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. It is also a testament to the diverse applicability of **LaplacesDemon** to Bayesian inference.

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. The provided data, initial values, and model specification may be copy/pasted into an R file and updated with the `LaplacesDemon` or (usually) `LaplaceApproximation` functions. Although many of these examples update quickly, some examples are computationally intensive.

Notation in this vignette follows these standards: Greek letters represent parameters, lower case letters represent indices, lower case bold face letters represent scalars or vectors, probability distributions are represented with calligraphic font, upper case letters represent index limits, and upper case bold face letters represent matrices.

This vignette will grow over time as examples of more methods become included. Contributed examples are welcome. Please send contributed examples or discovered errors in a similar format in an email to `laplacesdemon@statisticat.com` for review and testing. All accepted contributions are, of course, credited.

Contents

1. ANCOVA

This example is essentially the same as the two-way ANOVA (see section 3), except that a covariate $\mathbf{X}_{,3}$ has been added, and its parameter is δ.

1.1. Form

$$\mathbf{y}_i \sim \mathcal{N}(\mu_i, \sigma_1^2)$$
$$\mu_i = \alpha + \beta[\mathbf{X}_{i,1}] + \gamma[\mathbf{X}_{i,2}] + \delta \mathbf{X}_{i,2}, \quad i = 1, \dots, N$$
$$\epsilon_i = \mathbf{y}_i - \mu_i$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_j \sim \mathcal{N}(0, \sigma_2^2), \quad j = 1, \dots, (J-1)$$
$$\beta_J = -\sum_{j=1}^{J-1} \beta_j$$
$$\gamma_k \sim \mathcal{N}(0, \sigma_3^2), \quad k = 1, \dots, (K-1)$$
$$\gamma_K = -\sum_{k=1}^{K-1} \gamma_k$$
$$\delta \sim \mathcal{N}(0, 1000)$$
$$\sigma_m \sim \mathcal{HC}(25), \quad m = 1, \dots, 3$$

1.2. 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)),
     runif(N,-2,2)), N, 3)
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)])
delta <- runif(1,-2,2)
y <- alpha + beta[X[,1]] + gamma[X[,2]] + delta*X[,3] + rnorm(N,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),
     delta=0, 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)
```

1.3. Initial Values

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

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
     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
     delta <- parm[grep("delta", Data$parm.names)]
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     beta.prior <- sum(dnorm(beta, 0, sigma[2], log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sigma[3], log=TRUE))
     delta.prior <- dnorm(delta, 0, sqrt(1000), log=TRUE)
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
     mu <- alpha + beta[Data$X[,1]] + gamma[Data$X[,2]] +
          delta*Data$X[,3]
     LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
     ### Variance Components
     s.beta <- sd(beta)
     s.gamma <- sd(gamma)
     s.epsilon <- sd(Data$y - mu)
     ### Log-Posterior
     LP <- LL + alpha.prior + beta.prior + gamma.prior + delta.prior +
          sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, beta[Data$J],
          gamma[Data$K], sigma, s.beta, s.gamma, s.epsilon), yhat=mu,
          parm=parm)
     return(Modelout)
```

```
    }
```

2. ANOVA, One-Way

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

2.1. Form

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

2.2. 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)
```

2.3. Initial Values

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

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

3. 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.epsilon` is the finite-population between-variance component.

3.1. Form

$$\mathbf{y}_i \sim \mathcal{N}(\mu_i, \sigma_1^2)$$

$$\mu_i = \alpha + \beta[\mathbf{X}_{i,1}] + \gamma[\mathbf{X}_{i,2}], \quad i = 1, \dots, N$$

$$\epsilon_i = \mathbf{y}_i - \mu_i$$

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

$$\beta_j \sim \mathcal{N}(0, \sigma_2^2), \quad j = 1, \dots, (J-1)$$

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

$$\gamma_k \sim \mathcal{N}(0, \sigma_3^2), \quad k = 1, \dots, (K-1)$$

$$\gamma_K = -\sum_{k=1}^{K-1} \gamma_k$$

$$\sigma_m \sim \mathcal{HC}(25), \quad m = 1, \dots, 3$$

3.2. 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 <- alpha + beta[X[,1]] + gamma[X[,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)
```

3.3. Initial Values

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

3.4. 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$parm.names)])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     beta.prior <- sum(dnorm(beta, 0, sigma[2], log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sigma[3], log=TRUE))
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
     mu <- alpha + beta[Data$X[,1]] + gamma[Data$X[,2]]
     LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
     ### Variance Components
     s.beta <- sd(beta)
```

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

4. ARCH-M(1,1)

4.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2_t), \quad t = 1, \dots, T$$
$$\mathbf{y}^{new} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{new})$$
$$\mu_t = \alpha + \phi \mathbf{y}_{t-1} + \delta \sigma^2_{t-1}, \quad t = 1, \dots, (T+1)$$
$$\epsilon_t = \mathbf{y}_t - \mu_t$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\phi \sim \mathcal{N}(0, 1000)$$
$$\delta \sim \mathcal{N}(0, 1000)$$
$$\sigma^2_{new} = \theta_1 + \theta_2 \epsilon^2_T$$
$$\sigma^2_t = \theta_1 + \theta_2 \epsilon^2_{t-1}$$
$$\theta_k = \frac{1}{1 + \exp(-\theta_k)}, \quad k = 1, \dots, 2$$
$$\theta_k \sim \mathcal{N}(0, 1000) \in [-10, 10], \quad k = 1, \dots, 2$$

4.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
```

```
    -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
    0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
    -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
    -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
    -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
    0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
    -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
    0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
    -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
    -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
    0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
    -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
    0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
    0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
    0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
    -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
    0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
    -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
    0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "ynew", "sigma2.new")
parm.names <- c("alpha","phi","delta","logit.theta[1]","logit.theta[2]")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

4.3. Initial Values

```
Initial.Values <- c(rep(0,3), rep(0.5,2))
```

4.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1]; phi <- parm[2]; delta <- parm[3]
    theta <- invlogit(interval(parm[grep("logit.theta",
        Data$parm.names)], -10, 10))
    parm[grep("logit.theta", Data$parm.names)] <- logit(theta)
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
    delta.prior <- dnorm(delta, 0, sqrt(1000), log=TRUE)
    theta.prior <- sum(dnorm(theta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    mu <- c(alpha, alpha + phi*Data$y[-Data$T])
    epsilon <- Data$y - mu
    sigma2 <- c(theta[1], theta[1] + theta[2]*epsilon[-Data$T]^2)
```

```
mu <- mu + delta*sigma2
ynew <- alpha + phi*Data$y[Data$T] + delta*sigma2[Data$T]
sigma2.new <- theta[1] + theta[2]*epsilon[Data$T]^2
LL <- sum(dnorm(Data$y, mu, sqrt(sigma2), log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + phi.prior + delta.prior + theta.prior +
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, ynew, sigma2.new),
     yhat=mu, parm=parm)
return(Modelout)
}
```

5. Autoregression, AR(1)

5.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2), \quad t = 1, \ldots, T$$

$$\mathbf{y}^{new} = \alpha + \mu_{T+1}$$

$$\mu_t = \alpha + \phi \mathbf{y}_{t-1}, \quad t = 1, \ldots, (T+1)$$

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

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

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

5.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
```

```
    -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
    -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
    0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
    -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
    0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
    0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
    0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
    -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
    0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
    -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
    0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "sigma", "ynew")
parm.names <- c("alpha","phi","log.sigma")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

5.3. Initial Values

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

5.4. 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 <- c(alpha, alpha + phi*Data$y[-Data$T])
    ynew <- alpha + phi*Data$y[Data$T]
    LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + phi.prior + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma,ynew),
        yhat=mu, parm=parm)
    return(Modelout)
    }
```

6. Autoregressive Conditional Heteroskedasticity, ARCH(1,1)

6.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2_t), \quad t = 1, \ldots, T$$
$$\mathbf{y}^{new} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{new})$$
$$\mu_t = \alpha + \phi \mathbf{y}_{t-1}, \quad t = 1, \ldots, (T+1)$$
$$\epsilon_t = \mathbf{y}_t - \mu_t$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\phi \sim \mathcal{N}(0, 1000)$$
$$\sigma^2_{new} = \theta_1 + \theta_2 \epsilon^2_T$$
$$\sigma^2_t = \theta_1 + \theta_2 \epsilon^2_{t-1},$$
$$\theta_1 \sim \mathcal{N}(0, 1000) \in [0, \infty]$$
$$\theta_2 \sim \mathcal{U}(1.0E-100, 1)$$

6.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
     -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
     -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
     0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
     -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
     0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
     0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
     0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
     -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
     0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
     -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
```

```
    0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "ynew", "sigma2.new")
parm.names <- c("alpha","phi","logit.theta[1]","logit.theta[2]")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

6.3. Initial Values

```
Initial.Values <- c(rep(0,2), rep(0.5,2))
```

6.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1]; phi <- parm[2]
    theta <- invlogit(interval(parm[grep("logit.theta",
        Data$parm.names)], -10, 10))
    parm[grep("logit.theta", Data$parm.names)] <- logit(theta)
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
    theta.prior <- sum(dnorm(theta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    mu <- c(alpha, alpha + phi*Data$y[-Data$T])
    ynew <- alpha + phi*Data$y[Data$T]
    epsilon <- Data$y - mu
    sigma2 <- c(theta[1], theta[1] + theta[2]*epsilon[-Data$T]^2)
    sigma2.new <- theta[1] + theta[2]*epsilon[Data$T]^2
    LL <- sum(dnorm(Data$y, mu, sqrt(sigma2), log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + phi.prior + theta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, ynew,
        sigma2.new), yhat=mu, parm=parm)
    return(Modelout)
    }
```

7. Autoregressive Moving Average, ARMA(1,1)

7.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2), \quad t = 1, \dots, T$$
$$\mathbf{y}^{new} = \alpha + \phi \mathbf{y}_T + \theta \epsilon_T$$

$$\mu_t = \alpha + \phi \mathbf{y}_{t-1} + \theta \epsilon_{t-1}$$
$$\epsilon_t = \mathbf{y}_t - \mu_t$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\phi \sim \mathcal{N}(0, 1000)$$
$$\sigma \sim \mathcal{HC}(25)$$
$$\theta \sim \mathcal{N}(0, 1000)$$

7.2. Data

7.3. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
     -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
     -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
     0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
     -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
     0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
     0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
     0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
     -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
     0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
     -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
     0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "sigma", "ynew")
parm.names <- c("alpha","phi","sigma","theta")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

7.4. Initial Values

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

7.5. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]; phi <- parm[2]; theta <- parm[3]
     sigma <- exp(parm[4])
     ### 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)
     theta.prior <- dnorm(theta, 0, sqrt(1000), log=TRUE)
     ### Log-Likelihood
     mu <- c(alpha, alpha + phi*Data$y[-Data$T])
     epsilon <- Data$y - mu
     mu <- c(mu[1], mu[-1] + theta * epsilon[-Data$T])
     ynew <- alpha + phi*Data$y[Data$T] + theta*epsilon[Data$T]
     LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + phi.prior + sigma.prior + theta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, sigma, ynew),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

8. Beta Regression

8.1. Form

$$\mathbf{y} \sim \mathcal{BETA}(a, b)$$

$$a = \mu\phi$$

$$b = (1 - \mu)\phi$$

$$\mu = \Phi(\beta_1 + \beta_2 \mathbf{x})$$

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

$$\phi \sim \mathcal{G}(1, 1)$$

where Φ is the normal CDF.

