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 `laplacesdemon@statisticat.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

$$y \sim \mathcal{N}(\mu, \sigma_1^2)$$
$$\mu_i = \alpha + \beta[x_i], \quad i = 1, \ldots, N$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_j \sim \mathcal{N}(0, \sigma_2^2), \quad j = 1, \ldots, (J-1)$$
$$\beta_J = -\sum_{j=1}^{J-1} \beta_j$$
$$\sigma_{1:2} \sim \mathcal{HC}(25)$$

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[J] <- -sum(beta[1:(J-1)])
y <- rep(NA, N)
for (i in 1:N) {y[i] <- alpha + beta[x[i]] + rnorm(1,0,0.2)}
mon.names <- c("LP","beta[1]","sigma[1]","sigma[2]")
parm.names <- parm.names(list(alpha=0, beta=rep(0,J-1),
    log.sigma=rep(0,2)))
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)), rep(log(1),2))
```

1.4. Model

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

```
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
### Log(Prior Densities)
alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
beta.prior <- dnorm(beta, 0, sigma[2], log=TRUE)
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
### Log-Likelihood
mu <- rep(NA, Data$N)
for (j in 1:Data$J) {
    mu <- ifelse(Data$x == j, alpha + beta[j], mu)}
LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + sum(beta.prior) + sum(sigma.prior)
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,beta[Data$J],
    sigma), yhat=mu, parm=parm)
return(Modelout)
}
```

2. ANOVA, Two-Way

In this representation, σ^m are the superpopulation variance components, `s.beta` and `s.gamma` are the finite-population within-variance components of the factors or treatments, and `s.error` is the finite-population between-variance component.

2.1. Form

$$
\begin{aligned}
y_i &\sim \mathcal{N}(\mu_i, \sigma_1^2) \\
\mu_i &= \alpha + \beta[\mathbf{X}_{i,1}] + \gamma[\mathbf{X}_{i,2}], \quad i = 1, \ldots, N \\
\epsilon_i &= y_i - \mu_i \\
\alpha &\sim \mathcal{N}(0, 1000) \\
\beta_j &\sim \mathcal{N}(0, \sigma_2^2), \quad j = 1, \ldots, (J-1) \\
\beta_J &= -\sum_{j=1}^{J-1} \beta_j \\
\gamma_k &\sim \mathcal{N}(0, \sigma_3^2), \quad k = 1, \ldots, (K-1) \\
\beta_K &= -\sum_{k=1}^{K-1} \gamma_k \\
\sigma^m &\sim \mathcal{HC}(25), \quad m = 1, \ldots, 3
\end{aligned}
$$

2.2. Data

```
N <- 100
J <- 5 #Number of levels in factor (treatment) 1
```

```
K <- 3 #Number of levels in factor (treatment) 2
X <- matrix(cbind(round(runif(N, 0.5, J+0.49)),round(runif(N,0.5,K+0.49))),
    N, 2)
alpha <- runif(1,-1,1)
beta <- runif(J,-2,2)
beta[J] <- -sum(beta[1:(J-1)])
gamma <- runif(K,-2,2)
gamma[J] <- -sum(gamma[1:(K-1)])
y <- rep(NA, N)
for (i in 1:N) {y[i] <- alpha + beta[X[i,1]] + gamma[X[i,2]] +
    rnorm(1,0,0.1)}
mon.names <- c("LP","beta[5]","gamma[3]","sigma[1]","sigma[2]","sigma[3]",
    "s.beta","s.gamma","s.epsilon")
parm.names <- parm.names(list(alpha=0, beta=rep(0,J-1), gamma=rep(0,K-1),
    log.sigma=rep(0,3)))
MyData <- list(J=J, K=K, N=N, X=X, mon.names=mon.names,
    parm.names=parm.names, y=y)
```

2.3. Initial Values

```
Initial.Values <- c(0, rep(0,(J-1)), rep(0,(K-1)), rep(log(1),3))
```

2.4. Model

```
Model <- function(parm, Data)
    {
    ### 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
    gamma <- rep(NA,Data$K)
    gamma[1:(Data$K-1)] <- parm[grep("gamma", Data$parm.names)]
    gamma[K] <- -sum(gamma[1:(Data$K-1)]) #Sum-to-zero constraint
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    beta.prior <- dnorm(beta, 0, sigma[2], log=TRUE)
    gamma.prior <- dnorm(gamma, 0, sigma[3], log=TRUE)
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu <- rep(NA, Data$N)
    for (j in 1:Data$J) {
        mu <- ifelse(Data$X[,1] == j, alpha + beta[j], mu)}
    for (k in 1:Data$K) {
        mu <- ifelse(Data$X[,2] == k, mu + gamma[k], mu)}
```

```
LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
### Variance Components
s.beta <- sd(beta)
s.gamma <- sd(gamma)
s.epsilon <- sd(Data$y - mu)
### Log-Posterior
LP <- LL + alpha.prior + sum(beta.prior) + sum(gamma.prior) +
     sum(sigma.prior)
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, beta[Data$J],
     gamma[Data$K], sigma, s.beta, s.gamma, s.epsilon), yhat=mu,
     parm=parm)
return(Modelout)
}
```

3. Autoregression, AR(1)

3.1. Form

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

$$y_T^{new} \sim \mathcal{N}(\mu_T, \sigma^2)$$

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

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

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

$$\sigma \sim \mathcal{HC}(25)$$

3.2. 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", "sigma", paste("mu[",T,"]", sep=""))
parm.names <- c("alpha","phi","log.sigma")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

3.3. Initial Values

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

3.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]; phi <- parm[2]; sigma <- exp(parm[3])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
     sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
     ### Log-Likelihood
     mu <- alpha + phi*Data$y
     LL <- sum(dnorm(Data$y[2:(Data$T-1)], mu[1:(Data$T-2)],
          sigma, log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + phi.prior + sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma,mu[Data$T]),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

4. Binary Logit

4.1. Form

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

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)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    mu <- beta %*% t(Data$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)
    }
```

5. Binary Probit

5.1. Form

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

5.2. 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)
```

5.3. Initial Values

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

5.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    mu <- beta %*% t(Data$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)
    }
```

6. Binomial Logit

6.1. Form

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

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

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

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

where ϕ is the inverse CDF, and J=2.

