and the parameter vector β could be Laplace-distributed. Laplace's Demon recommends that users experiment with replacing the normal distribution with the Laplace distribution.

5.1. Form

$$y \sim L(\mu, \tau^{-1})$$

$$\mu = \mathbf{X}\beta$$

$$\beta_j \sim N(0, 1000), \quad j = 1, \dots, J$$

$$\tau \sim \Gamma(0.001, 0.001)$$

5.2. Data

```
N <- 10000
J <- 5
X <- matrix(1,N,J)
for (j in 2:J) {X[,j] <- rnorm(N,runif(1,-3,3),runif(1,0.1,1))}
beta <- runif(J,-3,3)
e <- rlaplace(N,0,0.1)
y <- as.vector(beta %*% t(X) + e)
mon.names <- c("LP", "tau")
parm.names <- rep(NA, J+1)
for (j in 1:J) {parm.names[j] <- paste("beta[",j,"]",sep="")}
```

```
parm.names[J+1] <- "log.tau"
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

5.3. Initial Values

```
Initial.Values <- c(rep(0,J), log(1))
```

5.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     beta.mu <- rep(0,Data$J)
     beta.tau <- rep(1.0E-3,Data$J)
     tau.alpha <- 1.0E-3
     tau.beta <- 1.0E-3
     ### Parameters
     beta <- parm[1:Data$J]
     tau <- exp(parm[Data$J+1])
     ### Log(Prior Densities)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
     ### Log-Likelihood
     mu <- beta %*% t(Data$X)
     LL <- sum(dlaplace(Data$y, mu, 1/sqrt(tau), log=TRUE))
     ### Log-Posterior
     LP <- LL + sum(beta.prior) + tau.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, tau), yhat=mu,
          parm=parm)
     return(Modelout)
     }
```

6. Linear Regression

6.1. Form

$$y \sim N(\mu, \tau^{-1})$$

$$\mu = \mathbf{X}\beta$$

$$\beta_j \sim N(0, 1000), \quad j = 1, \ldots, J$$

$$\tau \sim \Gamma(0.001, 0.001)$$

6.2. Data

```
N <- 10000
J <- 5
X <- matrix(1,N,J)
for (j in 2:J) {X[,j] <- rnorm(N,runif(1,-3,3),runif(1,0.1,1))}
beta <- runif(J,-3,3)
e <- rnorm(N,0,0.1)
y <- as.vector(beta %*% t(X) + e)
mon.names <- c("LP", "tau")
parm.names <- rep(NA, J+1)
for (j in 1:J) {parm.names[j] <- paste("beta[",j,"]",sep="")}
parm.names[J+1] <- "log.tau"
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

6.3. Initial Values

```
Initial.Values <- c(rep(0,J), log(1))
```

6.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     beta.mu <- rep(0,Data$J)
     beta.tau <- rep(1.0E-3,Data$J)
     tau.alpha <- 1.0E-3
     tau.beta <- 1.0E-3
     ### Parameters
     beta <- parm[1:Data$J]
     tau <- exp(parm[Data$J+1])
     ### Log(Prior Densities)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
     ### Log-Likelihood
     mu <- beta %*% t(Data$X)
     LL <- sum(dnorm(Data$y, mu, 1/sqrt(tau), log=TRUE))
     ### Log-Posterior
     LP <- LL + sum(beta.prior) + tau.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, tau), yhat=mu,
          parm=parm)
     return(Modelout)
     }
```