8.2. Data

```
N <- 10
x <- runif(N)
y <- qbeta(0.5, pnorm(2-3*x)*4, (1-pnorm(2-3*x))*4)
mon.names <- "LP"
parm.names <- c("beta[1]","beta[2]","log.phi")
MyData <- list(x=x, y=y, mon.names=mon.names, parm.names=parm.names)
```

8.3. Initial Values

```
Initial.Values <- c(rep(0,2), log(0.01))
```

8.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:2]; phi <- exp(parm[3])
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(10), log=TRUE))
     phi.prior <- dgamma(phi, 1, 1, log=TRUE)
     ### Log-Likelihood
     mu <- pnorm(beta[1] + beta[2]*Data$x)
     a <- mu * phi
     b <- (1-mu) * phi
     LL <- sum(dbeta(Data$y, a, b, log=TRUE))
     ### Log-Posterior
     LP <- LL + beta.prior + phi.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=mu, parm=parm)
     return(Modelout)
     }
```

9. Binary Logit

9.1. Form

$$\mathbf{y} \sim \mathcal{BERN}(\eta)$$

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

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

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

9.2. 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)
```

9.3. Initial Values

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

9.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     mu <- tcrossprod(beta, Data$X)
     eta <- invlogit(mu)
     ### Log-Likelihood
     LL <- sum(dbern(Data$y, eta, log=TRUE))
     yrep <- ifelse(eta >= (sum(Data$y)/length(Data$y)),1,0)
     ### Log-Posterior
     LP <- LL + beta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
          yhat=yrep, parm=parm)
     return(Modelout)
     }
```

10. Binary Probit

10.1. Form

$$\mathbf{y} \sim \mathcal{BERN}(\mathbf{p})$$

$$\mathbf{p} = \phi(\mu)$$

$$\mu = \mathbf{X}\beta \in [-10, 10]$$

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

where ϕ is the inverse CDF, and J=3.

10.2. 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)
```

10.3. Initial Values

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

10.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- tcrossprod(beta, Data$X)
     mu <- interval(mu, -10, 10)
     p <- pnorm(mu)
     LL <- sum(dbern(Data$y, p, log=TRUE))
     yrep <- ifelse(p >= (sum(Data$y)/length(Data$y)),1,0)
     ### Log-Posterior
     LP <- LL + beta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
     return(Modelout)
     }
```

11. Binomial Logit

11.1. Form

$$\mathbf{y} \sim \mathcal{BIN}(\mathbf{p}, \mathbf{n})$$

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

$$\mu = \beta_1 + \beta_2 \mathbf{x}$$

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

11.2. 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)
```

11.3. Initial Values

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

11.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- beta[1] + beta[2]*Data$x
     p <- invlogit(mu)
     LL <- sum(dbinom(Data$y, Data$n, p, log=TRUE))
     yrep <- p * Data$n
     ### Log-Posterior
     LP <- LL + beta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
     return(Modelout)
     }
```

12. Binomial Probit

12.1. Form

$$\mathbf{y} \sim \mathcal{BIN}(\mathbf{p}, \mathbf{n})$$

$$\mathbf{p} = \phi(\mu)$$

$$\mu = \beta_1 + \beta_2\mathbf{x} \in [-10, 10]$$

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

where ϕ is the inverse CDF, and J=2.

12.2. 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)
```

12.3. Initial Values

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

12.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    mu <- beta[1] + beta[2]*Data$x
    mu <- interval(mu, -10, 10)
    p <- pnorm(mu)
    LL <- sum(dbinom(Data$y, Data$n, p, log=TRUE))
    yrep <- p * Data$n
    ### Log-Posterior
    LP <- LL + beta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep,
        parm=parm)
    return(Modelout)
    }
```

13. Cluster Analysis

This is a parametric model-based cluster analysis, also called a finite mixture model or latent class cluster analysis.

13.1. Form

$$\mathbf{Y}_{i,j} \sim \mathcal{N}(\mu_{\theta[i],j}, \sigma^2_{\theta[i]}), \quad i = 1, \dots, N, \quad j = 1, \dots, J$$
$$\theta_i = \mathrm{Max}(\mathbf{p}_{i,1:C})$$
$$\mathbf{p}_{i,c} = \frac{\delta_{i,c}}{\sum_{c=1}^{C} \delta_{i,c}}$$
$$\pi_{1:C} \sim \mathcal{D}(\alpha_{1:C})$$
$$\pi_c = \frac{\sum_{i=1}^{N} \delta_{i,c}}{\sum \delta}$$
$$\alpha_c = 1$$
$$\delta_{i,C} = 1$$
$$\delta_{i,c} \sim \mathcal{N}(\log(\frac{1}{C}), 1000) \in [\exp(-10), \exp(10)], \quad c = 1, \dots, (C-1)$$
$$\mu_{c,j} \sim \mathcal{N}(0, \nu_j^2)$$
$$\sigma_c \sim \mathcal{HC}(25)$$
$$\nu_j \sim \mathcal{HC}(25)$$

13.2. Data

```
C <- 3 #Number of clusters
alpha <- rep(1,C) #Prior probability of cluster proportion
# Create a Y matrix
n <- 100; N <- 15 #Full sample; model sample
J <- 5 #Number of predictor variables
cluster <- round(runif(n,0.5,C+0.49))
centers <- matrix(runif(C*J, 0, 10), C, J)
Y.Full <- matrix(0, n, J)
for (i in 1:n) {for (j in 1:J)
     {Y.Full[i,j] <- rnorm(1,centers[cluster[i],j],1)}}
mean.temp <- colMeans(Y.Full)
sigma.temp <- apply(Y.Full,2,sd)
centers.cs <- (centers - matrix(rep(mean.temp,C), C, J, byrow=TRUE)) /
     (2 * matrix(rep(sigma.temp,C), C, J, byrow=TRUE))
for (j in 1:J) {Y.Full[,j] <- scale(Y.Full[,j],2)}
#summary(Y.Full)
MySample <- sample(1:n, N)
Y <- Y.Full[MySample,]
mon.names <- c("LP", parm.names(list(nu=rep(0,J), pi=rep(0,C),
```

```
    sigma=rep(0,C), theta=rep(0,N))))
parm.names <- parm.names(list(log.delta=matrix(0,N,C-1), mu=matrix(0,C,J),
    log.nu=rep(0,J), log.sigma=rep(0,C)))
MyData <- list(C=C, J=J, N=N, Y=Y, alpha=alpha, mon.names=mon.names,
    parm.names=parm.names)
```

13.3. Initial Values

```
Initial.Values <- c(runif(N*(C-1),-1,1), rep(0,C*J), rep(0,J), rep(0,C))
```

13.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    delta <- interval(parm[grep("log.delta", Data$parm.names)], -10, 10)
    parm[grep("log.delta", Data$parm.names)] <- delta
    delta <- matrix(c(exp(delta), rep(1, Data$N)), Data$N, Data$C)
    mu <- matrix(parm[grep("mu", Data$parm.names)], Data$C, Data$J)
    nu <- exp(parm[grep("log.nu",Data$parm.names)])
    pi <- colSums(delta) / sum(delta)
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
    delta.prior <- sum(dtrunc(delta, "norm", a=exp(-10), b=exp(10),
        mean=log(1/Data$C), sd=sqrt(1000), log=TRUE))
    mu.prior <- sum(dnorm(mu, 0, matrix(rep(nu,Data$C), Data$C,
        Data$J, byrow=TRUE), log=TRUE))
    nu.prior <- sum(dhalfcauchy(nu, 25, log=TRUE))
    pi.prior <- ddirichlet(pi, Data$alpha, log=TRUE)
    sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
    ### Log-Likelihood
    p <- delta / rowSums(delta)
    theta <- apply(p,1,which.max)
    LL <- sum(dnorm(Data$Y, mu[theta,], sigma[theta], log=TRUE))
    Yrep <- mu[theta,]
    ### Log-Posterior
    LP <- LL + delta.prior + mu.prior + nu.prior + pi.prior +
        sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,nu,pi,sigma,theta),
        yhat=Yrep, parm=parm)
    return(Modelout)
    }
```

14. Conditional Autoregression (CAR), Poisson

This CAR example is 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 $\mathbf{y}$ consists of counts for $i = 1, \ldots, 56$ counties in Scotland. A single predictor $\mathbf{x}$ is provided. The errors, ϵ, are allowed to include spatial effects as smoothing by spatial effects from areal neighbors. Interactions $\mathbf{w}$ 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.

14.1. Form

$$\mathbf{y} \sim \mathcal{P}(\lambda)$$

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

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

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

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

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

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

14.2. 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)
```

14.3. Initial Values

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

14.4. 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)
     }
```

15. 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).

15.1. Form

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

15.2. 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)
```

15.3. Initial Values

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

15.4. 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[grep("beta", Data$parm.names)]
    gamma <- parm[grep("gamma", Data$parm.names)]
    delta <- matrix(parm[grep("delta", Data$parm.names)],
        Data$J, Data$K)
    ### Log(Prior Densities)
    alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
    beta.prior <- sum(dnorm(beta, 0, beta.sigma, log=TRUE))
    beta.sigma.prior <- dhalfcauchy(beta.sigma, 25, log=TRUE)
    gamma.prior <- sum(dnorm(gamma, 0, gamma.sigma, log=TRUE))
    gamma.sigma.prior <- dhalfcauchy(gamma.sigma, 25, log=TRUE)
    delta.prior <- sum(dnorm(delta, 0, delta.sigma, log=TRUE))
    delta.sigma.prior <- dhalfcauchy(delta.sigma, 25, log=TRUE)
    ### Log-Likelihood
    lambda <- exp(alpha + beta[Data$r] + gamma[Data$c] +
        diag(delta[Data$r,Data$c]))
    LL <- sum(dpois(Data$y, lambda, log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + beta.prior + beta.sigma.prior +
        gamma.prior + gamma.sigma.prior + 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)
    }
```

16. Discrete Choice, Conditional Logit

16.1. Form

$$\mathbf{y}_i \sim \mathcal{CAT}(\mathbf{p}_{i,1:J}), \quad i = 1, \ldots, N, \quad j = 1, \ldots, J$$
$$\mathbf{p}_{i,j} = \frac{\phi_{i,j}}{\sum_{j=1}^{J} \phi_{i,j}}$$
$$\phi = \exp(\mu)$$
$$\mu_{i,j} = \beta_{j,1:K}\mathbf{X}_{i,1:K} + \gamma\mathbf{Z}_{i,1:C} \in [-700, 700], \quad j = 1, \ldots, (J-1)$$
$$\mu_{i,J} = \gamma\mathbf{Z}_{i,1:C}$$
$$\beta_{j,k} \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, (J-1)$$
$$\gamma_c \sim \mathcal{N}(0, 1000)$$