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)
    {
    ### Parameters
    beta <- parm
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), 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)
    }
```

7. Binomial Probit

7.1. Form

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

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

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

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

where ϕ is the inverse CDF, and J=2.

7.2. 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)
```

7.3. Initial Values

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

7.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm
    ### Log(Prior Densities)
    beta.prior <- dnorm(beta, 0, sqrt(1000), 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)
    }
```

8. Conditional Autoregression (CAR), Poisson

This CAR example a slightly modified form of example 7.3 (Model A) in Congdon (2003). The Scottish lip cancer data also appears in the WinBUGS (Spiegelhalter, Thomas, Best, and Lunn 2003) examples and is a widely analyzed example. The data y consists of counts for $i = 1, \ldots, 56$ counties in Scotland. A single predictor x is provided. The errors, ϵ, are allowed to include spatial effects as smoothing by spatial effects from areal neighbors. Interactions w between counties are in terms of dummy indicators for contiguity (areal neighbors). The list of NN areal neighbors is in the *adj* variable, and cumulative positions are in variable C. The vector ϵ_μ is the mean of each area's error, and is a weighted average of errors in contiguous areas.

8.1. Form

$$y \sim \text{Pois}(\lambda)$$

$$\lambda = \exp(\log(E) + \beta_1 + \beta_2 x + \epsilon)$$

$$\epsilon \sim \mathcal{N}(\epsilon_\mu, \sigma^2)$$

$$\epsilon_{\mu[i]} = \rho \sum_{j=1}^{J} w_{i,j}\epsilon_j, \quad i = 1, \ldots, N$$

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

$$\rho \sim \mathrm{U}(-1, 1)$$

$$\sigma \sim \mathcal{HC}(25)$$

8.2. Data

```
N <- 56 #Number of areas
NN <- 264 #Number of adjacent areas
y <- c(9,39,11,9,15,8,26,7,6,20,13,5,3,8,17,9,2,7,9,7,16,31,11,7,19,15,7,
    10,16,11,5,3,7,8,11,9,11,8,6,4,10,8,2,6,19,3,2,3,28,6,1,1,1,1,0,0)
E <- c( 1.4,8.7,3.0,2.5,4.3,2.4,8.1,2.3,2.0,6.6,4.4,1.8,1.1,3.3,7.8,4.6,
    1.1,4.2,5.5,4.4,10.5,22.7,8.8,5.6,15.5,12.5,6.0,9.0,14.4,10.2,4.8,
    2.9,7.0,8.5,12.3,10.1,12.7,9.4,7.2,5.3,18.8,15.8,4.3,14.6,50.7,8.2,
    5.6,9.3,88.7,19.6,3.4,3.6,5.7,7.0,4.2,1.8) #Expected
x <- c(16,16,10,24,10,24,10,7,7,16,7,16,10,24,7,16,10,7,7,10,7,16,10,7,1,1,
    7,7,10,10,7,24,10,7,7,0,10,1,16,0,1,16,16,0,1,7,1,1,0,1,1,0,1,1,16,10)
adj <- c(5,9,11,19, #Area 1 is adjacent to areas 5, 9, 11, and 19
         7,10, #Area 2 is adjacent to areas 7 and 10
         6,12,
         18,20,28,
         1,11,12,13,19,
         3,8,
         2,10,13,16,17,
         6,
         1,11,17,19,23,29,
         2,7,16,22,
         1,5,9,12,
         3,5,11,
         5,7,17,19,
         31,32,35,
         25,29,50,
         7,10,17,21,22,29,
         7,9,13,16,19,29,
         4,20,28,33,55,56,
         1,5,9,13,17,
         4,18,55,
         16,29,50,
         10,16,
         9,29,34,36,37,39,
         27,30,31,44,47,48,55,56,
         15,26,29,
```

```
          25,29,42,43,
          24,31,32,55,
          4,18,33,45,
          9,15,16,17,21,23,25,26,34,43,50,
          24,38,42,44,45,56,
          14,24,27,32,35,46,47,
          14,27,31,35,
          18,28,45,56,
          23,29,39,40,42,43,51,52,54,
          14,31,32,37,46,
          23,37,39,41,
          23,35,36,41,46,
          30,42,44,49,51,54,
          23,34,36,40,41,
          34,39,41,49,52,
          36,37,39,40,46,49,53,
          26,30,34,38,43,51,
          26,29,34,42,
          24,30,38,48,49,
          28,30,33,56,
          31,35,37,41,47,53,
          24,31,46,48,49,53,
          24,44,47,49,
          38,40,41,44,47,48,52,53,54,
          15,21,29,
          34,38,42,54,
          34,40,49,54,
          41,46,47,49,
          34,38,49,51,52,
          18,20,24,27,56,
          18,24,30,33,45,55)
# C has length N+1 and refers to cumulative position (-1) in the adj
# variable.  For example, area 1 begins at 0 (position 1-1), and
# area 2 begins at 4 (position 5-1), etc.
C <- c(0,4,6,8,11,16,18,23,24,30,34,38,41,45,48,51,57,63,69,74,77,80,82,
    88,96,99,103,107,111,122,128,135,139,143,152,157,161,166,172,177,182,
    189,195,199,204,208,214,220,224,233,236,240,244,248,253,258,264)
mon.names <- c("LP","sigma")
parm.names <- parm.names(list(beta=rep(0,2), epsilon=rep(0,N), rho=0,
    log.sigma=0))
MyData <- list(C=C, E=E, N=N, NN=NN, adj=adj, mon.names=mon.names,
    parm.names=parm.names, x=x, y=y)
```

8.3. Initial Values

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

8.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:2]
    epsilon <- parm[grep("epsilon", Data$parm.names)]
    rho <- interval(parm[grep("rho", Data$parm.names)], -1, 1)
    parm[grep("rho", Data$parm.names)] <- rho
    w <- epsilon[Data$adj]
    epsilon.mu <- epsilon
    for (i in 1:N) {
        epsilon.mu[i] <- rho * sum(w[(Data$C[i]+1):(Data$C[i+1])])}
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    epsilon.prior <- sum(dnorm(epsilon, epsilon.mu, sigma,
        log=TRUE))
    rho.prior <- dunif(rho, -1, 1, log=TRUE)
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    ### Log-Likelihood
    lambda <- exp(log(Data$E) + beta[1] + beta[2]*Data$x/10 + epsilon)
    LL <- sum(dpois(Data$y, lambda, log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior + epsilon.prior + rho.prior + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma), yhat=lambda,
        parm=parm)
    return(Modelout)
    }
```

9. 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).