7. Multinomial Logit

7.1. Form

$$y_i \sim Cat(p_{i,1:J})$$

$$p_{i,j} = \frac{\phi_{i,j}}{\sum_{j=1}^{J} \phi_{i,j}}, \quad \sum_{j=1}^{J} p_{i,j} = 1$$

$$\phi = \exp(\mu)$$

$$\mu_{i,J} = 0$$

$$\mu_{i,j} = \mathbf{X}_{i,1:K}\beta_{j,1:K}, \quad j = 1, \ldots, (J-1)$$

$$\beta_{j,k} \sim N(0, 1000) \quad j = 1, \ldots, (J-1)$$

7.2. Data

```
y <- x01 <- x02 <- c(1:300)
y[1:100] <- 1
y[101:200] <- 2
y[201:300] <- 3
x01[1:100] <- rnorm(100, 25, 2.5)
x01[101:200] <- rnorm(100, 40, 4.0)
x01[201:300] <- rnorm(100, 35, 3.5)
x02[1:100] <- rnorm(100, 2.51, 0.25)
x02[101:200] <- rnorm(100, 2.01, 0.20)
x02[201:300] <- rnorm(100, 2.70, 0.27)
N <- length(y)
J <- 3 #Number of categories in y
K <- 3 #Number of predictors (including the intercept)
X <- matrix(c(rep(1,N),x01,x02),N,K)
mon.names <- "LP"
parm.names <- c("beta[1,1]","beta[1,2]","beta[1,3]","beta[2,1]",
     "beta[2,2]","beta[2,3]") ### Parameter Names [J,K]
MyData <- list(J=J, K=K, N=N, X=X, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

7.3. Initial Values

```
Initial.Values <- c(rep(0,(J-1)*K))
```

7.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     beta.mu <- rep(0,(Data$J-1)*Data$K)
     beta.tau <- rep(1.0E-3,(Data$J-1)*Data$K)
```

```
### Parameters
beta <- parm
### Log(Prior Densities)
beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
### Log-Posterior
mu <- matrix(0,Data$N,(Data$J-1))
mu[,1] <- beta[1] + beta[2]*Data$X[,2] + beta[3]*Data$X[,3]
mu[,2] <- beta[4] + beta[5]*Data$X[,2] + beta[6]*Data$X[,3]
mu <- ifelse(mu > 700, 700, mu)
mu <- ifelse(mu < -700, -700, mu)
p <- phi <- matrix(c(exp(mu[,1]),exp(mu[,2]),rep(1,Data$N)),
     Data$N, Data$J)
for(j in 1:Data$J) {p[,j] <- phi[,j] / apply(phi,1,sum)}
### Log-Likelihood
Y <- matrix(0,Data$N,Data$J)
for (j in 1:Data$J) {Y[,j] <- ifelse(Data$y == j, 1, 0)}
LL <- sum(Y * log(p))
### Log-Posterior
LP <- LL + sum(beta.prior)
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
     yhat=as.vector(phi), parm=parm)
return(Modelout)
}
```

8. Normal, Multilevel

This is Gelman's school example (Gelman, Carlin, Stern, and Rubin 2004). Note that **LaplacesDemon** is much slower to converge compared to this example that uses the **R2WinBUGS** package (Gelman 2009), an R package on CRAN. However, also note that Laplace's Demon (eventually) provides a better answer (higher ESS, lower DIC, etc.).

8.1. Form

$$y_j \sim N(\theta_j, \tau_j^{-1})$$

$$\theta_j \sim N(\theta_\mu, \theta_\tau^{-1})$$

$$\theta_\mu \sim N(0, 1000)$$

$$\theta_\tau \sim \Gamma(0.001, 0.001)$$

$$\tau_j = sd^{-2}$$

8.2. Data

```
J <- 8
y <- c(28.4, 7.9, -2.8, 6.8, -0.6, 0.6, 18.0, 12.2)
```

```
sd <- c(14.9, 10.2, 16.3, 11.0, 9.4, 11.4, 10.4, 17.6)
mon.names <- c("LP","theta.sigma")
parm.names <- 2*J+2
for (j in 1:J) {parm.names[j] <- paste("theta[",j,"]",sep="")}
parm.names[J+1] <- paste("theta.mu[",j,"]",sep="")
parm.names[J+2] <- paste("log.theta.sigma[",j,"]",sep="")
MyData <- list(J=J, mon.names=mon.names, parm.names=parm.names, sd=sd, y=y)
```