16.2. Data

```
y <- x01 <- x02 <- z01 <- z02 <- 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)
z01[1:100] <- 1
z01[101:200] <- 2
z01[201:300] <- 3
z02[1:100] <- 40
z02[101:200] <- 50
z02[201:300] <- 100
N <- length(y)
J <- 3 #Number of categories in y
K <- 3 #Number of individual attributes (including the intercept)
C <- 2 #Number of choice-based attributes (intercept is not included)
X <- matrix(c(rep(1,N),x01,x02),N,K) #Design matrix of individual attrib.
Z <- matrix(c(z01,z02),N,C) #Design matrix of choice-based attributes
mon.names <- "LP"
parm.names <- parm.names(list(beta=matrix(0,J-1,K), gamma=rep(0,C)))
MyData <- list(C=C, J=J, K=K, N=N, X=X, Z=Z, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

16.3. Initial Values

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

16.4. Model

```
Model <- function(parm, Data)
     {     ### Parameters
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
     gamma <- parm[grep("gamma", Data$parm.names)]
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- matrix(rep(tcrossprod(gamma, Data$Z),J),Data$N,Data$J)
     mu[,1] <- mu[,1] + tcrossprod(beta[1,], Data$X)
     mu[,2] <- mu[,2] + tcrossprod(beta[2,], Data$X)
     mu <- interval(mu, -700, 700)
     phi <- exp(mu)
     p <- phi / rowSums(phi)
     LL <- sum(dcat(Data$y, p, log=TRUE))
     yrep <- apply(p,1,which.max)
     ### Log-Posterior
     LP <- LL + beta.prior + gamma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
     return(Modelout)
     }
```

17. Discrete Choice, Mixed Logit

17.1. Form

$$\mathbf{y}_i \sim \mathcal{CAT}(\mathbf{p}_{i,1:J}), \quad i = 1, \dots, N, \quad j = 1, \dots, J$$

$$\mathbf{p}_{i,j} = \frac{\phi_{i,j}}{\sum_{j=1}^{J} \phi_{i,j}}$$

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

$$\mu_{i,j} = \beta_{j,1:K}\mathbf{X}_{i,1:K} + \gamma\mathbf{Z}_{i,1:C} \in [-700, 700], \quad j = 1, \dots, (J-1)$$

$$\mu_{i,J} = \gamma\mathbf{Z}_{i,1:C}$$

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

$$\gamma_c \sim \mathcal{N}(\zeta_{\mu[c]}, \zeta^2_{\sigma[c]})$$

$$\zeta_{\mu[c]} \sim \mathcal{N}(0, 1000)$$

$$\zeta_{\sigma[c]} \sim \mathcal{HC}(25)$$

17.2. Data

```
y <- x01 <- x02 <- z01 <- z02 <- 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)
z01[1:100] <- 1
z01[101:200] <- 2
z01[201:300] <- 3
z02[1:100] <- 40
z02[101:200] <- 50
z02[201:300] <- 100
N <- length(y)
J <- 3 #Number of categories in y
K <- 3 #Number of individual attributes (including the intercept)
C <- 2 #Number of choice-based attributes (intercept is not included)
X <- matrix(c(rep(1,N),x01,x02),N,K) #Design matrix of individual attrib.
Z <- matrix(c(z01,z02),N,C) #Design matrix of choice-based attributes
mon.names <- c("LP", parm.names(list(zeta.sigma=rep(0,C))))
parm.names <- parm.names(list(beta=matrix(0,J-1,K), gamma=rep(0,C),
     zeta.mu=rep(0,C), log.zeta.sigma=rep(0,C)))
MyData <- list(C=C, J=J, K=K, N=N, X=X, Z=Z, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

17.3. Initial Values

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

17.4. Model

```
Model <- function(parm, Data)
     {     ### Parameters
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
     gamma <- parm[grep("gamma", Data$parm.names)]
     zeta.mu <- parm[grep("zeta.mu", Data$parm.names)]
     zeta.sigma <- exp(parm[grep("log.zeta.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sqrt(1000), log=TRUE))
     zeta.mu.prior <- sum(dnorm(zeta.mu, 0, sqrt(1000), log=TRUE))
     zeta.sigma.prior <- sum(dhalfcauchy(zeta.sigma, 25, log=TRUE))
```

```
### Log-Likelihood
mu <- matrix(rep(rowSums(gamma * Data$Z),J),Data$N,Data$J)
mu[,1] <- mu[,1] + tcrossprod(beta[1,], Data$X)
mu[,2] <- mu[,2] + tcrossprod(beta[2,], Data$X)
mu <- interval(mu, -700, 700)
phi <- exp(mu)
p <- phi / rowSums(phi)
LL <- sum(dcat(Data$y, p, log=TRUE))
yrep <- apply(p,1,which.max)
### Log-Posterior
LP <- LL + beta.prior + gamma.prior + zeta.mu.prior + zeta.sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,zeta.sigma.prior),
     yhat=yrep, parm=parm)
return(Modelout)
}
```

18. Discrete Choice, Multinomial Probit

18.1. Form

$$\mathbf{Z}_{i,1:J} \sim \mathcal{N}_J(\mu_{i,1:J}, \Sigma), \quad i = 1, \ldots, N$$

$$\mathbf{Z}_{i,j} \in \begin{cases} [0,10] & \text{if } \mathbf{y}_i = j \\ [-10,0] & \end{cases}$$

$$\mu_{1:N,j} = \mathbf{X}\beta_{j,1:K} + \mathbf{W}\gamma[a, 1:C]$$

$$\mathbf{a} = \begin{cases} 1 & \text{if } \mathbf{y}_i < J \\ 2 & \end{cases}$$

$$\Sigma \sim \mathcal{IW}(J, \mathbf{R}), \quad \mathbf{R} = \mathbf{I}_J, \quad \Sigma[1,1] = 1$$

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

$$\beta_{J,k} = -\sum_{j=1}^{J-1} \beta_{j,k}$$

$$\gamma_{1,1:C} \sim \mathcal{N}(0, 1000)$$

$$\gamma_{2,c} = -\gamma_{1,c}, \quad c = 1, \ldots, C$$

$$\mathbf{Z}_{i,j} \sim \mathcal{N}(0, 1000) \in [-10, 10]$$

18.2. Data

```
y <- x1 <- x2 <- w1 <- w2 <- c(1:30)
y[1:10] <- 1
y[11:20] <- 2
```

```
y[21:30] <- 3
x1[1:10] <- rnorm(10, 25, 2.5)
x1[11:20] <- rnorm(10, 40, 4.0)
x1[21:30] <- rnorm(10, 35, 3.5)
x2[1:10] <- rnorm(10, 2.51, 0.25)
x2[11:20] <- rnorm(10, 2.01, 0.20)
x2[21:30] <- rnorm(10, 2.70, 0.27)
w1[1:10] <- 10
w1[11:20] <- 4
w1[21:30] <- 1
w2[1:10] <- 40
w2[11:20] <- 50
w2[21:30] <- 100
N <- length(y)
J <- length(unique(y)) #Number of categories in y
K <- 3 #Number of columns to be in design matrix X
R <- diag(J)
X <- matrix(c(rep(1,N),x1,x2),N,K)
C <- 2 #Number of choice-based attributes
W <- matrix(c(w1,w2),N,C) #Design matrix of choice-based attributes
mon.names <- "LP"
sigma.temp <- parm.names(list(Sigma=diag(J)), uppertri=1)
parm.names <- c(sigma.temp[2:length(sigma.temp)],
     parm.names(list(beta=matrix(0,(J-1),K), gamma=rep(0,C),
     Z=matrix(0,N,J))))
MyData <- list(J=J, K=K, N=N, R=R, W=W, X=X, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

18.3. Initial Values

```
Initial.Values <- c(rep(0,length(R[upper.tri(R, diag=TRUE)])-1),
     rep(0,(J-1)*K), rep(0,C), rep(0,N,J))
```

18.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
     beta <- rbind(beta, colSums(beta)*-1) #Sum to zero constraint
     gamma <- parm[grep("gamma", Data$parm.names)]
     gamma <- rbind(gamma, gamma*-1) #Sum to zero constraint
     Sigma <- matrix(NA, Data$J, Data$J)
     Sigma[upper.tri(Sigma, diag=TRUE)] <- c(0, parm[grep("Sigma",
          Data$parm.names)])
     Sigma[lower.tri(Sigma)] <- Sigma[upper.tri(Sigma)]
```

```
diag(Sigma) <- exp(diag(Sigma))
Z <- matrix(parm[grep("Z", Data$parm.names)], Data$N, Data$J)
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
gamma.prior <- sum(dnorm(gamma, 0, sqrt(1000), log=TRUE))
Sigma.prior <- dinvwishart(Sigma, Data$J, Data$R, log=TRUE)
Z.prior <- sum(dnorm(Z, 0, sqrt(1000), log=TRUE))
### Log-Likelihood
mu <- matrix(0,Data$N,Data$J)
mu <- matrix(c(rep(tcrossprod(gamma[1,], Data$W),J),
     tcrossprod(gamma[2,], Data$W)),Data$N,Data$J)
for (j in 1:Data$J) {mu[,j] <- mu[,j] + tcrossprod(beta[j,], Data$X)}
Y <- indmat(Data$y)
Z <- ifelse(Z > 10, 10, Z); Z <- ifelse({Y == 0} & {Z > 0}, 0, Z)
Z <- ifelse(Z < -10, -10, Z); Z <- ifelse({Y == 1} & {Z < 0}, 0, Z)
parm[grep("Z", Data$parm.names)] <- as.vector(Z)
LL <- sum(dmvn(Z, mu, Sigma, log=TRUE))
yrep <- apply(Z, 1, which.max)
#eta <- exp(mu)
#p <- eta / rowSums(eta)
### Log-Posterior
LP <- LL + beta.prior + gamma.prior + Sigma.prior + Z.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
return(Modelout)
}
```

19. Distributed Lag, Koyck

This example applies Koyck or geometric distributed lags to $k = 1, \dots, K$ discrete events in covariate $\mathbf{x}$, transforming the covariate into a N x K matrix $\mathbf{X}$ and creates a N x K lag matrix $\mathbf{L}$.

