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 2011). 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. ANOVA, One-Way

When $J = 2$, this is a Bayesian form of a t-test.

1.1. Form

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

$$\mu_i = \alpha + \beta[x_i], \quad i = 1, \ldots, N$$

$$\alpha \sim \mathcal{N}(0, 1000)$$

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

$$\beta_J = -\sum_{j=1}^{J-1} \beta_j$$

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

1.2. Data

```
N <- 100
J <- 5
x <- round(runif(N, 0.5, J+0.49))
alpha <- runif(1,-1,1)
beta <- runif(J,-2,2)
beta[1] <- -sum(beta[2:J])
sigma <- runif(J,0.1,0.5)
y <- rep(NA, N)
for (i in 1:N) {y[i] <- alpha + beta[x[i]] + runif(1,0,sigma[x[i]])}
mon.names <- c("LP","beta[1]","tau")
parm.names <- parm.names(list(alpha=0, beta=rep(0,J-1), log.tau=0))
MyData <- list(J=J, N=N, mon.names=mon.names, parm.names=parm.names, x=x,
     y=y)
```

1.3. Initial Values

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

1.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     alpha.mu <- 0
     alpha.tau <- 1.0E-3
     beta.mu <- rep(0,Data$J-1)
     beta.tau <- rep(1.0E-3,Data$J-1)
     tau.alpha <- 1.0E-3
     tau.beta <- 1.0E-3
     ### Parameters
     alpha <- parm[1]
     beta <- rep(NA,Data$J)
     beta[1:(Data$J-1)] <- parm[2:Data$J]
     beta[J] <- -sum(beta[1:(Data$J-1)]) #Sum-to-zero constraint
     tau <- exp(parm[Data$J+1])
     ### 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)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
```

```
    ### Log-Likelihood
    mu <- rep(NA, Data$N)
    for (j in 1:Data$J) {
         mu <- ifelse(Data$x == j, alpha + beta[j], mu)}
    LL <- sum(dnorm(Data$y, mu, 1/sqrt(tau), log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + sum(beta.prior) + tau.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,beta[Data$J],tau),
         yhat=mu, parm=parm)
    return(Modelout)
    }
```

2. Autoregression, AR(1)

2.1. Form

$$y_t \sim \mathcal{N}(\mu_{t-1}, \tau^{-1}), \quad t = 2, \dots, (T-1)$$

$$y_T^{new} \sim \mathcal{N}(\mu_T, \tau^{-1})$$

$$\mu_t = \alpha + \phi y_t, \quad t = 1, \dots, T$$

$$\alpha \sim \mathcal{N}(0, 1000)$$

$$\phi \sim \mathcal{N}(0, 1000)$$

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

2.2. 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)
```

2.3. Initial Values

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

2.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    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)
    }
```

3. Binary Logit

3.1. Form

$$y \sim \mathrm{Bern}(\eta)$$
$$\eta = \frac{1}{1+\exp(-\mu)}$$
$$\mu = \mathbf{X}\beta$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, J$$

3.2. 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 <- parm.names(list(beta=rep(0,J)))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

3.3. Initial Values

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

3.4. Model

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

4. Binary Probit

4.1. Form

$$y \sim \text{Bern}(p)$$
$$p = \phi(\mu)$$
$$\mu = \mathbf{X}\beta$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$

4.2. 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 <- parm.names(list(beta=rep(0,J)))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

4.3. Initial Values

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

4.4. Model

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

5. Binomial Logit

5.1. Form

$$y \sim \mathrm{Bin}(p, n)$$

$$p = \frac{1}{1 + \exp(-\mu)}$$

$$\mu = \beta_1 + \beta_2 x$$

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

where ϕ is the inverse CDF, and J=2.

5.2. 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)
```

5.3. Initial Values

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

5.4. Model

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

6. Binomial Probit

6.1. Form

$$y \sim \mathrm{Bin}(p, n)$$

$$p = \phi(\mu)$$

$$\mu = \beta_1 + \beta_2 x$$

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

where ϕ is the inverse CDF, and J=2.

6.2. 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)
```

6.3. Initial Values

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

6.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    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
    mu <- beta[1] + beta[2]*Data$x
    mu <- interval(mu, -10, 10)
    p <- pnorm(mu)
    LL <- sum(dbinom(Data$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)
    }
```

7. Contingency Table

The two-way contingency table, matrix $\mathbf{Y}$, can easily be extended to more dimensions. For this example, it is vectorized as y, and used like an ANOVA data set. Contingency table $\mathbf{Y}$ has J rows and K columns. The cell counts are fit with Poisson regression, according to intercept α, main effects β_j for each row, main effects γ_k for each column, and interaction effects $\delta_{j,k}$ for dependence effects. An omnibus (all cells) test of independence is done by estimating two models (one with δ, and one without), and a large enough Bayes Factor indicates a violation of independence when the model with δ fits better than the model without δ. In an ANOVA-like style, main effects contrasts can be used to distinguish rows or groups of rows from each other, as well as with columns. Likewise, interaction effects contrasts can be used to test independence in groups of $\delta_{j,k}$ elements. Finally, single-cell interactions can be used to indicate violations of independence for a given cell, such as when zero is not within its 95% probability interval. Although a little different, this example is similar to a method presented by Albert (1997).