9.1. Form

$$\mathbf{Y}_{j,k} \sim \text{Pois}(\lambda_{j,k}), \quad j = 1, \ldots, J, \quad k = 1, \ldots, K$$
$$\lambda_{j,k} = \exp(\alpha + \beta_j + \gamma_k + \delta_{j,k}), \quad j = 1, \ldots, J, \quad k = 1, \ldots, K$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_j \sim \mathcal{N}(0, \beta_\sigma^2), \quad j = 1, \ldots, J$$
$$\beta_\sigma \sim \mathcal{HC}(25)$$
$$\gamma_k \sim \mathcal{N}(0, \gamma_\sigma^2), \quad k = 1, \ldots, K$$
$$\gamma_\sigma \sim \mathcal{HC}(25)$$
$$\delta_{j,k} \sim \mathcal{N}(0, \delta_\sigma^2)$$
$$\delta_\sigma \sim \mathcal{HC}(25)$$

9.2. 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.sigma","gamma.sigma","delta.sigma")
parm.names <- parm.names(list(alpha=0, beta=rep(0,J), gamma=rep(0,J),
    log.b.sigma=0, log.g.sigma=0, log.d.sigma=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)
```

9.3. Initial Values

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

9.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    beta.sigma <- exp(parm[grep("log.b.sigma", Data$parm.names)])
    gamma.sigma <- exp(parm[grep("log.g.sigma", Data$parm.names)])
    delta.sigma <- exp(parm[grep("log.d.sigma", 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$K)
### Log(Prior Densities)
alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
beta.prior <- dnorm(beta, 0, beta.sigma, log=TRUE)
beta.sigma.prior <- dhalfcauchy(beta.sigma, 25, log=TRUE)
gamma.prior <- dnorm(gamma, 0, gamma.sigma, log=TRUE)
gamma.sigma.prior <- dhalfcauchy(gamma.sigma, 25, log=TRUE)
delta.prior <- dnorm(delta, 0, delta.sigma, log=TRUE)
delta.sigma.prior <- dhalfcauchy(delta.sigma, 25, log=TRUE)
### Log-Likelihood
lambda <- rep(NA, Data$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.sigma.prior +
     sum(gamma.prior) + gamma.sigma.prior + sum(delta.prior) +
     delta.sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, beta.sigma,
     gamma.sigma, delta.sigma), yhat=lambda, parm=parm)
return(Modelout)
}
```

10. 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 [$t = (T_m + 2), \ldots, T$].

10.1. Form

$$y_t \sim \mathcal{N}(\mu_t, \sigma_V^2), \quad t = 1, \ldots, T_m$$
$$y_t^{new} \sim \mathcal{N}(\mu_t, \sigma_V^2), \quad t = (T_m + 1), \ldots, T$$
$$\mu_t = \alpha + x_t \beta_t, \quad t = 1, \ldots, T$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_1 \sim \mathcal{N}(0, 1000)$$

$$\beta_t \sim \mathcal{N}(\beta_{t-1}, \sigma_W^2), \quad t = 2, \ldots, T$$
$$\sigma_V \sim \mathcal{HC}(25)$$
$$\sigma_W \sim \mathcal{HC}(25)$$

10.2. 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.sigma=0,
    log.v.sigma=0))
MyData <- list(T=T, T.m=T.m, mon.names=mon.names, parm.names=parm.names,
    x=x, y=y)
```

10.3. Initial Values

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

10.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1]
    beta <- parm[2:(Data$T+1)]
    beta.w.sigma <- exp(parm[Data$T+2])
    v.sigma <- exp(parm[Data$T+3])
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    beta.prior <- rep(0,Data$T)
    beta.prior[1] <- dnorm(beta[1], 0, sqrt(1000), log=TRUE)
    beta.prior[2:Data$T] <- dnorm(beta[2:Data$T], beta[1:(Data$T-1)],
        beta.w.sigma, log=TRUE)
    beta.w.sigma.prior <- dhalfcauchy(beta.w.sigma, 25, log=TRUE)
    v.sigma.prior <- dhalfcauchy(v.sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu <- alpha + beta*Data$x
    LL <- sum(dnorm(Data$y[1:Data$T.m], mu[1:Data$T.m], v.sigma,
```

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

11. 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.

11.1. Form

$$\mathbf{Y}_{i,m} \sim \mathcal{N}(\mu_{i,m}, \sigma_m^2), \quad i = 1, \ldots, N, \quad m = 1, \ldots, M$$

$$\mu_{i,m} = \alpha_m + \lambda_m \mathbf{F}_{i,f[m]}, \quad i = 1, \ldots, N, \quad m = 1, \ldots, M$$

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

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

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

$$\sigma_m \sim \mathcal{HC}(25), \quad m = 1, \ldots, M$$

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

11.2. 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.sigma=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)
```

11.3. Initial Values

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

11.4. Model

```
Model <- function(parm, Data)
    {
    ### 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))]
    sigma <- exp(parm[min(grep("log.sigma", Data$parm.names)):max(grep(
        "log.sigma", Data$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, 0, sqrt(1000), log=TRUE)
    lambda.prior <- dnorm(lambda, 0, sqrt(1000), log=TRUE)
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
    F.prior <- dmvn(F, 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, sigma, log=TRUE))
    ### Log-Posterior
    LP <- sum(LL) + sum(alpha.prior) + sum(lambda.prior) +
        sum(sigma.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)
    }
```

12. Factor Analysis, Exploratory

Factor scores are in matrix $\mathbf{F}$ 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 $P = 3$.

12.1. Form

$$\mathbf{Y}_{i,m} \sim \mathcal{N}(\mu_{i,m}, \sigma_m^2), \quad i = 1, \ldots, N, \quad m = 1, \ldots, M$$

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

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

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

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

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

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

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

$$\sigma_m \sim \mathcal{HC}(25), \quad m = 1, \ldots, M$$

12.2. 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.sigma=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)
```

12.3. Initial Values

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

12.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[min(grep("alpha", Data$parm.names)):max(grep("alpha",
```