19.1. Form

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

$$\mu_t = \alpha + \phi \mathbf{y}_{t-1} + \sum_{k=1}^{K} \mathbf{X}_{t,k} \beta \lambda^{\mathbf{L}[t,k]}, \quad k = 1, \dots, K, \quad t = 2, \dots, T$$

$$\mu_1 = \alpha + \sum_{k=1}^{K} \mathbf{X}_{1,k} \beta \lambda^{\mathbf{L}[1,k]}, \quad k = 1, \dots, K$$

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

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

$$\lambda \sim \mathcal{U}(0, 1)$$
$$\phi \sim \mathcal{N}(0, 1000)$$
$$\sigma \sim \mathcal{HC}(25)$$

19.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
     -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
     -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
     0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
     -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
     0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
     0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
     0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
     -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
     0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
     -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
     0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
x <- c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
     1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
T <- length(y)
```

```
K <- length(which(x != 0))
L <- X <- matrix(0, T, K)
for (i in 1:K) {
     X[which(x != 0)[i]:T,i] <- x[which(x != 0)[i]]
     L[(which(x != 0)[i]):T,i] <- 0:(T - which(x != 0)[i])}
mon.names <- "LP"
parm.names <- c("alpha","beta","lambda","phi","log.sigma")
MyData <- list(L=L, T=T, X=X, mon.names=mon.names, parm.names=parm.names,
     y=y)
```

19.3. Initial Values

```
Initial.Values <- c(rep(0,2), 0.5, 0, log(1))
```

19.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]; beta <- parm[2]
     lambda <- interval(parm[3],0,1); parm[3] <- lambda
     phi <- parm[4]; sigma <- exp(parm[5])
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     beta.prior <- dnorm(beta, 0, sqrt(1000), log=TRUE)
     lambda.prior <- dunif(lambda, 0, 1, log=TRUE)
     phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
     sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
     ### Log-Likelihood
     mu <- c(alpha, alpha + phi*Data$y[-Data$T]) +
          rowSums(Data$X * beta * lambda^Data$L)
     LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + beta.prior + lambda.prior + phi.prior +
          sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=mu, parm=parm)
     return(Modelout)
     }
```

20. Dynamic Linear Model (DLM)

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

20.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2_V), \quad t = 1, \dots, T_m$$
$$\mathbf{y}^{new}_t \sim \mathcal{N}(\mu_t, \sigma^2_V), \quad t = (T_m + 1), \dots, T$$
$$\mu_t = \alpha + \mathbf{x}_t \beta_t, \quad t = 1, \dots, T$$
$$\alpha \sim \mathcal{N}(0, 1000)$$
$$\beta_1 \sim \mathcal{N}(0, 1000)$$
$$\beta_t \sim \mathcal{N}(\beta_{t-1}, \sigma^2_W), \quad t = 2, \dots, T$$
$$\sigma_V \sim \mathcal{HC}(25)$$
$$\sigma_W \sim \mathcal{HC}(25)$$

20.2. 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)
```

20.3. Initial Values

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

20.4. 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)
}
```

21. Exponential Smoothing

21.1. Form

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

$$\mu_t = \alpha \mathbf{y}_{t-1} + (1-\alpha)\mu_{t-1}, \quad t = 2, \dots, T$$

$$\alpha \sim \mathcal{U}(0,1)$$

$$\sigma \sim \mathcal{HC}$$

21.2. Data

```
T <- 10
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")
parm.names <- c("alpha","log.sigma")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

21.3. Initial Values

```
Initial.Values <- c(0.5, log(1))
```

21.4. Model

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

22. Factor Analysis, Confirmatory

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

22.1. Form

$$\mathbf{Y}_{i,m} \sim \mathcal{N}(\mu_{i,m}, \sigma^2_m), \quad i = 1, \dots, N, \quad m = 1, \dots, M$$
$$\mu_{i,m} = \alpha_m + \lambda_m \mathbf{F}_{i,\mathbf{f}[m]}, \quad i = 1, \dots, N, \quad m = 1, \dots, M$$
$$\mathbf{F}_{i,1:P} \sim \mathcal{N}_P(\gamma, \Omega^{-1}), \quad i = 1, \dots, N$$
$$\alpha_m \sim \mathcal{N}(0, 1000), \quad m = 1, \dots, M$$
$$\lambda_m \sim \mathcal{N}(0, 1000), \quad m = 1, \dots, M$$
$$\sigma_m \sim \mathcal{HC}(25), \quad m = 1, \dots, M$$
$$\Omega \sim \mathcal{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_P$$

22.2. 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)
```

22.3. Initial Values

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

22.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[grep("alpha", Data$parm.names)]
     lambda <- parm[grep("lambda", Data$parm.names)]
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     F <- matrix(parm[grep("F", Data$parm.names)], Data$N, Data$P)
     Omega <- matrix(NA, Data$P, Data$P)
     Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
          Data$parm.names)]
     Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
     Sigma <- solve(Omega)
     ### Log(Prior Densities)
     alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
     lambda.prior <- sum(dnorm(lambda, 0, sqrt(1000), log=TRUE))
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
     F.prior <- sum(dmvn(F, Data$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 <- LL + alpha.prior + lambda.prior + sigma.prior + F.prior +
          Omega.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,mu[1,1]),
          yhat=mu, parm=parm)
```

```
return(Modelout)
}
```

23. 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 $P = 3$.

23.1. Form

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

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

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

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

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

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

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

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

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

23.2. 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)
```

23.3. Initial Values

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

23.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[grep("alpha", Data$parm.names)]
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    F <- matrix(parm[grep("F", Data$parm.names)], Data$N, Data$P)
    Lambda <- matrix(parm[grep("Lambda", Data$parm.names)],
        Data$P, Data$M)
    Omega <- matrix(NA, Data$P, Data$P)
    Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
        Data$parm.names)]
    Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
    Sigma <- solve(Omega)
    ### Log(Prior Densities)
    alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
    sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
    Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
    F.prior <- sum(dmvn(F, Data$gamma, Sigma, log=TRUE))
    Lambda.prior <- sum(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] + rowSums(nu[,1,])}
    LL <- sum(dnorm(Data$Y, mu, sigma, log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + sigma.prior + Omega.prior + F.prior +
        Lambda.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,mu[1,1]),
        yhat=mu, parm=parm)
    return(Modelout)
    }
```

24. Factor Regression

This example of factor regression is constrained to the case where the number of factors is equal to the number of independent variables (IVs) less the intercept, or $J - 1$. The purpose of this form of factor regression is to orthogonalize the IVs with respect to $\mathbf{y}$, rather than variable reduction. This method is the combination of confirmatory factor analysis in section 22 and linear regression in section 31.

24.1. Form

$$\mathbf{y}_i \sim \mathcal{N}(\nu, \sigma_J^2)$$
$$\nu = \mu\beta$$
$$\mu_{i,1} = 1$$
$$\mu_{i,j+1} = \mu_{i,j}, \quad j = 1, \dots, (J-1)$$
$$\mathbf{X}_{i,j} \sim \mathcal{N}(\mu_{i,j}, \sigma_j^2), \quad i = 1, \dots, N, \quad j = 2, \dots, J$$
$$\mu_{i,j} = \alpha_j + \lambda_j \mathbf{F}_{i,j}, \quad i = 1, \dots, N, \quad j = 2, \dots, J$$
$$\mathbf{F}_{i,1:J} \sim \mathcal{N}_{J-1}(0, \Omega^{-1}), \quad i = 1, \dots, N$$
$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, (J-1)$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, J$$
$$\lambda_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, (J-1)$$
$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \dots, J$$
$$\Omega \sim \mathcal{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_{J-1}$$

24.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
J <- 4
y <- log(demonsnacks$Calories)
X <- cbind(1, as.matrix(demonsnacks[,c(7,8,10)]))
for (j in 2:J) {X[,j] <- CenterScale(X[,j])}
S <- diag((J-1))
mon.names <- "LP"
parm.names <- parm.names(list(alpha=rep(0,J-1), beta=rep(0,J),
     lambda=rep(0,J-1), log.sigma=rep(0,J), F=matrix(0,N,J-1),
     Omega=diag(J-1)), uppertri=c(0,0,0,0,0,1))
MyData <- list(J=J, N=N, S=S, X=X, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

24.3. Initial Values

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

24.4. Model

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

```
### Parameters
alpha <- parm[grep("alpha", Data$parm.names)]
beta <- parm[grep("beta", Data$parm.names)]
lambda <- parm[grep("lambda", Data$parm.names)]
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
F <- matrix(parm[grep("F", Data$parm.names)], Data$N, Data$J-1)
Omega <- matrix(NA, Data$J-1, Data$J-1)
Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
     Data$parm.names)]
Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
Sigma <- solve(Omega)
### Log(Prior Densities)
alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
lambda.prior <- sum(dnorm(lambda, 0, sqrt(1000), log=TRUE))
sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
Omega.prior <- dwishart(Omega, Data$N, Data$S, log=TRUE)
F.prior <- sum(dmvn(F, rep(0,Data$J-1), Sigma, log=TRUE))
### Log-Likelihood
mu <- matrix(alpha, Data$N, Data$J-1, byrow=TRUE) +
     matrix(lambda, Data$N, Data$J-1, byrow=TRUE) * F
nu <- tcrossprod(beta, cbind(rep(1,Data$N),mu))
LL <- sum(dnorm(Data$y, nu, sigma[Data$J], log=TRUE))
### Log-Posterior
LP <- LL + alpha.prior + beta.prior + lambda.prior + sigma.prior +
     F.prior + Omega.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=nu, parm=parm)
return(Modelout)
}
```

25. GARCH(1,1)

25.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2_t), \quad t = 1, \dots, T$$

$$\mathbf{y}^{new} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{new})$$

$$\mu_t = \alpha + \phi \mathbf{y}_{t-1}, \quad t = 1, \dots, (T+1)$$

$$\epsilon_t = \mathbf{y}_t - \mu_t$$

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

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

$$\sigma^2_{new} = \theta_1 + \theta_2 \epsilon^2_T + \theta_3 \sigma^2_T$$

$$\sigma_t^2 = \theta_1 + \theta_2 \epsilon_{t-1}^2 + \theta_3 \sigma_{t-1}^2$$

$$\theta_k = \frac{1}{1 + \exp(-\theta_k)}, \quad k = 1, \dots, 3$$

$$\theta_k \sim \mathcal{N}(0, 1000) \in [-10, 10], \quad k = 1, \dots, 3$$

25.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
     -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
     -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
     0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
     -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
     0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
     0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
     0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
     -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
     0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
     -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
     0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "ynew", "sigma2.new")
parm.names <- c("alpha","phi","logit.theta[1]","logit.theta[2]",
     "logit.theta[3]")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

25.3. Initial Values

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

25.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]; phi <- parm[2]
     theta <- invlogit(interval(parm[grep("logit.theta",
          Data$parm.names)], -10, 10))
     parm[grep("logit.theta", Data$parm.names)] <- logit(theta)
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
     theta.prior <- sum(dnorm(theta, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- c(alpha, alpha + phi*Data$y[-Data$T])
     ynew <- alpha + phi*Data$y[Data$T]
     epsilon <- Data$y - mu
     sigma2 <- c(theta[1], theta[1] + theta[2]*epsilon[-Data$T]^2)
     sigma2[-1] <- sigma2[-1] + theta[3]*sigma2[-Data$T]
     sigma2.new <- theta[1] + theta[2]*epsilon[Data$T]^2 +
          theta[3]*sigma2[Data$T]
     LL <- sum(dnorm(Data$y, mu, sqrt(sigma2), log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + phi.prior + theta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, ynew, sigma2.new),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

26. GARCH-M(1,1)

26.1. Form

$$\mathbf{y}_t \sim \mathcal{N}(\mu_t, \sigma^2_t), \quad t = 1, \dots, T$$

$$\mathbf{y}^{new} \sim \mathcal{N}(\mu_{T+1}, \sigma^2_{new})$$

$$\mu_t = \alpha + \phi \mathbf{y}_{t-1} + \delta \sigma^2_{t-1}, \quad t = 1, \dots, (T+1)$$

$$\epsilon_t = \mathbf{y}_t - \mu_t$$

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

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

$$\sigma^2_{new} = \theta_1 + \theta_2 \epsilon^2_T + \theta_3 \sigma^2_T$$

$$\sigma^2_t = \theta_1 + \theta_2 \epsilon^2_{t-1} + \theta_3 \sigma^2_{t-1}$$

$$\theta_k = \frac{1}{1 + \exp(-\theta_k)}, \quad k = 1, \ldots, 3$$

$$\theta_k \sim \mathcal{N}(0, 1000) \in [-10, 10], \quad k = 1, \ldots, 3$$

26.2. Data