7.1. Form

$$\mathbf{Y}_{j,k} \sim \text{Pois}(\lambda_{j,k}), \quad j = 1,\dots,J, \quad k = 1,\dots,K$$
$$\lambda_{j,k} = \exp(\alpha + \beta_j + \gamma_k + \delta_{j,k}), \quad j = 1,\dots,J, \quad k = 1,\dots,K$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_j \sim \mathcal{N}(0, \beta_\tau^{-1}), \quad j = 1,\dots,J$$
$$\beta_\tau \sim \Gamma(0.001, 0.001)$$
$$\gamma_k \sim \mathcal{N}(0, \gamma_\tau^{-1}), \quad k = 1,\dots,K$$
$$\gamma_\tau \sim \Gamma(0.001, 0.001)$$
$$\delta_{j,k} \sim \mathcal{N}(0, \delta_\tau^{-1})$$
$$\delta_\tau \sim \Gamma(0.001, 0.001)$$

7.2. Data

```
J <- 4 #Rows
K <- 4 #Columns
Y <- matrix(c(10,20,60,20, 40,30,10,40, 10,10,40,10, 40,50,1,40), J, K,
     dimnames=list(c("Chrysler","Ford","Foreign","GM"),
     c("I-4","I-6","V-6","V-8")))
y <- as.vector(Y)
N <- length(y) #Cells
r <- rep(1:J, N/J)
c <- rep(1,K)
for (k in 2:K) {c <- c(c, rep(k, K))}
mon.names <- c("LP","beta.tau","gamma.tau","delta.tau")
parm.names <- parm.names(list(alpha=0, beta=rep(0,J), gamma=rep(0,J),
     log.b.tau=0, log.g.tau=0, log.d.tau=0, delta=matrix(0,J,K)))
MyData <- list(J=J, K=K, N=N, c=c, mon.names=mon.names,
     parm.names=parm.names, r=r, y=y)
```

7.3. Initial Values

```
Initial.Values <- c(0, rep(0,J), rep(0,K), rep(0,3), rep(0,J*K))
```

7.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     beta.tau <- exp(parm[grep("log.b.tau", Data$parm.names)])
     gamma.tau <- exp(parm[grep("log.g.tau", Data$parm.names)])
     delta.tau <- exp(parm[grep("log.d.tau", Data$parm.names)])
     ### Parameters
     alpha <- parm[grep("alpha", Data$parm.names)]
     beta <- parm[min(grep("beta", Data$parm.names)):max(
          grep("beta", Data$parm.names))]
     gamma <- parm[min(grep("gamma", Data$parm.names)):max(
          grep("gamma", Data$parm.names))]
     delta <- matrix(parm[min(grep("delta",
          Data$parm.names)):max(grep("delta", Data$parm.names))],
          Data$J, Data$K)
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, 1/sqrt(0.001), log=TRUE)
     beta.prior <- dnorm(beta, 0, 1/sqrt(beta.tau), log=TRUE)
     beta.tau.prior <- dgamma(beta.tau, 1.0E-3, 1.0E-3, log=TRUE)
     gamma.prior <- dnorm(gamma, 0, 1/sqrt(gamma.tau), log=TRUE)
     gamma.tau.prior <- dgamma(gamma.tau, 1.0E-3, 1.0E-3, log=TRUE)
     delta.prior <- dnorm(delta, 0, 1/sqrt(delta.tau), log=TRUE)
     delta.tau.prior <- dgamma(delta.tau, 1.0E-3, 1.0E-3, log=TRUE)
     ### Log-Likelihood
     lambda <- rep(NA, Data$N)
     for (i in 1:Data$N) {
          lambda[i] <- exp(alpha + beta[r[i]] + gamma[c[i]] +
               delta[r[i],c[i]])}
     LL <- sum(dpois(Data$y, lambda, log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + sum(beta.prior) + beta.tau.prior +
          sum(gamma.prior) + gamma.tau.prior + sum(delta.prior) +
          delta.tau.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, beta.tau,
          gamma.tau, delta.tau), yhat=lambda, parm=parm)
     return(Modelout)
     }
```

8. Dynamic Linear Model (DLM)

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

8.1. Form

$$y_t \sim \mathcal{N}(\mu_t, \tau_V^{-1}), \quad t = 1, \dots, T_m$$
$$y_t^{new} \sim \mathcal{N}(\mu_t, \tau_V^{-1}), \quad t = (T_m + 1), \dots, T$$
$$\mu_t = \alpha + x_t \beta_t, \quad t = 1, \dots, T$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_1 \sim \mathcal{N}(0, 1000)$$
$$\beta_t \sim \mathcal{N}(\beta_{t-1}, \tau_W^{-1}), \quad t = 2, \dots, T$$
$$\tau_V \sim \Gamma(0.001, 0.001)$$
$$\tau_W \sim \Gamma(0.001, 0.001)$$

8.2. 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 <- parm.names(list(alpha=0, beta=rep(0,T), log.beta.w.tau=0,
     log.v.tau=0))
MyData <- list(T=T, T.m=T.m, mon.names=mon.names, parm.names=parm.names,
     x=x, y=y)
```

8.3. Initial Values

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

8.4. 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)
    }
```

9. Factor Analysis, Confirmatory

Factor scores are in matrix $\mathbf{F}$, factor loadings for each variable are in vector λ, and f is vector that indicates which variable loads on which factor.

9.1. Form

$$\mathbf{Y}_{i,m} \sim \mathcal{N}(\mu_{i,m}, \tau_m^{-1}), \quad i = 1, \dots, N, \quad m = 1, \dots, M$$
$$\mu_{i,m} = \alpha_m + \lambda_m \mathbf{F}_{i,f[m]}, \quad i = 1, \dots, N, \quad m = 1, \dots, M$$
$$\mathbf{F}_{i,1:P} \sim N_P(\gamma, \Omega^{-1}), \quad i = 1, \dots, N$$
$$\alpha_m \sim \mathcal{N}(0, 1000), \quad m = 1, \dots, M$$
$$\lambda_m \sim \mathcal{N}(0, 1000), \quad m = 1, \dots, M$$
$$\Omega \sim \mathrm{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_P$$

9.2. Data

```
data(swiss)
Y <- cbind(swiss$Agriculture, swiss$Examination, swiss$Education,
     swiss$Catholic, swiss$Infant.Mortality)
M <- NCOL(Y) #Number of variables
N <- NROW(Y) #Number of records
P <- 3 #Number of factors
f <- c(1,3,2,2,1) #Indicator f for the factor for each variable m
gamma <- rep(0,P)
```