```
    Data$parm.names))]
sigma <- exp(parm[min(grep("log.sigma", Data$parm.names)):max(grep(
    "log.sigma", 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$parm.names))]
Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
Sigma <- solve(Omega)
### Log(Prior Densities)
alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
F.prior <- dmvn(F, Data$gamma, Sigma, log=TRUE)
Lambda.prior <- dnorm(Lambda, 0, sqrt(1000), 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, sigma, log=TRUE))
### Log-Posterior
LP <- sum(LL) + sum(alpha.prior) + sum(sigma.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)
}
```

13. Kriging

This is an example of universal kriging of y given $\mathbf{X}$, regression effects β, and spatial effects ζ. Euclidean distance between spatial coordinates (longitude and latitude) is used for each of $i = 1, \ldots, N$ records of y. An additional record is created from the same data-generating process to compare the accuracy of interpolation. For the spatial component, ϕ is the rate of spatial decay and κ is the scale. κ is often difficult to identify, so it is set to 1 (Gaussian), but may be allowed to vary up to 2 (Exponential). In practice, ϕ is also often difficult to identify. While Σ is spatial covariance, spatial correlation is $\rho = \exp(-\phi\mathbf{D})$.

13.1. Form

$$y \sim \mathcal{N}(\mu, \sigma_1^2)$$

$$\mu = \mathbf{X}\beta + \zeta$$
$$y^{new} = \mathbf{X}\beta + \sum_{i=1}^{N}\left(\frac{\rho_i}{\sum \rho}\zeta_i\right)$$
$$\rho = \exp(-\phi \mathbf{D}^{new})^{\kappa}$$
$$\zeta \sim \mathcal{N}_N(\zeta_\mu, \Sigma)$$
$$\Sigma = \sigma_2^2 \exp(-\phi \mathbf{D})^{\kappa}$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, 2$$
$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \ldots, 2$$
$$\phi \sim \mathrm{U}(1, 5)$$
$$\zeta_\mu = 0$$
$$\kappa = 1$$

13.2. Data

```
N <- 20
longitude <- runif(N+1,0,100)
latitude <- runif(N+1,0,100)
D <- as.matrix(dist(cbind(longitude,latitude), diag=TRUE, upper=TRUE))
Sigma <- 10000 * exp(-1.5 * D)
zeta <- as.vector(apply(rmvn(1000, rep(0,N+1), Sigma), 2, mean))
beta <- c(50,2)
X <- matrix(runif((N+1)*2,-2,2),(N+1),2); X[,1] <- 1
mu <- as.vector(beta %*% t(X))
y <- mu + zeta
longitude.new <- longitude[N+1]; latitude.new <- latitude[N+1]
Xnew <- X[N+1,]; ynew <- y[N+1]
longitude <- longitude[1:N]; latitude <- latitude[1:N]
X <- X[1:N,]; y <- y[1:N]
D <- as.matrix(dist(cbind(longitude,latitude), diag=TRUE, upper=TRUE))
D.new <- sqrt((longitude - longitude.new)^2 + (latitude - latitude.new)^2)
mon.names <- c("LP","sigma[1]","sigma[2]","ynew")
parm.names <- parm.names(list(zeta=rep(0,N), beta=rep(0,2),
    log.sigma=rep(0,2), phi=0))
MyData <- list(D=D, D.new=D.new, N=N, X=X, Xnew=Xnew, mon.names=mon.names,
    parm.names=parm.names, y=y)
```

13.3. Initial Values

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

13.4. Model

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

```
### Parameters
beta <- parm[grep("beta", Data$parm.names)]
zeta <- parm[grep("zeta", Data$parm.names)]
kappa <- 1
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
phi <- interval(parm[grep("phi", Data$parm.names)], 1, 5)
parm[grep("phi", Data$parm.names)] <- phi
Sigma <- sigma[2]*sigma[2] * exp(-phi * Data$D)^kappa
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
zeta.prior <- dmvn(zeta, rep(0, Data$N), Sigma, log=TRUE)
sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
phi.prior <- dunif(phi, 1, 5, log=TRUE)
### Interpolation
rho <- exp(-phi * Data$D.new)^kappa
ynew <- sum(beta * Data$Xnew) + sum(rho / sum(rho) * zeta)
### Log-Likelihood
mu <- beta %*% t(Data$X) + zeta
LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + zeta.prior + sigma.prior + phi.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma,ynew),
    yhat=mu, parm=parm)
return(Modelout)
}
```

14. 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 (see Robust Regression in section 24), 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.

14.1. Form

$$y \sim \mathrm{L}(\mu, \sigma^2)$$

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

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

$$\sigma \sim \mathcal{HC}(25)$$

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

14.3. Initial Values

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

14.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    sigma <- exp(parm[Data$J+1])
    ### Log(Prior Densities)
    beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu <- beta %*% t(Data$X)
    LL <- sum(dlaplace(Data$y, mu, sigma, log=TRUE))
    ### Log-Posterior
    LP <- LL + sum(beta.prior) + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, sigma), yhat=mu,
        parm=parm)
    return(Modelout)
    }
```

15. Linear Regression

15.1. Form

$$y \sim \mathcal{N}(\mu, \sigma^2)$$

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

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

$$\sigma \sim \mathcal{HC}(25)$$

15.2. 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", "sigma")
parm.names <- parm.names(list(beta=rep(0,J), log.sigma=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)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    sigma <- exp(parm[Data$J+1])
    ### Log(Prior Densities)
    beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
    sigma.prior <- dgamma(sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu <- beta %*% t(Data$X)
    LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
    ### Log-Posterior
    LP <- LL + sum(beta.prior) + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, sigma), yhat=mu,
        parm=parm)
    return(Modelout)
    }
```

16. Linear Regression, Multilevel

16.1. Form

$$y \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu_i = \mathbf{X}\beta_{m[i],1:J}$$
$$\beta_{g,1:J} \sim \mathcal{N}_J(\gamma, \Sigma), \quad g = 1, \ldots, G$$
$$\Sigma = \Omega^{-1}$$
$$\Omega \sim \mathrm{W}(J, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_J$$
$$\gamma_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\sigma \sim \mathcal{HC}(25)$$

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

16.2. 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","sigma")
parm.names <- parm.names(list(beta=matrix(0,G,J), log.sigma=0,
    gamma=rep(0,J), Omega=S), 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)
```

16.3. Initial.Values

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

16.4. Model

```
Model <- function(parm, Data)
    {
    ### 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))]
sigma <- exp(parm[grep("log.sigma", 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$S, log=TRUE)
beta.prior <- dmvn(beta, gamma, Sigma, log=TRUE)
gamma.prior <- dnorm(gamma, 0, sqrt(1000), log=TRUE)
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
### Log-Likelihood
mu <- Data$y
for (i in 1:Data$N) {mu[i] <- sum(beta[Data$m[i],] * Data$X[i,])}
LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
### Log-Posterior
LP <- LL + Omega.prior + sum(beta.prior) + sum(gamma.prior) +
    sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma),
    yhat=mu, parm=parm)
return(Modelout)
}
```

17. 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.