```
y <- c(0.02, -0.51, -0.30, 1.46, -1.26, -2.15, -0.91, -0.53, -1.91,
     2.64, 1.64, 0.15, 1.46, 1.61, 1.96, -2.67, -0.19, -3.28,
     1.89, 0.91, -0.71, 0.74, -0.10, 3.20, -0.80, -5.25, 1.03,
     -0.40, -1.62, -0.80, 0.77, 0.17, -1.39, -1.28, 0.48, -1.02,
     0.09, -1.09, 0.86, 0.36, 1.51, -0.02, 0.47, 0.62, -1.36,
     1.12, 0.42, -4.39, -0.87, 0.05, -5.41, -7.38, -1.01, -1.70,
     0.64, 1.16, 0.87, 0.28, -1.69, -0.29, 0.13, -0.65, 0.83,
     0.62, 0.05, -0.14, 0.01, -0.36, -0.32, -0.80, -0.06, 0.24,
     0.23, -0.37, 0.00, -0.33, 0.21, -0.10, -0.10, -0.01, -0.40,
     -0.35, 0.48, -0.28, 0.08, 0.28, 0.23, 0.27, -0.35, -0.19,
     0.24, 0.17, -0.02, -0.23, 0.03, 0.02, -0.17, 0.04, -0.39,
     -0.12, 0.16, 0.17, 0.00, 0.18, 0.06, -0.36, 0.22, 0.14,
     -0.17, 0.10, -0.01, 0.00, -0.18, -0.02, 0.07, -0.06, 0.06,
     -0.05, -0.08, -0.07, 0.01, -0.06, 0.01, 0.01, -0.02, 0.01,
     0.01, 0.12, -0.03, 0.08, -0.10, 0.01, -0.03, -0.08, 0.04,
     -0.09, -0.08, 0.01, -0.05, 0.08, -0.14, 0.06, -0.11, 0.09,
     0.06, -0.12, -0.01, -0.05, -0.15, -0.05, -0.03, 0.04, 0.00,
     -0.12, 0.04, -0.06, -0.05, -0.07, -0.05, -0.14, -0.05, -0.01,
     -0.12, 0.05, 0.06, -0.10, 0.00, 0.01, 0.00, -0.08, 0.00,
     0.00, 0.07, -0.01, 0.00, 0.09, 0.33, 0.13, 0.42, 0.24,
     -0.36, 0.22, -0.09, -0.19, -0.10, -0.08, -0.07, 0.05, 0.07,
     0.07, 0.00, -0.04, -0.05, 0.03, 0.08, 0.26, 0.10, 0.08,
     0.09, -0.07, -0.33, 0.17, -0.03, 0.07, -0.04, -0.06, -0.06,
     0.07, -0.03, 0.00, 0.08, 0.27, 0.11, 0.11, 0.06, -0.11,
     -0.09, -0.21, 0.24, -0.12, 0.11, -0.02, -0.03, 0.02, -0.10,
     0.00, -0.04, 0.01, 0.02, -0.03, -0.10, -0.09, 0.17, 0.07,
     -0.05, -0.01, -0.05, 0.01, 0.00, -0.08, -0.05, -0.08, 0.07,
     0.06, -0.14, 0.02, 0.01, 0.04, 0.00, -0.13, -0.17)
T <- length(y)
mon.names <- c("LP", "ynew", "sigma2.new")
parm.names <- c("alpha","phi","delta","logit.theta[1]","logit.theta[2]",
     "logit.theta[3]")
MyData <- list(T=T, mon.names=mon.names, parm.names=parm.names, y=y)
```

26.3. Initial Values

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

26.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1]; phi <- parm[2]; delta <- parm[3]
     theta <- invlogit(interval(parm[grep("logit.theta",
          Data$parm.names)], -10, 10))
     parm[grep("logit.theta", Data$parm.names)] <- logit(theta)
     ### Log(Prior Densities)
     alpha.prior <- dnorm(alpha, 0, sqrt(1000), log=TRUE)
     phi.prior <- dnorm(phi, 0, sqrt(1000), log=TRUE)
     delta.prior <- dnorm(delta, 0, sqrt(1000), log=TRUE)
     theta.prior <- sum(dnorm(theta, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- c(alpha, alpha + phi*Data$y[-Data$T])
     epsilon <- Data$y - mu
     sigma2 <- c(theta[1], theta[1] + theta[2]*epsilon[-Data$T]^2)
     sigma2[-1] <- sigma2[-1] + theta[3]*sigma2[-Data$T]
     sigma2.new <- theta[1] + theta[2]*epsilon[Data$T]^2 +
          theta[3]*sigma2[Data$T]
     mu <- mu + delta*sigma2
     ynew <- alpha + phi*Data$y[Data$T] + delta*sigma2[Data$T]
     LL <- sum(dnorm(Data$y, mu, sqrt(sigma2), log=TRUE))
     ### Log-Posterior
     LP <- LL + alpha.prior + phi.prior + delta.prior + theta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, ynew, sigma2.new),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

27. Geographically Weighted Regression

27.1. Form

$$\mathbf{y}_{i,k} \sim \mathcal{N}(\mu_{i,k}, \tau_{i,k}^{-1}), \quad i = 1, \ldots, N, \quad k = 1, \ldots, N$$

$$\mu_{i,1:N} = \mathbf{X}\beta_{i,1:J}$$

$$\tau = \frac{1}{\sigma^2}\mathbf{w}\nu$$

$$\mathbf{w} = \frac{\exp(-0.5\mathbf{Z}^2)}{\mathbf{h}}$$

$$\alpha \sim \mathcal{U}(1.5, 100)$$

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

$$\mathbf{h} \sim \mathcal{N}(0.1, 1000) \in [0.1, \infty]$$
$$\nu_{i,k} \sim \mathcal{G}(\alpha, 2), \quad i = 1, \ldots, N, \quad k = 1, \ldots, N$$
$$\sigma_i \sim \mathcal{HC}(25), \quad i = 1, \ldots, N$$

27.2. Data

```
crime <- c(18.802, 32.388, 38.426, 0.178, 15.726, 30.627, 50.732,
     26.067, 48.585, 34.001, 36.869, 20.049, 19.146, 18.905, 27.823,
     16.241, 0.224, 30.516, 33.705, 40.970, 52.794, 41.968, 39.175,
     53.711, 25.962, 22.541, 26.645, 29.028, 36.664, 42.445, 56.920,
     61.299, 60.750, 68.892, 38.298, 54.839, 56.706, 62.275, 46.716,
     57.066, 54.522, 43.962, 40.074, 23.974, 17.677, 14.306, 19.101,
     16.531, 16.492)
income <- c(21.232, 4.477, 11.337, 8.438, 19.531, 15.956, 11.252,
     16.029, 9.873, 13.598, 9.798, 21.155, 18.942, 22.207, 18.950,
     29.833, 31.070, 17.586, 11.709, 8.085, 10.822, 9.918, 12.814,
     11.107, 16.961, 18.796, 11.813, 14.135, 13.380, 17.017, 7.856,
     8.461, 8.681, 13.906, 14.236, 7.625, 10.048, 7.467, 9.549,
     9.963, 11.618, 13.185, 10.655, 14.948, 16.940, 18.739, 18.477,
     18.324, 25.873)
housing <- c(44.567, 33.200, 37.125, 75.000, 80.467, 26.350, 23.225,
     28.750, 18.000, 96.400, 41.750, 47.733, 40.300, 42.100, 42.500,
     61.950, 81.267, 52.600, 30.450, 20.300, 34.100, 23.600, 27.000,
     22.700, 33.500, 35.800, 26.800, 27.733, 25.700, 43.300, 22.850,
     17.900, 32.500, 22.500, 53.200, 18.800, 19.900, 19.700, 41.700,
     42.900, 30.600, 60.000, 19.975, 28.450, 31.800, 36.300, 39.600,
     76.100, 44.333)
easting <- c(35.62, 36.50, 36.71, 33.36, 38.80, 39.82, 40.01, 43.75,
     39.61, 47.61, 48.58, 49.61, 50.11, 51.24, 50.89, 48.44, 46.73,
     43.44, 43.37, 41.13, 43.95, 44.10, 43.70, 41.04, 43.23, 42.67,
     41.21, 39.32, 41.09, 38.3, 41.31, 39.36, 39.72, 38.29, 36.60,
     37.60, 37.13, 37.85, 35.95, 35.72, 35.76, 36.15, 34.08, 30.32,
     27.94, 27.27, 24.25, 25.47, 29.02)
northing <- c(42.38, 40.52, 38.71, 38.41, 44.07, 41.18, 38.00, 39.28,
     34.91, 36.42, 34.46, 32.65, 29.91, 27.80, 25.24, 27.93, 31.91,
     35.92, 33.46, 33.14, 31.61, 30.40, 29.18, 28.78, 27.31, 24.96,
     25.90, 25.85, 27.49, 28.82, 30.90, 32.88, 30.64, 30.35, 32.09,
     34.08, 36.12, 36.30, 36.40, 35.60, 34.66, 33.92, 30.42, 28.26,
     29.85, 28.21, 26.69, 25.71, 26.58)
N <- length(crime)
J <- 3 #Number of predictors, including the intercept
X <- matrix(c(rep(1,N), income, housing),N,J)
D <- as.matrix(dist(cbind(northing,easting), diag=TRUE, upper=TRUE))
Z <- D / sd(as.vector(D))
y <- matrix(0,N,N); for (i in 1:N) {for (k in 1:N) {y[i,k] <- crime[k]}}
mon.names <- c("LP",parm.names(list(LAR2=rep(0,N))))
```

```
parm.names <- parm.names(list(alpha=0, beta=matrix(0,N,J), log.h=0,
     log.nu=matrix(0,N,N), log.sigma=rep(0,N)))
MyData <- list(J=J, N=N, X=X, Z=Z, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