```
S <- diag(P)
mon.names <- c("LP","mu[1,1]")
parm.names <- parm.names(list(F=matrix(0,N,P), lambda=rep(0,M),
    Omega=diag(P), alpha=rep(0,M), log.tau=rep(0,M)),
    uppertri=c(0,0,1,0,0))
MyData <- list(M=M, N=N, P=P, S=S, Y=Y, f=f, gamma=gamma,
    mon.names=mon.names, parm.names=parm.names)
```

9.3. Initial Values

```
Initial.Values <- c(rep(0, N*P), rep(0, M),
    S[upper.tri(S, diag=TRUE)], rep(0,M), rep(0,M))
```

9.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    alpha.mu <- rep(0, Data$M)
    alpha.tau <- rep(1.0E-3, Data$M)
    lambda.mu <- rep(0, Data$M)
    lambda.tau <- rep(1.0E-3, Data$M)
    tau.alpha <- rep(1.0E-3, Data$M)
    tau.beta <- rep(1.0E-3, Data$M)
    ### Parameters
    alpha <- parm[min(grep("alpha", Data$parm.names)):max(grep("alpha",
        Data$parm.names))]
    lambda <- parm[min(grep("lambda", Data$parm.names)):max(grep("lambda",
        Data$parm.names))]
    tau <- exp(parm[min(grep("log.tau", Data$parm.names)):max(grep(
        "log.tau", Data$parm.names))])
    F <- matrix(parm[min(grep("F", Data$parm.names)):max(grep("F",
        Data$parm.names))], Data$N, Data$P)
    Omega <- matrix(NA, Data$P, Data$P)
    Omega[upper.tri(Omega, diag=TRUE)] <- parm[min(grep("Omega",
        Data$parm.names)):max(grep("Omega", Data$parm.names))]
    Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
    Sigma <- solve(Omega)
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
    lambda.prior <- dnorm(lambda, lambda.mu, 1/sqrt(lambda.tau),
        log=TRUE)
    tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
    Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
    F.prior <- rep(NA, Data$N)
    for (i in 1:Data$N) {
        F.prior[i] <- dmvn(F[i,], Data$gamma, Sigma, log=TRUE)}
    ### Log-Likelihood
    mu <- Data$Y
    for (m in 1:Data$M) { mu[,m] <- alpha[m] + lambda[m] * F[,Data$f[m]]}
    LL <- sum(dnorm(Data$Y, mu, 1/sqrt(tau), log=TRUE))
    ### Log-Posterior
    LP <- sum(LL) + sum(alpha.prior) + sum(lambda.prior) +
        sum(tau.prior) + sum(F.prior) + Omega.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,mu[1,1]),
        yhat=mu, parm=parm)
    return(Modelout)
    }
```

10. Factor Analysis, Exploratory

Factor scores are in matrix **F** and factor loadings are in matrix Λ. Although the calculation for the recommended number of factors to explore P is also provided below (Fokoue 2004), this example sets $P = 3$.

10.1. Form

$$\mathbf{Y}_{i,m} \sim \mathcal{N}(\mu_{i,m}, \tau_m^{-1}), \quad i = 1, \dots, N, \quad m = 1, \dots, M$$

$$\mu_{i,m} = \alpha_m + \sum_{p=1}^{P} \nu_{i,m,p}, \quad i = 1, \dots, N, \quad m = 1, \dots, M$$

$$\nu_{i,m,p} = \mathbf{F}_{i,p}\Lambda_{p,m}, \quad i = 1, \dots, N, \quad m = 1, \dots, M, \quad p = 1, \dots, P$$

$$\mathbf{F}_{i,1:P} \sim N_P(\gamma, \Omega^{-1}), \quad i = 1, \dots, N$$

$$\alpha_m \sim \mathcal{N}(0, 1000), \quad m = 1, \dots, M$$

$$\gamma_p = 0, \quad p = 1, \dots, P$$

$$\Lambda_{p,m} \sim \mathcal{N}(0, 1000), \quad p = 1, \dots, P, \quad m = 1, \dots, M$$

$$\Omega \sim \mathrm{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_P$$

$$\tau_m \sim \Gamma(0.001, 0.001), \quad m = 1, \dots, M$$

10.2. Data

```
data(swiss)
Y <- cbind(swiss$Agriculture, swiss$Examination, swiss$Education,
swiss$Catholic, swiss$Infant.Mortality)
M <- NCOL(Y) #Number of variables
N <- NROW(Y) #Number of records
P <- trunc(0.5*(2*M + 1 - sqrt(8*M + 1))) #Number of factors to explore
P <- 3 #Number of factors to explore (override for this example)
gamma <- rep(0,P)
```

```
S <- diag(P)
mon.names <- c("LP","mu[1,1]")
parm.names <- parm.names(list(F=matrix(0,N,P), Lambda=matrix(0,P,M),
    Omega=diag(P), alpha=rep(0,M), log.tau=rep(0,M)),
    uppertri=c(0,0,1,0,0))
MyData <- list(M=M, N=N, P=P, S=S, Y=Y, gamma=gamma, mon.names=mon.names,
    parm.names=parm.names)
```

10.3. Initial Values

```
Initial.Values <- c(rep(0, (N*P + P*M)),
    S[upper.tri(S, diag=TRUE)], rep(0,M), rep(0,M))
```