8.3. Initial Values

```
Initial.Values <- rep(0,J+2)
```

8.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperprior Parameters
     theta.mu.mu <- 0
     theta.mu.tau <- 1.0E-3
     ### Prior Parameters
     theta.mu <- parm[Data$J+1]
     theta.sigma <- exp(parm[Data$J+2])
     tau.alpha <- 1.0E-3
     tau.beta <- 1.0E-3
     ### Parameters
     theta <- parm[1:Data$J]; tau <- 1/(sd*sd)
     ### Log(Prior Densities)
     theta.mu.prior <- dnorm(theta.mu, theta.mu.mu,
          1/sqrt(theta.mu.tau), log=TRUE)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
     theta.prior <- dnorm(theta, theta.mu, theta.sigma, log=TRUE)
     ### Log-Likelihood
     LL <- sum(dnorm(Data$y, theta, 1/sqrt(tau), log=TRUE))
     ### Log-Posterior
     LP <- LL + theta.mu.prior + sum(theta.prior) + sum(tau.prior)
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, theta.sigma),
          yhat=theta, parm=parm)
     return(Modelout)
     }
```

9. Poisson Regression

9.1. Form

$$
\begin{aligned}
y &\sim Pois(\lambda) \\
\lambda &= \exp(\mathbf{X}\beta) \\
\beta_j &\sim N(0, 1000), \quad j = 1, \dots, J
\end{aligned}
$$

9.2. Data

```
N <- 10000
J <- 5
X <- matrix(runif(N*J,-2,2),N,J); X[,1] <- 1
beta <- runif(J,-2,2)
y <- as.vector(round(exp(beta %*% t(X))))
mon.names <- "LP"
parm.names <- rep(NA,J)
for (j in 1:J) {parm.names[j] <- paste("beta[",j,"]",sep="")}
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

9.3. Initial Values

```
Initial.Values <- rep(0,J)
```

9.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     beta.mu <- rep(0,Data$J)
     beta.tau <- rep(1.0E-3,Data$J)
     ### Parameters
     beta <- parm
     ### Log(Prior Densities)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     ### Log-Likelihood
     lambda <- exp(beta %*% t(Data$X))
     LL <- sum(dpois(Data$y, lambda, log=TRUE))
     ### Log-Posterior
     LP <- LL + sum(beta.prior)
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
          yhat=lambda, parm=parm)
     return(Modelout)
     }
```

10. Seemingly Unrelated Regression (SUR)

The following data was used by Zellner (1962) when introducing the Seemingly Unrelated Regression methodology. In this particular example, the elements of the precision matrix Ω are constrained. Hopefully I will soon discover, or someone can show me, a better way of maintaining positive-definiteness. Nonetheless, this form seems to work well.

10.1. Form

$$Y_{t,k} \sim N_K(\mu, \Sigma), \quad t = 1, \dots, T; \quad k = 1, \dots, K$$

$$\mu_{1,t} = \alpha_1 + \alpha_2 X_{t,1} + \alpha_3 X_{t,2}, \quad t = 1, \dots, T$$

$$\mu_{2,t} = \beta_1 + \beta_2 X_{t,3} + \beta_3 X_{t,4}, \quad t = 1, \dots, T$$

$$\Sigma = \Omega^{-1}$$

$$\Omega \sim Wishart(\nu, S), \quad 1.0E - 7 \leq \Omega_{i,l}$$

$$S = R^{-1}$$

$$\alpha_j \sim N(0, 1000), \quad j = 1, \dots, J$$

$$\beta_j \sim N(0, 1000), \quad j = 1, \dots, J$$

where each element i and l of precision matrix Ω is constrained to the interval [1.0E-7,∞), J=3, K=2, and T=20.

10.2. Data