27.3. Initial Values

```
Initial.Values <- c(runif(1,1.5,100), rep(0,N*J), log(1), rep(0,N*N),
     log(rep(1,N)))
```

27.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- interval(parm[grep("alpha", Data$parm.names)], 1.5, 100)
     parm[grep("alpha", Data$parm.names)] <- alpha
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$N, Data$J)
     h <- exp(parm[grep("log.h", Data$parm.names)]) + 0.1
     nu <- exp(matrix(parm[grep("log.nu", Data$parm.names)],
          Data$N, Data$N))
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     alpha.prior <- dunif(alpha, 1.5, 100, log=TRUE)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     h.prior <- dtrunc(h, "norm", a=0.1, b=Inf, mean=0.1, sd=sqrt(1000),
          log=TRUE)
     nu.prior <- sum(dgamma(nu, alpha, 2, log=TRUE))
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
     w <- exp(-0.5 * Z^2) / h
     tau <- (1/sigma^2) * w * nu
     mu <- matrix(NA, Data$N, Data$N)
     for (i in 1:N) {mu[i,] <- tcrossprod(beta[i,], Data$X)}
     LL <- sum(dnorm(Data$y, mu, sqrt(1/tau), log=TRUE))
     WSE <- w * nu * (Data$y - mu)^2; w.y <- w * nu * Data$y
     WMSE <- rowMeans(WSE); y.w <- rowSums(w.y) / rowSums(w)
     LAR2 <- 1 - WMSE / sd(y.w)^2
     ### Log-Posterior
     LP <- LL + alpha.prior + beta.prior + h.prior + nu.prior + sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,LAR2), yhat=mu,
          parm=parm)
     return(Modelout)
     }
```

28. Kriging

This is an example of universal kriging of $\mathbf{y}$ 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 $\mathbf{y}$. 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 $\rho = \exp(-\phi \mathbf{D})$. To extend this to a large data set, consider the predictive process kriging example in section 29.

28.1. Form

$$\mathbf{y} \sim \mathcal{N}(\mu, \sigma_1^2)$$
$$\mu = \mathbf{X}\beta + \zeta$$
$$\mathbf{y}^{new} = \mathbf{X}\beta + \sum_{i=1}^{N} \left(\frac{\rho_i}{\sum \rho} \zeta_i \right)$$
$$\rho = \exp(-\phi \mathbf{D}^{new})^\kappa$$
$$\zeta \sim \mathcal{N}_N(\zeta_\mu, \Sigma)$$
$$\Sigma = \sigma_2^2 \exp(-\phi \mathbf{D})^\kappa$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, 2$$
$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \ldots, 2$$
$$\phi \sim \mathcal{U}(1, 5)$$
$$\zeta_\mu = 0$$
$$\kappa = 1$$

28.2. 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(tcrossprod(beta, 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)
```

28.3. Initial Values

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

28.4. 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$Xnew) + sum(rho / sum(rho) * zeta)
     ### Log-Likelihood
     mu <- tcrossprod(beta, Data$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)
     }
```

29. Kriging, Predictive Process

The first K of N records in $\mathbf{y}$ are used as knots for the parent process, and the predictive process involves records $(K+1), \ldots, N$. For more information on kriging, see section 28.

29.1. Form

$$\mathbf{y} \sim \mathcal{N}(\mu, \sigma_1^2)$$
$$\mu_{1:K} = \mathbf{X}_{1:K,1:J}\beta + \zeta$$
$$\mu_{(K+1):N} = \mathbf{X}_{(K+1):N,1:J}\beta + \sum_{p=1}^{N-K} \frac{\lambda_{p,1:K}}{\sum_{q=1}^{N-K} \lambda_{q,1:K}} \zeta^T$$
$$\lambda = \exp(-\phi \mathbf{D}_P)^\kappa$$
$$\mathbf{y}^{new} = \mathbf{X}\beta + \sum_{k=1}^{K} (\frac{\rho_k}{\sum \rho} \zeta_k)$$
$$\rho = \exp(-\phi \mathbf{D}^{new})^\kappa$$
$$\zeta \sim \mathcal{N}_K(0, \Sigma)$$
$$\Sigma = \sigma_2^2 \exp(-\phi \mathbf{D})^\kappa$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, 2$$
$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \ldots, 2$$
$$\phi \sim \mathrm{N}(0, 1000) \in [0, \infty]$$
$$\kappa = 1$$

29.2. Data

```
N <- 100
K <- 30 #Number of knots
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(tcrossprod(beta, 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[1:K],latitude[1:K]), diag=TRUE,
     upper=TRUE))
D.P <- matrix(0, N-K, K)
for (i in (K+1):N) {
     D.P[K+1-i,] <- sqrt((longitude[1:K] - longitude[i])^2 +
          (latitude[1:K] - latitude[i])^2)}
```

```
D.new <- sqrt((longitude[1:K] - longitude.new)^2 +
     (latitude[1:K] - latitude.new)^2)
mon.names <- c("LP","sigma[1]","sigma[2]","ynew")
parm.names <- parm.names(list(zeta=rep(0,K), beta=rep(0,2),
     log.sigma=rep(0,2), log.phi=0))
MyData <- list(D=D, D.new=D.new, D.P=D.P, K=K, N=N, X=X, Xnew=Xnew,
     mon.names=mon.names, parm.names=parm.names, y=y)
```

29.3. Initial Values

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

29.4. 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$parm.names)])
     phi <- exp(parm[grep("log.phi", Data$parm.names)])
     Sigma <- sigma[2]*sigma[2] * exp(-phi * Data$D)^kappa
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     zeta.prior <- dmvn(zeta, rep(0, Data$K), 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 <- tcrossprod(beta, Data$X)
     mu[1:Data$K] <- mu[1:Data$K] + zeta
     lambda <- exp(-phi * Data$D.P)^kappa
     mu[(Data$K+1):Data$N] <- mu[(Data$K+1):Data$N] +
          rowSums(lambda / rowSums(lambda) *
          matrix(zeta, Data$N - Data$K, Data$K, byrow=TRUE))
     LL <- sum(dnorm(Data$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)
     }
```

30. Laplace Regression

This linear regression specifies that $\mathbf{y}$ 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 48), 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.

30.1. Form

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

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

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

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

30.2. 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(tcrossprod(beta, 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)
```

30.3. Initial Values

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

30.4. Model

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

```
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
    ### Log-Likelihood
    mu <- tcrossprod(beta, Data$X)
    LL <- sum(dlaplace(Data$y, mu, sigma, log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, sigma), yhat=mu,
         parm=parm)
    return(Modelout)
    }
```

31. Linear Regression

31.1. Form

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

31.2. 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(tcrossprod(beta, 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)
```

31.3. Initial Values

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

31.4. Model

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

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

32. Linear Regression, Frequentist

By eliminating prior probabilities, a frequentist linear regression example is presented. Although frequentism is not endorsed here, the purpose of this example is to illustrate how the **LaplacesDemon** package can be used for Bayesian or frequentist inference.

32.1. Form

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

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

32.2. 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(tcrossprod(beta, X) + e)
mon.names <- c("LL", "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)
```

32.3. Initial Values

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

32.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     sigma <- exp(parm[Data$J+1])
     ### Log-Likelihood
     mu <- tcrossprod(beta, Data$X)
     LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
     Modelout <- list(LP=LL, Dev=-2*LL, Monitor=c(LL, sigma), yhat=mu,
          parm=parm)
     return(Modelout)
     }
```

33. Linear Regression, Multilevel

33.1. Form

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

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

33.2. 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 <- rowSums(beta[m,] * X) + rnorm(N,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)
```

33.3. Initial.Values

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

33.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- matrix(parm[1:(Data$G * Data$J)], Data$G, Data$J)
     gamma <- parm[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 <- sum(dmvn(beta, gamma, Sigma, log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sqrt(100), log=TRUE))
     sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
     ### Log-Likelihood
     mu <- rowSums(beta[Data$m,] * Data$X)
     LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
     ### Log-Posterior
     LP <- LL + Omega.prior + beta.prior + gamma.prior + sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma),
          yhat=mu, parm=parm)
     return(Modelout)
     }
```

34. 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.

34.1. Form

$$\mathbf{y} \sim \mathcal{N}(\mu_2, \sigma_2^2)$$
$$\mu_2 = \mathbf{X}\beta$$
$$\mathbf{X}_{1:N,2} \sim \mathcal{N}(\mu_1, \sigma_1^2)$$
$$\mu_1 = \mathbf{X}_{1:N,(1,3:J)}\alpha$$
$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, (J-1)$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \ldots, J$$
$$\sigma_k \sim \mathcal{HC}(25), \quad k = 1, \ldots, 2$$

34.2. 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] <- tcrossprod(alpha, X[,-2]) + rnorm(N,0,0.1)
beta <- runif(J,-2,2)
y <- as.vector(tcrossprod(beta, 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)
```

34.3. Initial Values

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

34.4. 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$J+1)])
### Log(Prior Densities)
alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
### Log-Likelihood
mu1 <- tcrossprod(alpha, Data$X[,-2])
X.imputed <- Data$X
X.imputed[,2] <- ifelse(is.na(Data$X[,2]), mu1, Data$X[,2])
LL1 <- sum(dnorm(X.imputed[,2], mu1, sigma[1], log=TRUE))
mu2 <- tcrossprod(beta, X.imputed)
y.imputed <- ifelse(is.na(Data$y), mu2, Data$y)
LL2 <- sum((1-Data$M) * dnorm(y.imputed, mu2, sigma[2], log=TRUE))
### Log-Posterior
LP <- LL1 + LL2 + alpha.prior + beta.prior + sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL2, Monitor=c(LP,sigma),
     yhat=mu2, parm=parm)
return(Modelout)
}
```

35. Linear Regression with Missing Response

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

35.1. Form

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

35.2. 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)
```

35.3. Initial Values

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

35.4. Model

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

36. MANCOVA

Since this is a multivariate extension of ANCOVA, please see the ANCOVA example in section 1 for a univariate introduction.

36.1. Form

$$\mathbf{Y}_{i,1:J} \sim \mathcal{N}(\mu_{i,1:J}, \Sigma), \quad i = 1, \ldots, N$$

$$\mu_{i,k} = \alpha_k + \beta_{k,\mathbf{X}[i,1]} + \gamma_{k,\mathbf{X}[i,1]} + \mathbf{X}_{1:N,3:(C+J)}\delta_{k,1:C}$$

$$\epsilon_{i,k} = \mathbf{Y}_{i,k} - \mu_{i,k}$$

$$\alpha_k \sim \mathcal{N}(0, 1000), \quad k = 1, \ldots, K$$

$$\beta_{k,l} \sim \mathcal{N}(0, \sigma_1^2), \quad l = 1, \ldots, (L-1)$$

$$\beta_{1:K,L} = -\sum_{l=1}^{L-1} \beta_{1:K,l}$$

$$\gamma_{k,m} \sim \mathcal{N}(0, \sigma_2^2), \quad m = 1, \dots, (M-1)$$

$$\gamma_{1:K,M} = -\sum_{m=1}^{M-1} \beta_{1:K,m}$$

$$\delta_{k,c} \sim \mathcal{N}(0, 1000)$$

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

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

$$\sigma_{1:J} \sim \mathcal{HC}(25)$$

36.2. Data