17.1. Form

$$y \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

$$\mu_2 = \mathbf{X}\beta$$

$$X_{1:N,2} \sim \mathcal{N}(\mu_1, \sigma_1^2)$$

$$\mu_1 = \mathbf{X}_{1:N,(1,3:J)}\alpha$$

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

$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\sigma_k \sim \mathcal{HC}(25), \quad k = 1, \ldots, 2$$

17.2. 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
M <- ifelse(is.na(y), 1, 0)
X[sample(1:N, round(N*0.05)),2] <- NA
mon.names <- c("LP","sigma[1]","sigma[2]")
parm.names <- parm.names(list(alpha=rep(0,J-1), beta=rep(0,J),
    log.sigma=rep(0,2)))
MyData <- list(J=J, M=M, N=N, X=X, mon.names=mon.names, parm.names=parm.names,
    y=y)
```

17.3. Initial Values

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

17.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1:(Data$J-1)]
    beta <- parm[Data$J:(2*Data$J - 1)]
    sigma <- exp(parm[(2*Data$J):(2*Data$J+1)])
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu1 <- alpha %*% t(Data$X[,-2])
    X.imputed <- Data$X
    X.imputed[,2] <- ifelse(is.na(Data$X[,2]), mu1, Data$X[,2])
    LL1 <- sum(dnorm(X.imputed[,2], mu1, sigma[1], log=TRUE))
    mu2 <- beta %*% t(X.imputed)
    y.imputed <- ifelse(is.na(Data$y), mu2, Data$y)
    LL2 <- sum((1-Data$M) * dnorm(y.imputed, mu2, sigma[2], log=TRUE))
    ### Log-Posterior
```

```
    LP <- LL1 + LL2 + sum(alpha.prior) + sum(beta.prior) + sum(sigma.prior)
    Modelout <- list(LP=LP, Dev=-2*LL2, Monitor=c(LP,sigma),
        yhat=mu2, parm=parm)
    return(Modelout)
    }
```

18. 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.

18.1. Form

$$y \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = \mathbf{X}\beta$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\sigma \sim \mathcal{HC}(25)$$

18.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
J <- NCOL(demonsnacks)
y <- log(demonsnacks$Calories)
y[sample(1:N, round(N*0.05))] <- NA
M <- ifelse(is.na(y), 1, 0)
X <- cbind(1, as.matrix(demonsnacks[,c(1,3:10)]))
for (j in 2:J) {X[,j] <- CenterScale(X[,j])}
mon.names <- c("LP","sigma")
parm.names <- parm.names(list(beta=rep(0,J), log.sigma=0))
MyData <- list(J=J, M=M, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

18.3. Initial Values

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

18.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    sigma <- exp(parm[Data$J+1])
```

```
### Log(Prior Densities)
beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
sigma.prior <- dgamma(sigma, 25, log=TRUE)
### Log-Likelihood
mu <- beta %*% t(Data$X)
y.imputed <- ifelse(is.na(Data$y), mu, Data$y)
LL <- sum((1-Data$M) * dnorm(y.imputed, mu, sigma, log=TRUE))
### Log-Posterior
LP <- LL + sum(beta.prior) + sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma),
    yhat=mu, parm=parm)
return(Modelout)
}
```

19. Multinomial Logit

19.1. Form

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

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

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

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

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

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

19.2. 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)
```

19.3. Initial Values

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

19.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    mu <- 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)
    Y <- indmat(Data$y)
    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)
    }
```

20. Multinomial Logit, Nested

20.1. Form

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

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

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

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

$$R_{1:N} = \exp(\mu_{1:N,1})$$
$$S_{1:N,1:2} = \exp(\mu_{1:N,2:3})$$
$$I = \log(V)$$
$$V_i = \sum_{k=1}^{K} S_{i,k}, \quad i = 1, \ldots, N$$
$$\mu_{1:N,1} = \mathbf{X}\iota$$
$$\mu_{1:N,2} = \mathbf{X}\beta_{2,1:K}$$
$$\iota = \alpha\beta_{1,1:K}$$
$$\alpha \sim \text{Exp}(1) \in [0, 2]$$
$$\beta_{j,k} \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, (J-1) \quad k = 1, \ldots, K$$

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.

20.2. 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)
```

20.3. Initial Values

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

20.4. Model

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

```
### Hyperparameters
alpha.rate <- 1
### Parameters
alpha <- interval(parm[1],0,2); parm[1] <- alpha
beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
### Log(Prior Densities)
alpha.prior <- dtrunc(alpha, "exp", a=0, b=2, rate=alpha.rate,
    log=TRUE)
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
### Log-Likelihood
mu <- P <- matrix(0,Data$N,Data$J)
iota <- alpha * beta[1,]
mu[,1] <- iota %*% t(Data$X)
mu[,2] <- beta[2,] %*% t(Data$X)
mu <- interval(mu, -700, 700)
R <- exp(mu[,1])
S <- exp(mu[,2:3])
V <- apply(S,1,sum)
I <- log(V)
P[,1] <- R / (R + exp(alpha*I))
P[,2] <- (1 - P[,1]) * S[,1] / V
P[,3] <- (1 - P[,1]) * S[,2] / V
Y <- indmat(Data$y)
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)
}
```

21. 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.).