10.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    alpha.mu <- rep(0, Data$M)
    alpha.tau <- rep(1.0E-3, Data$M)
    Lambda.mu <- 0
    Lambda.tau <- 1.0E-3
    tau.alpha <- rep(1.0E-3, Data$M)
    tau.beta <- rep(1.0E-3, Data$M)
    ### Parameters
    alpha <- parm[min(grep("alpha", Data$parm.names)):max(grep("alpha",
        Data$parm.names))]
    tau <- exp(parm[min(grep("log.tau", Data$parm.names)):max(grep(
        "log.tau", Data$parm.names))])
    F <- matrix(parm[min(grep("F", Data$parm.names)):max(grep("F",
        Data$parm.names))], Data$N, Data$P)
    Lambda <- matrix(parm[min(grep("Lambda", Data$parm.names)):max(grep(
        "Lambda", Data$parm.names))], Data$P, Data$M)
    Omega <- matrix(NA, Data$P, Data$P)
    Omega[upper.tri(Omega, diag=TRUE)] <- parm[min(grep("Omega",
        Data$parm.names)):max(grep("Omega", Data$parm.names))]
    Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
    Sigma <- solve(Omega)
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
    tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
    Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
    F.prior <- rep(NA, Data$N)
    for (i in 1:Data$N) {
        F.prior[i] <- dmvn(F[i,], Data$gamma, Sigma, log=TRUE)}
    Lambda.prior <- dnorm(Lambda, Lambda.mu, 1/sqrt(Lambda.tau),
        log=TRUE)
    ### Log-Likelihood
    mu <- Data$Y
    nu <- array(NA, dim=c(Data$N, Data$M, Data$P))
    for (p in 1:Data$P) {nu[, ,p] <- F[,p, drop=FALSE] %*% Lambda[p,]}
    for (m in 1:Data$M) {mu[,m] <- alpha[m] + apply(nu[,1,],1,sum)}
    LL <- sum(dnorm(Data$Y, mu, 1/sqrt(tau), log=TRUE))
    ### Log-Posterior
    LP <- sum(LL) + sum(alpha.prior) + sum(tau.prior) + Omega.prior +
        sum(F.prior) + sum(Lambda.prior)
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,mu[1,1]),
        yhat=mu, parm=parm)
    return(Modelout)
    }
```

11. 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.

11.1. Form

$$y \sim \mathrm{L}(\mu, \tau^{-1})$$
$$\mu = \mathbf{X}\beta$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\tau \sim \Gamma(0.001, 0.001)$$

11.2. 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 <- parm.names(list(beta=rep(0,J), log.tau=0))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

11.3. Initial Values

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

11.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     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)
     }
```

12. Linear Regression

12.1. Form

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

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

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

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

12.2. 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 <- parm.names(list(beta=rep(0,J), log.tau=0))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

12.3. Initial Values

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

12.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     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)
     }
```

13. Linear Regression, Multilevel

13.1. Form

$$
\begin{gathered}
y \sim \mathcal{N}(\mu, \tau^{-1}) \\
\mu_i = \mathbf{X}\beta_{m[i],1:J} \\
\beta_{g,1:J} \sim \mathcal{N}_J(\gamma, \Sigma), \quad g = 1, \ldots, G \\
\Sigma = \Omega^{-1} \\
\Omega \sim \mathrm{W}(J, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_J \\
\gamma_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J \\
\tau \sim \Gamma(0.001, 0.001)
\end{gathered}
$$

where m is a vector of length N, and each element indicates the multilevel group ($g = 1, \ldots, G$) for the associated record.

13.2. Data

```
N <- 30
J <- 2 ### Number of predictors (including intercept)
G <- 2 ### Number of Multilevel Groups
X <- matrix(rnorm(N,0,1),N,J); X[,1] <- 1
Sigma <- matrix(runif(J*J,-1,1),J,J)
diag(Sigma) <- runif(J,1,5)
gamma <- runif(J,-1,1)
beta <- matrix(NA,G,J)
for (g in 1:G) {beta[g,] <- rmvn(1, gamma, Sigma)}
m <- round(runif(N,0.5,(G+0.49))) ### Multilevel group indicator
y <- rep(NA,N)
for (i in 1:N) {y[i] = sum(beta[m[i],] * X[i,]) + rnorm(1,0,0.1)}
S <- diag(J)
mon.names <- c("LP","tau")
parm.names <- parm.names(list(beta=matrix(0,G,J), log.tau=0,
     gamma=rep(0,J), Omega=diag(P)), uppertri=c(0,0,0,1))
MyData <- list(G=G, J=J, N=N, S=S, X=X, m=m, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

13.3. Initial.Values

```
Initial.Values <- c(rep(0,G*J), log(1), rep(0,J),
     S[upper.tri(S, diag=TRUE)])
```

13.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     gamma.mu <- rep(0,Data$J)
     gamma.tau <- rep(1.0E-3,Data$J)
     tau.alpha <- 1.0E-3
     tau.beta <- 1.0E-3
     ### Parameters
     beta <- matrix(parm[1:(Data$G * Data$J)], Data$G, Data$J)
     gamma <- parm[min(grep("gamma", Data$parm.names)):max(grep(
          "gamma", Data$parm.names))]
     tau <- exp(parm[grep("log.tau", Data$parm.names)])
     Omega <- matrix(NA, Data$J, Data$J)
     Omega[upper.tri(Omega, diag=TRUE)] <- parm[min(grep("Omega",
          Data$parm.names)):  max(grep("Omega", Data$parm.names))]
     Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
     Sigma <- solve(Omega)
     ### Log(Prior Densities)
     Omega.prior <- dwishart(Omega, Data$J, Data$S, log=TRUE)
     beta.prior <- rep(0,Data$G)
     for (g in 1:Data$G) {
          beta.prior[g] <- dmvn(beta[g,], gamma, Sigma, log=TRUE)}
     gamma.prior <- dnorm(gamma, gamma.mu, 1/sqrt(gamma.tau), log=TRUE)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
     ### Log-Likelihood
     mu <- Data$y
     for (i in 1:Data$N) {mu[i] <- sum(beta[Data$m[i],] * Data$X[i,])}
     LL <- sum(dnorm(Data$y, mu, 1/sqrt(tau), log=TRUE))
     ### Log-Posterior
     LP <- LL + Omega.prior + sum(beta.prior) + sum(gamma.prior) +
          tau.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,tau),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

14. Linear Regression with Full Missingness

With ‘full missingness’, there are missing values for both the response and at least one predictor. This is a minimal example, since there are missing values in only one of the predictors. Initial values do not need to be specified for missing values in a predictor, unless another predictor variable with missing values is used to predict the missing values of a predictor. More effort is involved in specifying a model with a missing predictor that is predicted by another missing predictor. The full likelihood approach to full missingness is excellent as long as the model is identifiable. When it is not identifiable, then imputation may be done in a previous stage. In this example, `X[,2]` is the only predictor with missing values.