```
T <- 20
year <- c(1935,1936,1937,1938,1939,1940,1941,1942,1943,1944,1945,1946,
     1947,1948,1949,1950,1951,1952,1953,1954)
IG <- c(33.1,45.0,77.2,44.6,48.1,74.4,113.0,91.9,61.3,56.8,93.6,159.9,
     147.2,146.3,98.3,93.5,135.2,157.3,179.5,189.6)
VG <- c(1170.6,2015.8,2803.3,2039.7,2256.2,2132.2,1834.1,1588.0,1749.4,
     1687.2,2007.7,2208.3,1656.7,1604.4,1431.8,1610.5,1819.4,2079.7,
     2371.6,2759.9)
CG <- c(97.8,104.4,118.0,156.2,172.6,186.6,220.9,287.8,319.9,321.3,319.6,
     346.0,456.4,543.4,618.3,647.4,671.3,726.1,800.3,888.9)
IW <- c(12.93,25.90,35.05,22.89,18.84,28.57,48.51,43.34,37.02,37.81,
     39.27,53.46,55.56,49.56,32.04,32.24,54.38,71.78,90.08,68.60)
VW <- c(191.5,516.0,729.0,560.4,519.9,628.5,537.1,561.2,617.2,626.7,
     737.2,760.5,581.4,662.3,583.8,635.2,723.8,864.1,1193.5,1188.9)
CW <- c(1.8,0.8,7.4,18.1,23.5,26.5,36.2,60.8,84.4,91.2,92.4,86.0,111.1,
     130.6,141.8,136.7,129.7,145.5,174.8,213.5)
Y <- matrix(c(IG,IW),T,2)
R <- matrix(0.001,2,2); diag(R) <- 1
mon.names <- c("LP","Sigma[1,1]","Sigma[2,1]","Sigma[1,2]","Sigma[2,2]")
parm.names <- c("alpha[1]","alpha[2]","alpha[3]","beta[1]","beta[2]",
     "beta[3]","log.Omega[1,1]","log.Omega[2,1]","log.Omega[2,2]")
MyData <- list(R=R, T=T, Y=Y, CG=CG, CW=CW, IG=IG, IW=IW, VG=VG, VW=VW,
```

```
    mon.names=mon.names, parm.names=parm.names)
```

10.3. Initial Values

```
Initial.Values <- c(rep(0,6), log(as.vector(R)[c(1,2,4)]))
```

10.4. Model

```
Model <- function(parm, Data)
     {
     ### Constraints
     Omega <- matrix(exp(parm[c(7,8,8,9)]),NROW(Data$R),NROW(Data$R))
     Omega <- ifelse(Omega < 1.0E-7, 1.0E-7, Omega)
     parm[7:9] <- log(as.vector(Omega)[c(1,2,4)])
     ### Prior Parameters
     alpha.mu <- rep(0,3)
     alpha.tau <- rep(1.0E-3,3)
     beta.mu <- rep(0,3)
     beta.tau <- rep(1.0E-3,3)
     ### Parameters
     alpha <- parm[1:3]
     beta <- parm[4:6]
     R <- Data$R
     S <- solve(R)
     Sigma <- solve(Omega)
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     Omega.prior <- log(dwishart(Omega,NROW(R),S))
     ### Log-Likelihood
     mu <- matrix(0,T,2)
     mu[,1] <- alpha[1] + alpha[2]*Data$CG + alpha[3]*Data$VG
     mu[,2] <- beta[1] + beta[2]*Data$CW + beta[3]*Data$VW
     LL <- rep(0, NROW(Data$Y))
     for (i in 1:length(LL)) {
          LL[i] <- sum(dmvn(Data$Y[i,], mu[i,], Sigma, log=TRUE))}
     ### Log-Posterior
     LP <- sum(LL) + sum(alpha.prior) + sum(beta.prior) + Omega.prior
     Modelout <- list(LP=LP, Dev=-2*sum(LL),
          Monitor=c(LP, as.vector(Sigma)), yhat=as.vector(mu), parm=parm)
     return(Modelout)
     }
```

11. Zero-Inflated Poisson (ZIP)

11.1. Form

$$y \sim Pois(\Lambda_{1:N,2})$$
$$z \sim Bern(\Lambda_{1:N,1})$$
$$z_i = 1 \quad when \quad y_i = 0, \quad else \quad z_i = 0, \quad i = 1, \dots, N$$
$$\Lambda_{i,2} = 0 \quad if \quad \Lambda_{i,1} \geq 0.5, \quad i = 1, \dots, N$$
$$\Lambda_{1:N,1} = \log[1 + \exp(\mathbf{X}_1 \alpha)]$$
$$\Lambda_{1:N,2} = \exp(\mathbf{X}_2 \beta)$$
$$\alpha_j \sim N(0, 1000), \quad j = 1, \dots, J_1$$
$$\beta_j \sim N(0, 1000), \quad j = 1, \dots, J_2$$

11.2. Data