21.1. Form

$$\begin{aligned} y_j &\sim \mathcal{N}(\theta_j, \tau_j^{-1}), \quad j = 1, \ldots, J \\ \theta_j &\sim \mathcal{N}(\theta_\mu, \theta_\tau^{-1}), \quad j = 1, \ldots, J \\ \theta_\mu &\sim \mathcal{N}(0, 1000) \\ \theta_\tau &= \frac{1}{\theta_\sigma^2} \end{aligned}$$

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

$$\tau_j = \sigma_j^{-2}, \quad j = 1, \ldots, J$$

21.2. 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)
```

21.3. Initial Values

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

21.4. 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(Hyperprior and 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)
```

```
    }
```

22. Panel, Autoregressive Poisson

22.1. Form

$$\mathbf{Y} \sim \text{Pois}(\Lambda)$$
$$\Lambda_{1:N,1} = \exp(\alpha + \beta x)$$
$$\Lambda_{1:N,t} = \exp(\alpha + \beta x + \rho \log(\mathbf{Y}_{1:N,t-1})), \quad t = 2, \ldots, T$$
$$\alpha_i \sim \mathcal{N}(\alpha_\mu, \alpha_\sigma^2), \quad i = 1, \ldots, N$$
$$\alpha_\mu \sim \mathcal{N}(0, 1000)$$
$$\alpha_\sigma \sim \mathcal{HC}(25)$$
$$\beta \sim \mathcal{N}(0, 1000)$$
$$\rho \sim \mathcal{N}(0, 1000)$$

22.2. Data

```
N <- 10
T <- 10
alpha <- rnorm(N,2,0.5)
rho <- 0.5
beta <- 0.5
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*log(Y[,t-1]))}
Y <- round(Y)
mon.names <- c("LP","alpha.sigma")
parm.names <- parm.names(list(alpha=rep(0,N), alpha.mu=0,
    log.alpha.sigma=0, beta=0, rho=0))
MyData <- list(N=N, T=T, Y=Y, mon.names=mon.names, parm.names=parm.names,
    x=x)
```

22.3. Initial Values

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

22.4. Model

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

```
### Hyperparameters
alpha.mu <- parm[Data$N+1]
alpha.sigma <- exp(parm[Data$N+2])
### Parameters
alpha <- parm[1:Data$N]
beta <- parm[grep("beta", Data$parm.names)]
rho <- parm[grep("rho", Data$parm.names)]
### Log(Hyperprior and Prior Densities)
alpha.mu.prior <- dnorm(alpha.mu, 0, sqrt(1000), log=TRUE)
alpha.sigma.prior <- dhalfcauchy(alpha.sigma, 25, log=TRUE)
alpha.prior <- sum(dnorm(alpha, alpha.mu, alpha.sigma, log=TRUE))
beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
rho.prior <- dnorm(rho, 0, sqrt(1000), log=TRUE)
### Log-Likelihood
Lambda <- Data$Y
Lambda[,1] <- exp(alpha + beta*x)
Lambda[,2:Data$T] <- exp(alpha + beta*Data$x +
    rho*log(Data$Y[,1:(Data$T-1)]))
LL <- sum(dpois(Data$Y, Lambda, log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + alpha.mu.prior + alpha.sigma.prior +
    beta.prior + rho.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,alpha.sigma),
    yhat=Lambda, parm=parm)
return(Modelout)
}
```

23. Poisson Regression

23.1. Form

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

23.2. 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)
```

23.3. Initial Values

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

23.4. 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)
     }
```

24. Robust Regression

By replacing the normal distribution with the Student t distribution, linear regression is often called robust regression. As an alternative approach to robust regression, consider Laplace regression (see section 14).

$$y \sim \mathrm{t}(\mu, \sigma^2, \nu)$$

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

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

$$\sigma \sim \mathcal{HC}(25)$$

$$\nu \sim \mathcal{HC}(25)$$

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

24.2. Initial Values

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

24.3. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    sigma <- exp(parm[Data$J+1])
    nu <- exp(parm[Data$J+2])
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    nu.prior <- dhalfcauchy(nu, 25, log=TRUE)
    ### Log-Likelihood
    mu <- beta %*% t(Data$X)
    LL <- sum(dst(Data$y, mu, sigma, nu, log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior + sigma.prior + nu.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma,nu), yhat=mu,
        parm=parm)
    return(Modelout)
    }
```

25. Seemingly Unrelated Regression (SUR)

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

25.1. 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.

25.2. 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)
```

25.3. Initial Values

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

25.4. Model

```
Model <- function(parm, Data)
    {
    ### 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)
sum(alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE))
sum(beta.prior <- dnorm(beta, 0, sqrt(1000), 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$VW
LL <- sum(dmvn(Data$Y, mu, Sigma, log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + beta.prior + Omega.prior
Modelout <- list(LP=LP, Dev=-2*LL,
    Monitor=c(LP, as.vector(Sigma)), yhat=mu, parm=parm)
return(Modelout)
}
```

26. Space-Time, Nonseparable

This approach to space-time or spatiotemporal modeling applies kriging both to the stationary spatial and temporal components, where space is continuous and time is discrete. Matrix Ξ contains the space-time effects. Spatial coordinates are given in longitude and latitude for $s = 1, \ldots, S$ points in space and measurements are taken across time-periods $t = 1, \ldots, T$ for $\mathbf{Y}_{s,t}$. The dependent variable is also a function of design matrix $\mathbf{X}$ and regression effects vector β. For more information on kriging, see section 13. This example uses a nonseparable, stationary covariance function in which space and time are separable only when $\psi = 0$.

26.1. Form

$$\mathbf{Y}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \sigma_1^2), \quad s = 1, \ldots, S, \quad t = 1, \ldots, T$$

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

$$\Xi \sim \mathcal{N}_{ST}(\Xi_\mu, \Sigma)$$

$$\Sigma = \sigma_2^2 \exp\left(-\frac{D_S}{\phi_1}^{\kappa} - \frac{D_T}{\phi_2}^{\lambda} - \psi\frac{D_S}{\phi_1}^{\kappa}\frac{D_T}{\phi_2}^{\lambda}\right)$$