14.1. Form

$$y \sim \mathcal{N}(\mu_2, \tau_2^{-1})$$
$$\mu_2 = \mathbf{X}\beta$$
$$X_{1:N,2} \sim \mathcal{N}(\mu_1, \tau_1^{-1})$$
$$\mu_1 = \mathbf{X}_{1:N,(1,3:J)}\alpha$$
$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J-1$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\tau_k \sim \Gamma(0.001, 0.001), \quad k = 1, \ldots, 2$$

14.2. Data

```
N <- 1000
J <- 5
X <- matrix(runif(N*J,-2,2),N,J)
X[,1] <- 1
alpha <- runif((J-1),-2,2)
X[,2] <- alpha %*% t(X[,-2]) + rnorm(N,0,0.1)
beta <- runif(J,-2,2)
y <- as.vector(beta %*% t(X) + rnorm(N,0,0.1))
y[sample(1:N, round(N*0.05))] <- NA
X[sample(1:N, round(N*0.05)),2] <- NA
mon.names <- c("LP","tau[1]","tau[2]")
parm.names <- parm.names(list(alpha=rep(0,J-1), beta=rep(0,J),
     log.tau=rep(0,2)))
MyData <- list(J=J, N=N, X=X, mon.names=mon.names, parm.names=parm.names,
     y=y)
```

14.3. Initial Values

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

14.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     alpha.mu <- rep(0,(Data$J-1))
     alpha.tau <- rep(1.0E-3,(Data$J-1))
     beta.mu <- rep(0,Data$J)
     beta.tau <- rep(1.0E-3,Data$J)
     tau.alpha <- rep(1.0E-3,2)
     tau.beta <- rep(1.0E-3,2)
     ### Parameters
     alpha <- parm[1:(Data$J-1)]
     beta <- parm[Data$J:(2*Data$J - 1)]
     tau <- exp(parm[(2*Data$J):(2*Data$J+1)])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
     ### Log-Likelihood
     mu1 <- alpha %*% t(Data$X[,-2])
     X.imputed <- Data$X
     X.imputed[,2] <- ifelse(is.na(Data$X[,2]), mu1, Data$X[,2])
     LL1 <- sum(dnorm(X.imputed[,2], mu1, 1/sqrt(tau[1]), log=TRUE))
     mu2 <- beta %*% t(X.imputed)
     y.imputed <- ifelse(is.na(Data$y), mu2, Data$y)
     LL2 <- sum(dnorm(y.imputed, mu2, 1/sqrt(tau[2]), log=TRUE))
     ### Log-Posterior
     LP <- LL1 + LL2 + sum(alpha.prior) + sum(beta.prior) + sum(tau.prior)
     Modelout <- list(LP=LP, Dev=-2*LL2, Monitor=c(LP,tau),
          yhat=mu2, parm=parm)
     return(Modelout)
     }
```

15. Linear Regression with Missing Response

Initial values do not need to be specified for missing values in this response, y. Instead, at each iteration, missing values in y are replaced with their estimate in μ.

15.1. Form

$$y \sim \mathcal{N}(\mu, \tau^{-1})$$
$$\mu = \mathbf{X}\beta$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, J$$
$$\tau \sim \Gamma(0.001, 0.001)$$

15.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
J <- NCOL(demonsnacks)
y <- log(demonsnacks$Calories)
y[sample(1:N, round(N*0.05))] <- NA
X <- cbind(1, as.matrix(demonsnacks[,c(1,3:10)]))
for (j in 2:J) {X[,j] <- CenterScale(X[,j])}
mon.names <- c("LP","tau")
```

```
parm.names <- parm.names(list(beta=rep(0,J), log.tau=0))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

15.3. Initial Values

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

15.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     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$X)
     y.imputed <- ifelse(is.na(Data$y), mu, Data$y)
     LL <- sum(dnorm(y.imputed, 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)
     }
```

16. Multinomial Logit

16.1. Form

$$y_i \sim \mathrm{Cat}(p_{i,1:J}), \quad i = 1, \dots, N$$

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

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

$$\mu_{i,J} = 0, \quad i = 1, \dots, N$$

$$\mu_{i,j} = \mathbf{X}_{i,1:K}\beta_{j,1:K}, \quad j = 1, \ldots, (J-1)$$
$$\beta_{j,k} \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, (J-1), \quad k = 1, \ldots, K$$

16.2. 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)
```

16.3. Initial Values

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

16.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     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 <- sum(dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE))
     ### Log-Posterior
     mu <- Y <- matrix(0,Data$N,Data$J)
     mu[,1] <- beta[1:3] %*% t(Data$X)
     mu[,2] <- beta[4:6] %*% t(Data$X)
     mu <- interval(mu, -700, 700)
     phi <- exp(mu)
```

```
    p <- phi / apply(phi,1,sum)
    for (j in 1:Data$J) {Y[,j] <- ifelse(Data$y == j, 1, 0)}
    LL <- sum(Y * log(p))
    ### Log-Posterior
    LP <- LL + beta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=p, parm=parm)
    return(Modelout)
    }
```