```
N <- 1000
J1 <- 4
J2 <- 3
X1 <- matrix(runif(N*J1,-2,2),N,J1); X1[,1] <- 1
X2 <- matrix(runif(N*J2,-2,2),N,J2); X2[,1] <- 1
alpha <- runif(J1,-1,1)
beta <- runif(J2,-1,1)
p <- as.vector(invlogit(alpha %*% t(X1) + rnorm(N,0,0.1)))
mu <- as.vector(round(exp(beta %*% t(X2) + rnorm(N,0,0.1))))
y <- ifelse(p > 0.5, 0, mu)
z <- ifelse(y == 0, 1, 0)
mon.names <- "LP"
parm.names <- rep(NA, J1+J2)
for (j in 1:J1) {parm.names[j] <- paste("alpha[", j, "]", sep="")}
for (j in 1:J2) {parm.names[J1+j] <- paste("beta[", j, "]", sep="")}
MyData <- list(J1=J1, J2=J2, N=N, X1=X1, X2=X2, mon.names=mon.names,
     parm.names=parm.names, y=y, z=z)
```

11.3. Initial Values

```
Initial.Values <- c(alpha,beta)
```

11.4. Model

```
Model <- function(parm, Data)
     {
     ### Prior Parameters
     alpha.mu <- rep(0, Data$J1)
     alpha.tau <- rep(1.0E-3, Data$J1)
     beta.mu <- rep(0, Data$J2)
```

```
beta.tau <- rep(1.0E-3, Data$J2)
### Parameters
alpha <- parm[1:Data$J1]
beta <- parm[(Data$J1+1):(Data$J1 + Data$J2)]
### Log(Prior Densities)
alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
### Log-Likelihood
Lambda <- matrix(NA, Data$N, 2)
Lambda[,1] <- invlogit(alpha %*% t(Data$X1))
Lambda[,2] <- exp(beta %*% t(Data$X2))
Lambda[,2] <- ifelse(Lambda[,1] >= 0.5, 0, Lambda[,2])
LL1 <- sum(dbern(Data$z, Lambda[,1], log=TRUE))
LL2 <- sum(dpois(Data$y, Lambda[,2], log=TRUE))
### Log-Posterior
LP <- LL1 + LL2 + sum(alpha.prior) + sum(beta.prior)
Modelout <- list(LP=LP, Dev=-2*LL2, Monitor=LP,
     yhat=Lambda[,2], parm=parm)
return(Modelout)
}
```

References

Gelman A (2009). ***R2WinBUGS**: Running WinBUGS and OpenBUGS from R / S-PLUS.* R package version 2.1-16, URL http://www.R-project.org/package=R2WinBUGS.

Gelman A, Carlin J, Stern H, Rubin D (2004). *Bayesian Data Analysis*. 2nd edition. Chapman & Hall, Boca Raton, FL.

Hall B (2011). ***LaplacesDemon**: Software for Bayesian Inference.* R package version 11.03.06, URL http://www.R-project.org/package=LaplacesDemon.

Kotz S, Kozubowski T, Podgorski K (2001). *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance.* Birkauser, Boston.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Zellner A (1962). "An Efficient Method of Estimating Seemingly Unrelated Regression Equations and Tests for Aggregation Bias." *Journal of the American Statistical Assocation*, **57**, 348–368.

Affiliation:

Byron Hall
STATISTICAT, LLC
Farmington, CT
E-mail: statisticat@gmail.com
URL: http://www.statisticat.com/laplacesdemon.html