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

$$\phi_k \sim \mathrm{U}(1, 5), \quad k = 1, \ldots, 2$$

$$\sigma_k \sim \mathcal{HC}(25), \quad k = 1, \ldots, 2$$

$$\psi \sim \mathcal{HC}(25)$$

$$\Xi_\mu = 0$$

$$\kappa = 1, \quad \lambda = 1$$

26.2. Data

```
S <- 10
T <- 5
longitude <- runif(S,0,100)
latitude <- runif(S,0,100)
D.S <- as.matrix(dist(cbind(rep(longitude,T),rep(latitude,T)), diag=TRUE,
    upper=TRUE))
D.T <- as.matrix(dist(cbind(rep(1:T,each=S),rep(1:T,each=S)), diag=TRUE,
    upper=TRUE))
Sigma <- 10000 * exp(-D.S/3 - D.T/2 - 0.2*(D.S/3)*(D.T/2))
Xi <- as.vector(apply(rmvn(1000, rep(0,S*T), Sigma), 2, mean))
Xi <- matrix(Xi,S,T)
beta <- c(50,2)
X <- matrix(runif(S*2,-2,2),S,2); X[,1] <- 1
mu <- as.vector(beta %*% t(X))
Y <- mu + Xi
mon.names <- c("LP","psi","sigma[1]","sigma[2]")
parm.names <- parm.names(list(Xi=matrix(0,S,T), beta=rep(0,2),
    phi=rep(0,2), log.sigma=rep(0,2), log.psi=0))
MyData <- list(D.S=D.S, D.T=D.T, S=S, T=T, X=X, Y=Y, mon.names=mon.names,
    parm.names=parm.names)
```

26.3. Initial Values

```
Initial.Values <- c(rep(0,S*T), mean(Y), 0, rep(1,2), rep(0,2), 0)
```

26.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    Xi.mu <- rep(0,Data$S*Data$T)
    ### Parameters
    beta <- parm[grep("beta", Data$parm.names)]
    Xi <- parm[grep("Xi", Data$parm.names)]
    kappa <- 1; lambda <- 1
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    phi <- interval(parm[grep("phi", Data$parm.names)], 1, 5)
    parm[grep("phi", Data$parm.names)] <- phi
    psi <- exp(parm[grep("log.psi", Data$parm.names)])
    Sigma <- sigma[2]*sigma[2] * exp(-(Data$D.S / phi[1])^kappa -
        (Data$D.T / phi[2])^lambda -
        psi*(Data$D.S / phi[1])^kappa * (Data$D.T / phi[2])^lambda)
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    Xi.prior <- dmvn(Xi, Xi.mu, Sigma, log=TRUE)
```

```
sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
phi.prior <- sum(dunif(phi, 1, 5, log=TRUE))
psi.prior <- dhalfcauchy(psi, 25, log=TRUE)
### Log-Likelihood
Xi <- matrix(Xi, Data$S, Data$T)
mu <- as.vector(beta %*% t(Data$X)) + Xi
LL <- sum(dnorm(Data$Y, mu, sigma[1], log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + Xi.prior + sigma.prior + phi.prior + psi.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,psi,sigma),
    yhat=mu, parm=parm)
return(Modelout)
}
```

27. Space-Time, Separable

This introductory approach to space-time or spatiotemporal modeling applies kriging both to the stationary spatial and temporal components, where space is continuous and time is discrete. Vector ζ contains the spatial effects and vector θ contains the temporal effects. Spatial coordinates are given in longitude and latitude for $s = 1, \ldots, S$ points in space and measurements are taken across time-periods $t = 1, \ldots, T$ for $\mathbf{Y}_{s,t}$. The dependent variable is also a function of design matrix $\mathbf{X}$ and regression effects vector β. For more information on kriging, see section 13. This example uses separable space-time covariances, which is more convenient but usually less appropriate than a nonseparable covariance function.

27.1. Form

$$\mathbf{Y}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \tau_1^{-1}), \quad s = 1, \ldots, S, \quad t = 1, \ldots, T$$

$$\mu_{s,t} = \mathbf{X}_{s,1:J}\beta + \zeta_s + \Theta_{s,t}$$

$$\Theta_{s,1:T} = \theta$$

$$\theta \sim \mathcal{N}_N(\theta_\mu, \Sigma_T)$$

$$\Sigma_T = \frac{1}{\tau_3}\exp(-\phi_2 \mathbf{D}_T)^\lambda$$

$$\zeta \sim \mathcal{N}_N(\zeta_\mu, \Sigma_S)$$

$$\Sigma_S = \frac{1}{\tau_2}\exp(-\phi_1 \mathbf{D}_S)^\kappa$$

$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, 2$$

$$\tau_k \sim \Gamma(0.001, 0.001), \quad k = 1, \ldots, 3$$

$$\phi_k \sim \mathrm{U}(1, 5), \quad k = 1, \ldots, 2$$

$$\zeta_\mu = 0$$

$$\theta_\mu = 0$$

$$\kappa = 1, \quad \lambda = 1$$

27.2. Data

```
S <- 20
T <- 10
longitude <- runif(S,0,100)
latitude <- runif(S,0,100)
D.S <- as.matrix(dist(cbind(longitude,latitude), diag=TRUE, upper=TRUE))
Sigma.S <- 10000 * exp(-1.5 * D.S)
zeta <- as.vector(apply(rmvn(1000, rep(0,S), Sigma.S), 2, mean))
D.T <- as.matrix(dist(cbind(c(1:T),c(1:T)), diag=TRUE, upper=TRUE))
Sigma.T <- 10000 * exp(-3 * D.T)
theta <- as.vector(apply(rmvn(1000, rep(0,T), Sigma.T), 2, mean))
Theta <- matrix(theta,S,T,byrow=TRUE)
beta <- c(50,2)
X <- matrix(runif(S*2,-2,2),S,2); X[,1] <- 1
mu <- as.vector(beta %*% t(X))
Y <- mu + zeta + Theta + matrix(rnorm(S*T,0,0.1),S,T)
mon.names <- c("LP","tau[1]","tau[2]","tau[3]")
parm.names <- parm.names(list(zeta=rep(0,S), theta=rep(0,T),
    beta=rep(0,2), phi=rep(0,2), log.tau=rep(0,3))
MyData <- list(D.S=D.S, D.T=D.T, S=S, T=T, X=X, Y=Y, mon.names=mon.names,
    parm.names=parm.names)
```

27.3. Initial Values