17. Multinomial Logit, Nested

17.1. Form

$$y_i \sim \mathrm{Cat}(P_{i,1:J}), \quad i = 1, \dots, N$$

$$P_{1:N,1} = \frac{R}{R + \exp(\alpha I)}$$

$$P_{1:N,2} = \frac{(1 - P_{1:N,1})S_{1:N,1}}{V}$$

$$P_{1:N,3} = \frac{(1 - P_{1:N,1})S_{1:N,2}}{V}$$

$$R_{1:N} = \exp(\mu_{1:N,1})$$

$$S_{1:N,1:2} = \exp(\mu_{1:N,2:3})$$

$$I = \log(V)$$

$$V_i = \sum_{k=1}^{K} S_{i,k}, \quad i = 1, \dots, N$$

$$\mu_{1:N,1} = \mathbf{X}\iota$$

$$\mu_{1:N,2} = \mathbf{X}\beta_{2,1:K}$$

$$\iota = \alpha\beta_{1,1:K}$$

$$\alpha \sim \mathrm{Exp}(1) \quad T[0,2]$$

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

where there are $J = 3$ categories of y, $K = 3$ predictors, R is the non-nested alternative, S is the nested alternative, V is the observed utility in the nest, α is effectively 1 - correlation and has a truncated exponential distribution, and ι is a vector of regression effects for the isolated alternative after α is taken into account. The third alternative is the reference category.

17.2. 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 <- c("LP",parm.names(list(iota=rep(0,K))))
parm.names <- parm.names(list(alpha=0, beta=matrix(0,J-1,K)))
MyData <- list(J=J, K=K, N=N, X=X, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

17.3. Initial Values

```
Initial.Values <- c(0.5, rep(0.1,(J-1)*K))
```

17.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     alpha.rate <- 1
     beta.mu <- 0; beta.tau <- 1.0E-3
     ### Parameters
     alpha <- interval(parm[1],0,2); parm[1] <- alpha
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
     ### Log(Prior Densities)
     alpha.prior <- dtrunc(alpha, "exp", a=0, b=2, rate=alpha.rate,
          log=TRUE)
     beta.prior <- sum(dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE))
     ### Log-Likelihood
     mu <- P <- Y <- matrix(0,Data$N,Data$J)
     iota <- alpha * beta[1,]
     mu[,1] <- iota %*% t(Data$X)
     mu[,2] <- beta[2,] %*% t(Data$X)
     mu <- interval(mu, -700, 700)
     R <- exp(mu[,1])
     S <- exp(mu[,2:3])
     V <- apply(S,1,sum)
     I <- log(V)
     P[,1] <- R / (R + exp(alpha*I))
     P[,2] <- (1 - P[,1]) * S[,1] / V
```

```
    P[,3] <- (1 - P[,1]) * S[,2] / V
    for (j in 1:Data$J) {Y[,j] <- ifelse(Data$y == j, 1, 0)}
    LL <- sum(Y * log(P))
    ### Log-Posterior
    LP <- LL + alpha.prior + beta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,iota), yhat=P,
         parm=parm)
    return(Modelout)
    }
```

18. 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.).

18.1. Form

$$
\begin{gathered}
y_j \sim \mathcal{N}(\theta_j, \tau_j^{-1}), \quad j = 1, \ldots, J \\
\theta_j \sim \mathcal{N}(\theta_\mu, \theta_\tau^{-1}), \quad j = 1, \ldots, J \\
\theta_\mu \sim \mathcal{N}(0, 1000) \\
\theta_\tau = \frac{1}{\theta_\sigma^2} \\
\sigma \sim \mathrm{U}(1.0E-100, 100) \\
\tau_j = \sigma_j^{-2}, \quad j = 1, \ldots, J
\end{gathered}
$$

18.2. 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.tau")
parm.names <- parm.names(list(theta=rep(0,J), theta.mu=0, sigma=0))
MyData <- list(J=J, mon.names=mon.names, parm.names=parm.names, sd=sd, y=y)
```

18.3. Initial Values

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

18.4. Model

```
Model <- function(parm, Data)
    {
```

```
### Hyperhyperparameters
theta.mu.mu <- 0
theta.mu.tau <- 1.0E-3
### Hyperparameters
theta.mu <- parm[Data$J+1]
sigma <- interval(parm[grep("sigma", Data$parm.names)], 1.0E-100, 100)
parm[grep("sigma", Data$parm.names)] <- sigma
theta.tau <- 1 / sigma^2
tau.alpha <- 1.0E-3
tau.beta <- 1.0E-3
### Parameters
theta <- parm[1:Data$J]; tau <- 1/(Data$sd*Data$sd)
### Log(Prior Densities)
theta.mu.prior <- dnorm(theta.mu, theta.mu.mu,
     1/sqrt(theta.mu.tau), log=TRUE)
sigma.prior <- dunif(sigma, 1.0E-100, 100, log=TRUE)
tau.prior <- sum(dgamma(tau, tau.alpha, tau.beta, log=TRUE))
theta.prior <- sum(dnorm(theta, theta.mu, 1/sqrt(theta.tau), log=TRUE))
### Log-Likelihood
LL <- sum(dnorm(Data$y, theta, 1/sqrt(tau), log=TRUE))
### Log-Posterior
LP <- LL + theta.mu.prior + sigma.prior + theta.prior + tau.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, theta.tau),
     yhat=theta, parm=parm)
return(Modelout)
}
```