```
C <- 2 #Number of covariates
J <- 2 #Number of factors (treatments)
K <- 3 #Number of endogenous (dependent) variables
L <- 4 #Number of levels in factor (treatment) 1
M <- 5 #Number of levels in factor (treatment) 2
N <- 100
X <- matrix(cbind(round(runif(N, 0.5, L+0.49)),round(runif(N,0.5,M+0.49)),
     runif(C*N,0,1)), N, J + C)
alpha <- runif(K,-1,1)
beta <- matrix(runif(K*L,-2,2), K, L)
beta[,L] <- -rowSums(beta[,-L])
gamma <- matrix(runif(K*M,-2,2), K, M)
gamma[,M] <- -rowSums(gamma[,-M])
delta <- matrix(runif(K*C), K, C)
Y <- matrix(NA,N,K)
for (k in 1:K) {
     Y[,k] <- alpha[k] + beta[k,X[,1]] + gamma[k,X[,2]] +
     tcrossprod(delta[k,], X[,-c(1,2)]) + rnorm(1,0,0.1)}
S <- diag(K)
mon.names <- c("LP", "s.o.beta", "s.o.gamma", "s.o.epsilon",
     parm.names(list(s.beta=rep(0,K), s.gamma=rep(0,K),
     s.epsilon=rep(0,K))))
parm.names <- parm.names(list(alpha=rep(0,K), beta=matrix(0,K,(L-1)),
     gamma=matrix(0,K,(M-1)), delta=matrix(0,K,C), Omega=diag(K),
     log.sigma=rep(0,2)), uppertri=c(0,0,0,0,1,0))
MyData <- list(C=C, J=J, K=K, L=L, M=M, N=N, S=S, X=X, Y=Y,
     mon.names=mon.names, parm.names=parm.names)
```

36.3. Initial Values

```
Initial.Values <- c(rep(0,K), rep(0,K*(L-1)), rep(0,K*(M-1)),
    rep(0,C*K), S[upper.tri(S, diag=TRUE)], rep(0,2))
```

36.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[grep("alpha", Data$parm.names)]
    beta <- matrix(c(parm[grep("beta", Data$parm.names)], rep(0,K)),
    Data$K, Data$L)
    beta[,L] <- -rowSums(beta[,-L])
    gamma <- matrix(c(parm[grep("gamma", Data$parm.names)], rep(0,K)),
        Data$K, Data$M)
    gamma[,M] <- -rowSums(gamma[,-M])
    delta <- matrix(parm[grep("delta", Data$parm.names)], Data$K, Data$C)
    Omega <- matrix(NA, Data$K, Data$K)
    Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
        Data$parm.names)]
    Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
    Sigma <- solve(Omega)
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
    alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
    beta.prior <- sum(dnorm(beta, 0, sigma[1], log=TRUE))
    gamma.prior <- sum(dnorm(gamma, 0, sigma[2], log=TRUE))
    delta.prior <- sum(dnorm(delta, 0, sqrt(1000), log=TRUE))
    Omega.prior <- dwishart(Omega, NROW(Data$S), Data$S, log=TRUE)
    sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
    ### Log-Likelihood
    mu <- matrix(0,Data$N,Data$K)
    for (k in 1:K) {
        mu[,k] <- alpha[k] + beta[k,Data$X[,1]] + gamma[k,Data$X[,2]] +
        tcrossprod(delta[k,], Data$X[,-c(1,2)])}
    LL <- sum(dmvn(Data$Y, mu, Sigma, log=TRUE))
    ### Variance Components, Omnibus
    s.o.beta <- sd(as.vector(beta))
    s.o.gamma <- sd(as.vector(gamma))
    s.o.epsilon <- sd(as.vector(Data$Y - mu))
    ### Variance Components, Univariate
    s.beta <- sd(t(beta))
    s.gamma <- sd(t(gamma))
    s.epsilon <- sd(Data$Y - mu)
    ### Log-Posterior
    LP <- LL + alpha.prior + beta.prior + gamma.prior + delta.prior +
```

```
    Omega.prior + sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, s.o.beta, s.o.gamma,
    s.o.epsilon, s.beta, s.gamma, s.epsilon), yhat=mu, parm=parm)
return(Modelout)
}
```

37. MANOVA

Since this is a multivariate extension of ANOVA, please see the two-way ANOVA example in section 3 for a univariate introduction.

37.1. Form

$$\mathbf{Y}_{i,1:J} \sim \mathcal{N}(\mu_{i,1:J}, \Sigma), \quad i = 1, \dots, N$$

$$\mu_{i,k} = \alpha_k + \beta_{k,\mathbf{X}[i,1]} + \gamma_{k,\mathbf{X}[i,1]}$$

$$\epsilon_{i,k} = \mathbf{Y}_{i,k} - \mu_{i,k}$$

$$\alpha_k \sim \mathcal{N}(0, 1000), \quad k = 1, \dots, K$$

$$\beta_{k,l} \sim \mathcal{N}(0, \sigma_1^2), \quad l = 1, \dots, (L-1)$$

$$\beta_{1:K,L} = -\sum_{l=1}^{L-1} \beta_{1:K,l}$$

$$\gamma_{k,m} \sim \mathcal{N}(0, \sigma_2^2), \quad m = 1, \dots, (M-1)$$

$$\gamma_{1:K,M} = -\sum_{m=1}^{M-1} \beta_{1:K,m}$$

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

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

$$\sigma_{1:J} \sim \mathcal{HC}(25)$$

37.2. Data

```
J <- 2 #Number of factors (treatments)
K <- 3 #Number of endogenous (dependent) variables
L <- 4 #Number of levels in factor (treatment) 1
M <- 5 #Number of levels in factor (treatment) 2
N <- 100
X <- matrix(cbind(round(runif(N, 0.5, L+0.49)),round(runif(N,0.5,M+0.49))),
    N, J)
alpha <- runif(K,-1,1)
beta <- matrix(runif(K*L,-2,2), K, L)
```

```
beta[,L] <- -rowSums(beta[,-L])
gamma <- matrix(runif(K*M,-2,2), K, M)
gamma[,M] <- -rowSums(gamma[,-M])
Y <- matrix(NA,N,K)
for (k in 1:K) {
     Y[,k] <- alpha[k] + beta[k,X[,1]] + gamma[k,X[,2]] + rnorm(1,0,0.1)}
S <- diag(K)
mon.names <- c("LP", "s.o.beta", "s.o.gamma", "s.o.epsilon",
     parm.names(list(s.beta=rep(0,K), s.gamma=rep(0,K),
     s.epsilon=rep(0,K))))
parm.names <- parm.names(list(alpha=rep(0,K), beta=matrix(0,K,(L-1)),
     gamma=matrix(0,K,(M-1)), Omega=diag(K), log.sigma=rep(0,2)),
     uppertri=c(0,0,0,1,0))
MyData <- list(J=J, K=K, L=L, M=M, N=N, S=S, X=X, Y=Y,
     mon.names=mon.names, parm.names=parm.names)
```

37.3. Initial Values

```
Initial.Values <- c(rep(0,K), rep(0,K*(L-1)), rep(0,K*(M-1)),
     S[upper.tri(S, diag=TRUE)], rep(0,2))
```

37.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[grep("alpha", Data$parm.names)]
     beta <- matrix(c(parm[grep("beta", Data$parm.names)], rep(0,K)),
          Data$K, Data$L)
     beta[,L] <- -rowSums(beta[,-L])
     gamma <- matrix(c(parm[grep("gamma", Data$parm.names)], rep(0,K)),
          Data$K, Data$M)
     gamma[,M] <- -rowSums(gamma[,-M])
     Omega <- matrix(NA, Data$K, Data$K)
     Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
          Data$parm.names)]
     Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
     Sigma <- solve(Omega)
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
     beta.prior <- sum(dnorm(beta, 0, sigma[1], log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sigma[2], log=TRUE))
     Omega.prior <- dwishart(Omega, NROW(Data$S), Data$S, log=TRUE)
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
```

```
mu <- matrix(0,Data$N,Data$K)
for (k in 1:K) {
     mu[,k] <- alpha[k] + beta[k,Data$X[,1]] + gamma[k,Data$X[,2]]}
LL <- sum(dmvn(Data$Y, mu, Sigma, log=TRUE))
### Variance Components, Omnibus
s.o.beta <- sd(as.vector(beta))
s.o.gamma <- sd(as.vector(gamma))
s.o.epsilon <- sd(as.vector(Data$Y - mu))
### Variance Components, Univariate
s.beta <- sd(t(beta))
s.gamma <- sd(t(gamma))
s.epsilon <- sd(Data$Y - mu)
### Log-Posterior
LP <- LL + alpha.prior + beta.prior + gamma.prior + Omega.prior +
     sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP, s.o.beta, s.o.gamma,
     s.o.epsilon, s.beta, s.gamma, s.epsilon), yhat=mu, parm=parm)
return(Modelout)
}
```

38. Mixture Model, Finite

This finite mixture model (FMM) imposes a multilevel structure on each of the J regression effects in β, so that mixture components share a common residual variance, ν_j. Identifiability is gained at the expense of some shrinkage.

38.1. Form

$$\mathbf{y} \sim \mathcal{N}(\mu_{1:N,m}, \sigma^2)$$

$$\mu_{1:N,m} = \mathbf{X}\beta_{m,1:J}, \quad m = 1, \dots, M$$

$$\beta_{m,j} \sim \mathcal{N}(0, \nu_j^2), \quad j = 1, \dots, J$$

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

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

$$\pi_{1:M} \sim \mathcal{D}(\alpha_{1:M})$$

$$\pi_m = \frac{\sum_{i=1}^N \delta_{i,m}}{\sum \delta}$$

$$\mathbf{p}_{i,m} = \frac{\delta_{i,m}}{\sum_{m=1}^M \delta_{i,m}}$$

$$\delta_{i,m} = \exp(\mathbf{X}\delta_{i,m}), \quad m = 1, \dots, (M-1)$$

$$\delta_{1:N,M} = 1$$

$$\delta_{i,m} \sim \mathcal{N}(0, 1000) \in [-10, 10], \quad m = 1, \ldots, (M-1)$$
$$\alpha_m = 1$$

38.2. Data

```
M <- 2 #Number of mixtures
alpha <- rep(1,M) #Prior probability of mixing probabilities
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", parm.names(list(pi=rep(0,M), sigma=0)))
parm.names <- parm.names(list(beta=matrix(0,M,J), log.nu=rep(0,J),
     log.delta=matrix(0,N,M-1), log.sigma=0))
MyData <- list(J=J, M=M, N=N, X=X, alpha=alpha, mon.names=mon.names,
     parm.names=parm.names, y=y)
```

38.3. Initial Values