```
Initial.Values <- c(rep(0,S), rep(0,T), rep(0,2), rep(1,2), rep(0,3))
```

27.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperparameters
    beta.mu <- 0
    beta.tau <- 1.0E-3
    zeta.mu <- rep(0,Data$S)
    theta.mu <- rep(0,Data$T)
    tau.alpha <- rep(1.0E-3,3)
    tau.beta <- rep(1.0E-3,3)
    ### Parameters
    beta <- parm[grep("beta", Data$parm.names)]
    zeta <- parm[grep("zeta", Data$parm.names)]
    theta <- parm[grep("theta", Data$parm.names)]
    kappa <- 1; lambda <- 1
    tau <- exp(parm[grep("log.tau", Data$parm.names)])
```

```
phi <- interval(parm[grep("phi", Data$parm.names)], 1, 5)
parm[grep("phi", Data$parm.names)] <- phi
Sigma.S <- 1/tau[2] * exp(-phi[1] * Data$D.S)^kappa
Sigma.T <- 1/tau[3] * exp(-phi[2] * Data$D.T)^lambda
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, beta.mu, 1/sqrt(beta.tau), log=TRUE))
zeta.prior <- dmvn(zeta, zeta.mu, Sigma.S, log=TRUE)
theta.prior <- dmvn(theta, theta.mu, Sigma.T, log=TRUE)
tau.prior <- sum(dgamma(tau, tau.alpha, tau.beta, log=TRUE))
phi.prior <- sum(dunif(phi, 1, 5, log=TRUE))
### Log-Likelihood
Theta <- matrix(theta, Data$S, Data$T, byrow=TRUE)
mu <- as.vector(beta %*% t(Data$X)) + zeta + Theta
LL <- sum(dnorm(Data$Y, mu, 1/sqrt(tau[1]), log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + zeta.prior + theta.prior + tau.prior +
    phi.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,tau),
    yhat=mu, parm=parm)
return(Modelout)
}
```

28. Survival Analysis

Although the dependent variable is usually denoted as t in survival analysis, it is denoted here as y so Laplace's Demon recognizes it as a dependent variable for posterior predictive checks.

28.1. Form

$$y_i \sim \text{Weib}(\gamma, \mu_i), \quad i = 1, \ldots, N$$
$$\mu = \exp(\mathbf{X}\beta)$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\gamma \sim \Gamma(1, 0.001)$$

28.2. Data

```
N <- 50
J <- 5
X <- matrix(runif(N*J,-2,2),N,J); X[,1] <- 1
beta <- runif(J,-1,1)
y <- as.vector(round(exp(beta %*% t(X)))) + 1
mon.names <- c("LP","gamma")
parm.names <- parm.names(list(beta=rep(0,J), log.gamma=0))
```

```
MyData <- list(J=J, N=N, X=X, mon.names=mon.names, parm.names=parm.names,
     y=y)
```

28.3. Initial Values

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

28.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     gamma <- exp(parm[Data$J+1])
     ### Log(Prior Densities)
     beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
     gamma.prior <- dgamma(gamma, 1, 1.0E-3, log=TRUE)
     ### Log-Likelihood
     mu <- exp(beta %*% t(Data$X)) + 1
     h <- (gamma/lambda)*(Data$y/lambda)^(gamma-1)
     S <- exp(-mu * Data$y^gamma)
     LL <- sum(dweibull(Data$y, gamma, mu, log=TRUE))
     ### Log-Posterior
     LP <- LL + sum(beta.prior) + gamma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, gamma),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

29. 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), which selects variables according to practical significance rather than statistical significance. 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.

29.1. Form

$$y \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = \mathbf{X}\beta$$
$$(\beta_j|\gamma_j) \sim (1-\gamma_j)\mathcal{N}(0, 100) + \gamma_j\mathcal{N}(0, \beta_\tau^{-1}) \quad j = 1, \ldots, J$$
$$\beta_\tau = \frac{1}{\sigma^2}$$
$$\sigma \sim \mathrm{U}(1.0E-100, 100)$$
$$\gamma_j = \lfloor \frac{1}{1+\exp(-\delta_j)} + 0.5 \rfloor, \quad j = 1, ..., J$$
$$\delta_j \sim \mathcal{N}(0, 10), \quad j = 1, \ldots, J$$
$$\sigma \sim \mathcal{HC}(25)$$

29.2. 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", "sigma", parm.names(list(gamma=rep(0,J))))
parm.names <- parm.names(list(beta=rep(0,J), delta=rep(0,J), sigma=0,
    log.sigma=0))
MyData <- list(J=J, X=X, mon.names=mon.names, parm.names=parm.names, y=y)
```

29.3. Initial Values

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

29.4. Model

```
Model <- function(parm, Data)
    {
    ### Hyperhyperparameters
    delta.mu <- logit(1/Data$J)
    ### Hyperparameters
    beta.mu <- rep(0, Data$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)]
### Parameters
beta <- parm[1:Data$J]
gamma <- round(invlogit(delta))
beta.tau <- ifelse(gamma == 0, 100, beta.tau)
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
### Log(Hyperprior and 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, sqrt(1000), log=TRUE))
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
### Log-Likelihood
mu <- beta %*% t(Data$X)
LL <- sum(dnorm(y, mu, sigma, log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + beta.sigma.prior + delta.prior + sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, min(beta.tau),
    sigma, gamma), yhat=mu, parm=parm)
return(Modelout)
}
```

30. Zero-Inflated Poisson (ZIP)

30.1. Form

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

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

$$z_i = \begin{cases} 1 & \text{if } y_i = 0 \\ 0 & \end{cases}$$

$$\Lambda_{i,2} = \begin{cases} 0 & \text{if } \Lambda_{i,1} \geq 0.5 \\ \Lambda_{i,2} & \end{cases}$$

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

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

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

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

30.2. 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)
```

30.3. Initial Values

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

30.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1:Data$J1]
    beta <- parm[min(grep("beta", Data$parm.names)):max(grep(
        "beta", Data$parm.names))]
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
    ### Log-Likelihood
    Lambda <- matrix(NA, Data$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.

Congdon P (2003). *Applied Bayesian Modelling*. John Wiley & Sons, West Sussex, England.

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 11.06.27, 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.

Spiegelhalter D, Thomas A, Best N, Lunn D (2003). *WinBUGS User Manual, Version 1.4*. MRC Biostatistics Unit, Institute of Public Health and Department of Epidemiology and Public Health, Imperial College School of Medicine, UK. http://www.mrc-bsu.cam.ac.uk/bugs.

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

Affiliation:

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