19. Panel, Autoregressive Poisson

19.1. Form

$$\mathbf{Y} \sim \mathrm{Pois}(\Lambda)$$

$$\Lambda_{1:N,1} = \exp(\alpha + \beta x)$$

$$\Lambda_{1:N,t} = \exp(\alpha + \beta x + \rho \mathbf{Y}_{1:N,t-1}), \quad t = 2, \dots, T$$

$$\alpha_i \sim \mathcal{N}_N(\alpha_\mu, \Sigma), \quad i = 1, \dots, N$$

$$\alpha_\mu \sim \mathcal{N}(0, 1000)$$

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

$$\Omega \sim \mathrm{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_N$$

$$\beta \sim \mathcal{N}(0, 1000)$$

19.2. Data

```
N <- 10
T <- 5
alpha <- runif(N,0,1)
rho <- 0.2
beta <- 0.1
x <- runif(N,0,1)
Y <- matrix(NA,N,T)
Y[,1] <- exp(alpha + beta*x)
for (t in 2:T) {Y[,t] <- exp(alpha + beta*x + rho*Y[,t-1])}
Y <- round(Y)
S <- diag(N)
mon.names <- c("LP")
parm.names <- parm.names(list(alpha=rep(0,N), alpha.mu=0, Omega=diag(N),
    beta=0, rho=0), uppertri=c(0,0,1,0,0))
MyData <- list(N=N, S=S, T=T, Y=Y, mon.names=mon.names,
    parm.names=parm.names, x=x)
```

19.3. Initial Values

```
Initial.Values <- c(rep(0,N), 0, S[upper.tri(S, diag=TRUE)], 0, 0)
```

19.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    alpha.mu.mu <- 0
    alpha.mu.tau <- 1.0E-3
    alpha.mu <- parm[Data$N+1]
    beta.mu <- 0; beta.tau <- 1.0E-3
    rho.mu <- 0; rho.tau <- 1.0E-3
    ### Parameters
    alpha <- parm[1:Data$N]
    beta <- parm[grep("beta", Data$parm.names)]
    rho <- parm[grep("rho", Data$parm.names)]
    Omega <- matrix(NA, Data$N, Data$N)
    Omega[upper.tri(Omega, diag=TRUE)] <- parm[min(grep("Omega",
        Data$parm.names)):max(grep("Omega", Data$parm.names))]
    Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
    Sigma <- solve(Omega)
    ### Log(Prior Densities)
    alpha.mu.prior <- dnorm(alpha.mu, alpha.mu.mu, 1/sqrt(alpha.mu.tau),
        log=TRUE)
    Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
    alpha.prior <- dmvn(alpha, rep(alpha.mu, Data$N), Sigma, log=TRUE)
    beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
    rho.prior <- dnorm(rho, rho.mu, 1/sqrt(rho.tau), log=TRUE)
    ### Log-Likelihood
    Lambda <- Data$Y
    Lambda[,1] <- exp(alpha + beta*x)
    Lambda[,2:Data$T] <- exp(alpha + beta*Data$x +
         rho*Data$Y[,1:(Data$T-1)])
    LL <- sum(dpois(Data$Y, Lambda, log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + alpha.mu.prior + Omega.prior + rho.prior +
         beta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=Lambda, parm=parm)
    return(Modelout)
    }
```

20. Poisson Regression

20.1. Form

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

20.2. 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 <- parm.names(list(beta=rep(0,J)))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

20.3. Initial Values

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

20.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    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)
    }
```

21. Seemingly Unrelated Regression (SUR)

The following data was used by Zellner (1962) when introducing the Seemingly Unrelated Regression methodology.

21.1. Form

$$Y_{t,k} \sim \mathcal{N}_K(\mu_{t,k}, \Sigma), \quad t = 1, \ldots, T; \quad k = 1, \ldots, K$$

$$\mu_{1,t} = \alpha_1 + \alpha_2 \mathbf{X}_{t,1} + \alpha_3 \mathbf{X}_{t,2}, \quad t = 1, \ldots, T$$

$$\mu_{2,t} = \beta_1 + \beta_2 \mathbf{X}_{t,3} + \beta_3 \mathbf{X}_{t,4}, \quad t = 1, \ldots, T$$

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

$$\Omega \sim \mathrm{W}(K, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_K$$

$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$

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

where J=3, K=2, and T=20.

21.2. 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)
S <- diag(NCOL(Y))
mon.names <- c("LP","Sigma[1,1]","Sigma[2,1]","Sigma[1,2]","Sigma[2,2]")
parm.names <- parm.names(list(alpha=rep(0,3), beta=rep(0,3),
    Omega=diag(2)), uppertri=c(0,0,1))
MyData <- list(S=S, T=T, Y=Y, CG=CG, CW=CW, IG=IG, IW=IW, VG=VG, VW=VW,
    mon.names=mon.names, parm.names=parm.names)
```

21.3. Initial Values

```
Initial.Values <- c(rep(0,3), rep(0,3), S[upper.tri(S, diag=TRUE)])
```

21.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    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]
    Omega <- matrix(parm[c(7,8,8,9)], NROW(Data$S), NROW(Data$S))
    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 <- dwishart(Omega, NROW(Data$S), Data$S, log=TRUE)
    ### Log-Likelihood
    mu <- matrix(0,Data$T,2)
    mu[,1] <- alpha[1] + alpha[2]*Data$CG + alpha[3]*Data$VG
    mu[,2] <- beta[1] + beta[2]*Data$CW + beta[3]*Data$VW
    LL <- rep(0, Data$T)
    for (t in 1:Data$T) {
        LL[t] <- sum(dmvn(Data$Y[t,], mu[t,], 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=mu, parm=parm)
    return(Modelout)
    }
```

22. Variable Selection

This example uses a modified form of the random-effects (or global adaptation) Stochastic Search Variable Selection (SSVS) algorithm presented in O'Hara and Sillanpaa (2009). Here, SSVS is applied to linear regression, though this method is widely applicable. For J variables, each regression effects vector β_j is conditional on γ_j, a binary inclusion variable. Each β_j is a discrete mixture distribution with respect to $\gamma_j = 0$ or $\gamma_j = 1$, with precision 100 or β_τ, respectively. As with other representations of SSVS, these precisions may require tuning.

With other representations of SSVS, each γ_j is Bernoulli-distributed, though this would be problematic in Laplace's Demon, because γ_j would be in the list of parameters (rather than monitors), and would not be stationary due to switching behavior. To keep γ in the monitors, an uninformative normal density is placed on each prior δ_j, with mean $1/J$ for J variables and precision $1.0E-3$. Each δ_j is transformed with the inverse logit and rounded to γ_j. Note that $\lfloor x + 0.5 \rfloor$ means to round x. The prior for δ can be manipulated to influence sparseness.

When the goal is to select the best model, each $\mathbf{X}_{1:N,j}$ is retained for a future run when the posterior mean of $\gamma_j \geq 0.5$. When the goal is model-averaging, the results of this model may be used directly.

22.1. Form

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

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

$$(\beta_j|\gamma_j) \sim (1-\gamma_j)\mathcal{N}(0, 100) + \gamma_j\mathcal{N}(0, \beta_\tau^{-1}) \quad j = 1, \dots, J$$

$$\beta_\tau = \frac{1}{\sigma^2}$$

$$\sigma \sim \mathrm{U}(1.0E-100, 100)$$

$$\gamma_j = \lfloor \frac{1}{1+\exp(-\delta_j)} + 0.5 \rfloor, \quad j = 1, ..., J$$

$$\delta_j \sim \mathcal{N}(0, 10), \quad j = 1, \dots, J$$

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

22.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
```

```
J <- NCOL(demonsnacks)
y <- log(demonsnacks$Calories)
X <- cbind(1, as.matrix(demonsnacks[,c(1,3:10)]))
for (j in 2:J) {X[,j] <- CenterScale(X[,j])}
mon.names <- c("LP", "beta.tau", "tau", parm.names(list(gamma=rep(0,J))))
parm.names <- parm.names(list(beta=rep(0,J), delta=rep(0,J), sigma=0,
    log.tau=0))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

22.3. Initial Values

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

22.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperhyperparameters
    delta.mu <- logit(1/Data$J)
    delta.tau <- 1.0E-3
    ### Hyperparameters
    beta.mu <- rep(0, Data$J)
    beta.sigma <- interval(parm[grep("sigma", Data$parm.names)], 1.0E-100, 100)
    parm[grep("sigma", Data$parm.names)] <- beta.sigma     beta.tau <- 1 / beta.sigma^2
    delta <- parm[(Data$J+1):(2*Data$J)]
    tau.alpha <- 1.0E-3
    tau.beta <- 1.0E-3
    ### Parameters
    beta <- parm[1:Data$J]
    gamma <- round(invlogit(delta))
    beta.tau <- ifelse(gamma == 0, 100, beta.tau)
    tau <- exp(parm[grep("log.tau", Data$parm.names)])
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE))
    beta.sigma.prior <- dunif(beta.sigma, 1.0E-100, 100, log=TRUE)
    delta.prior <- sum(dnorm(delta, delta.mu, 1/sqrt(delta.tau),
        log=TRUE))
    tau.prior <- dgamma(tau, tau.alpha, tau.beta, log=TRUE)
    ### Log-Likelihood
    mu <- beta %*% t(Data$X)
    LL <- sum(dnorm(y, mu, 1/sqrt(tau), log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior + beta.sigma.prior + delta.prior + tau.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, min(beta.tau),
        tau, gamma), yhat=mu, parm=parm)
    return(Modelout)
    }
```

23. Zero-Inflated Poisson (ZIP)

23.1. Form

$$y \sim \text{Pois}(\Lambda_{1:N,2})$$

$$z \sim \text{Bern}(\Lambda_{1:N,1})$$

$$z_i = 1 \quad \text{when} \quad y_i = 0, \quad \text{else} \quad z_i = 0, \quad i = 1, \dots, N$$

$$\Lambda_{i,2} = 0 \quad \text{if} \quad \Lambda_{i,1} \geq 0.5, \quad i = 1, \dots, N$$

$$\Lambda_{1:N,1} = \frac{1}{1 + \exp(-\mathbf{X}_1 \alpha)}$$

$$\Lambda_{1:N,2} = \exp(\mathbf{X}_2 \beta)$$

$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, J_1$$

$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, J_2$$

23.2. 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 <- parm.names(list(alpha=rep(0,J1), beta=rep(0,J2)))
MyData <- list(J1=J1, J2=J2, N=N, X1=X1, X2=X2, mon.names=mon.names,
     parm.names=parm.names, y=y, z=z)
```

23.3. Initial Values

```
Initial.Values <- rep(0,J1+J2)
```

23.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     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$J1]
     beta <- parm[min(grep("beta", Data$parm.names)):max(grep(
          "beta", Data$parm.names))]
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, alpha.mu, 1/sqrt(alpha.tau), log=TRUE)
     beta.prior <- dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE)
     ### Log-Likelihood
     Lambda <- matrix(NA, Data$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

Albert J (1997). "Bayesian Testing and Estimation of Association in a Two-Way Contingency Table." *Journal of the American Statistical Association*, **92**(438), 685–693.

Fokoue E (2004). "Stochastic Determination of the Intrinsic Structure in Bayesian Factor Analysis." Technical Report 2004-17, Statistical and Mathematical Sciences Institute, Research Triangle Park, NC, www.samsi.info.

Gelman A (2009). ***R2WinBUGS**: Running WinBUGS and OpenBUGS from R / S-PLUS.* R package version 2.1-18, URL http://cran.r-project.org/web/packages/R2WinBUGS/index.html.

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 11.03.27, URL http://cran.r-project.org/web/packages/LaplacesDemon/index.html.

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

O'Hara R, Sillanpaa M (2009). "A Review of Bayesian Variable Selection Methods: What, How and Which." *Journal of Bayesian Analysis*, **4**(1), 85–118.

R Development Core Team (2011). *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`