```
Initial.Values <- c(runif(M*J), rep(0,J), runif(N*(M-1),-1,1), 0)
```

38.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- matrix(parm[grep("beta", Data$parm.names)], Data$M, Data$J)
     delta <- interval(parm[grep("log.delta", Data$parm.names)], -10, 10)
     parm[grep("log.delta", Data$parm.names)] <- delta
     delta <- matrix(c(exp(delta), rep(1, Data$N)), Data$N, Data$M)
     pi <- colSums(delta) / sum(delta)
     nu <- exp(parm[grep("log.nu", Data$parm.names)])
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, matrix(rep(nu, Data$M), Data$M,
          Data$J, byrow=TRUE), log=TRUE))
     delta.prior <- sum(dtrunc(delta, "norm", a=exp(-10), b=exp(10),
          mean=log(1/Data$M), sd=sqrt(1000), log=TRUE))
     pi.prior <- ddirichlet(pi, Data$alpha, log=TRUE)
     nu.prior <- sum(dhalfcauchy(nu, 25, log=TRUE))
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
     p <- delta / rowSums(delta)
     LL <- mu <- matrix(NA, Data$N, Data$M)
```

```
for (m in 1:M) {mu[,m] <- tcrossprod(beta[m,], Data$X)}
p <- apply(p, 1, which.max)
mu <- diag(mu[,p])
LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + delta.prior + pi.prior + nu.prior +
     sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,pi,sigma),
     yhat=mu, parm=parm)
return(Modelout)
}
```

39. Multinomial Logit

39.1. Form

$$\mathbf{y}_i \sim \mathcal{CAT}(\mathbf{p}_{i,1:J}), \quad i = 1, \dots, N$$

$$\mathbf{p}_{i,j} = \frac{\phi_{i,j}}{\sum_{j=1}^{J} \phi_{i,j}}, \quad \sum_{j=1}^{J} \mathbf{p}_{i,j} = 1$$

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

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

$$\mu_{i,j} = \mathbf{X}_{i,1:K} \beta_{j,1:K} \in [-700, 700], \quad j = 1, \dots, (J-1)$$

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

39.2. 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)
```

39.3. Initial Values

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

39.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:(Data$J-1*Data$K)]
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     ### Log-Likelihood
     mu <- matrix(0,Data$N,Data$J)
     mu[,1] <- tcrossprod(beta[1:3], Data$X)
     mu[,2] <- tcrossprod(beta[4:6], Data$X)
     mu <- interval(mu, -700, 700)
     phi <- exp(mu)
     p <- phi / rowSums(phi)
     LL <- sum(dcat(Data$y, p, log=TRUE))
     yrep <- apply(p,1,which.max)
     ### Log-Posterior
     LP <- LL + beta.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
     return(Modelout)
     }
```

40. Multinomial Logit, Nested

40.1. Form

$$\mathbf{y}_i \sim \mathcal{CAT}(\mathbf{P}_{i,1:J}), \quad i = 1, \dots, N$$

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

$$\mathbf{P}_{1:N,2} = \frac{(1 - \mathbf{P}_{1:N,1})\mathbf{S}_{1:N,1}}{\mathbf{V}}$$

$$\mathbf{P}_{1:N,3} = \frac{(1 - \mathbf{P}_{1:N,1})\mathbf{S}_{1:N,2}}{\mathbf{V}}$$

$$\mathbf{R}_{1:N} = \exp(\mu_{1:N,1})$$
$$\mathbf{S}_{1:N,1:2} = \exp(\mu_{1:N,2:3})$$
$$\mathbf{I} = \log(\mathbf{V})$$
$$\mathbf{V}_i = \sum_{k=1}^{K} \mathbf{S}_{i,k}, \quad i = 1, \dots, N$$
$$\mu_{1:N,1} = \mathbf{X}\iota \in [-700, 700]$$
$$\mu_{1:N,2} = \mathbf{X}\beta_{2,1:K} \in [-700, 700]$$
$$\iota = \alpha\beta_{1,1:K}$$
$$\alpha \sim \mathcal{EXP}(1) \in [0, 2]$$
$$\beta_{j,k} \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, (J-1) \quad k = 1, \dots, K$$

where there are $J = 3$ categories of $\mathbf{y}$, $K = 3$ predictors, $\mathbf{R}$ is the non-nested alternative, $\mathbf{S}$ is the nested alternative, $\mathbf{V}$ 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.

40.2. 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)
```

40.3. Initial Values

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

40.4. 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$J)
iota <- alpha * beta[1,]
mu[,1] <- tcrossprod(iota, Data$X)
mu[,2] <- tcrossprod(beta[2,], Data$X)
mu <- interval(mu, -700, 700)
R <- exp(mu[,1])
S <- exp(mu[,2:3])
V <- rowSums(S)
I <- log(V)
P[,1] <- R / (R + exp(alpha*I))
P[,2] <- (1 - P[,1]) * S[,1] / V
P[,3] <- (1 - P[,1]) * S[,2] / V
LL <- sum(dcat(Data$y, P, log=TRUE))
yrep <- apply(P,1,which.max)
### Log-Posterior
LP <- LL + alpha.prior + beta.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,iota), yhat=yrep,
     parm=parm)
return(Modelout)
}
```

41. Multinomial Probit

In this form of MNP, the β parameters are sum-to-zero constraints in the reference category, and covariance matrix Σ includes all J categories of $\mathbf{y}$.

Note that the parameters and initial values for the upper triangular elements of Σ are read in as Σ, though the diagonal is read in as $\log(\Sigma)$, but still denoted as Σ. Apologies for any confusion this causes, and the diagonal elements could each be renamed manually in `parm.names`. The only reason this difference exists is that I am unsure of how to program that in `parm.names` for all occasions.

41.1. Form

$$\mathbf{Z}_{i,1:J} \sim \mathcal{N}_J(\mu_{i,1:J}, \Sigma), \quad i = 1, \ldots, N$$

$$\mathbf{Z}_{i,j} \in \begin{cases} \text{[0,10]} & \text{if } \mathbf{y}_i = j \\ \text{[-10,0]} & \end{cases}$$

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

$$\Sigma \sim \mathcal{IW}(J, \mathbf{R}), \quad \mathbf{R} = \mathbf{I}_J, \quad \Sigma[1,1] = 1$$

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

$$\beta_{J,k} = -\sum_{j=1}^{J-1} \beta_{j,k}$$

$$\mathbf{Z}_{i,j} \sim \mathcal{N}(0, 1000) \in [-10, 10]$$

41.2. Data

```
y <- x1 <- x2 <- c(1:30)
y[1:10] <- 1
y[11:20] <- 2
y[21:30] <- 3
x1[1:10] <- rnorm(10, 25, 2.5)
x1[11:20] <- rnorm(10, 40, 4.0)
x1[21:30] <- rnorm(10, 35, 3.5)
x2[1:10] <- rnorm(10, 2.51, 0.25)
x2[11:20] <- rnorm(10, 2.01, 0.20)
x2[21:30] <- rnorm(10, 2.70, 0.27)
N <- length(y)
J <- 3 #Number of categories in y
K <- 3 #Number of columns to be in design matrix X
R <- diag(J)
X <- matrix(c(rep(1,N),x1,x2),N,K)
mon.names <- "LP"
sigma.temp <- parm.names(list(Sigma=diag(J)), uppertri=1)
parm.names <- c(sigma.temp[2:length(sigma.temp)],
    parm.names(list(beta=matrix(0,(J-1),K), Z=matrix(0,N,J))))
MyData <- list(J=J, K=K, N=N, R=R, X=X, mon.names=mon.names,
    parm.names=parm.names, y=y)
```

41.3. Initial Values

```
Initial.Values <- c(rep(0,length(R[upper.tri(R, diag=TRUE)])-1),
    rep(0,(J-1)*K), rep(0,N,J))
```

41.4. Model

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

```
### Parameters
beta <- matrix(parm[grep("beta", Data$parm.names)], Data$J-1, Data$K)
beta <- rbind(beta, colSums(beta)*-1) #Sum to zero constraint
Sigma <- matrix(NA, Data$J, Data$J)
Sigma[upper.tri(Sigma, diag=TRUE)] <- c(0, parm[grep("Sigma",
     Data$parm.names)])
Sigma[lower.tri(Sigma)] <- Sigma[upper.tri(Sigma)]
diag(Sigma) <- exp(diag(Sigma))
Z <- matrix(parm[grep("Z", Data$parm.names)], Data$N, Data$J)
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
Sigma.prior <- dinvwishart(Sigma, Data$J, Data$R, log=TRUE)
Z.prior <- sum(dnorm(Z, 0, sqrt(1000), log=TRUE))
### Log-Likelihood
mu <- matrix(0,Data$N,Data$J)
for (j in 1:Data$J) {mu[,j] <- tcrossprod(beta[j,], Data$X)}
Y <- indmat(Data$y)
Z <- ifelse(Z > 10, 10, Z); Z <- ifelse({Y == 0} & {Z > 0}, 0, Z)
Z <- ifelse(Z < -10, -10, Z); Z <- ifelse({Y == 1} & {Z < 0}, 0, Z)
parm[grep("Z", Data$parm.names)] <- as.vector(Z)
LL <- sum(dmvn(Z, mu, Sigma, log=TRUE))
yrep <- apply(Z, 1, which.max)
#eta <- exp(mu)
#p <- eta / rowSums(eta)
### Log-Posterior
LP <- LL + beta.prior + Sigma.prior + Z.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=yrep, parm=parm)
return(Modelout)
}
```

42. 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 2011), an R package on CRAN. However, also note that Laplace's Demon (eventually) provides a better answer (higher ESS, lower DIC, etc.).

42.1. Form

$$\mathbf{y}_j \sim \mathcal{N}(\theta_j, \tau_j^{-1}), \quad j = 1, \dots, J$$
$$\theta_j \sim \mathcal{N}(\theta_\mu, \theta_\tau^{-1}), \quad j = 1, \dots, J$$
$$\theta_\mu \sim \mathcal{N}(0, 1000)$$
$$\theta_\tau = \frac{1}{\theta_\sigma^2}$$

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

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

42.2. 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)
```

42.3. Initial Values

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

42.4. Model

```
Model <- function(parm, Data)
    {
    ### 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, sqrt(1000), 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)
    }
```

43. Panel, Autoregressive Poisson

43.1. Form

$$\mathbf{Y} \sim \mathcal{P}(\Lambda)$$

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

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

$$\alpha_i \sim \mathcal{N}(\alpha_\mu, \alpha_\sigma^2), \quad i = 1, \dots, N$$

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

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

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

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

43.2. 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)
```

43.3. Initial Values

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

43.4. 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)
}
```

44. Penalized Spline Regression

This example is adapted from Crainiceanu, Ruppert, and Wand (2005). The user specifies the degree D of polynomials and the number K of knots. Regression effects β regard the polynomial in design matrix $\mathbf{X}$, and γ regard the splines in design matrix $\mathbf{S}$.

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

$$\mu = \mathbf{X}\beta + \mathbf{S}\gamma$$

$$\mathbf{S}_{i,k} = \begin{cases} (\mathbf{x}_i - k)^D & \text{if } \mathbf{S}_{i,k} > 0 \\ 0 & \end{cases}$$

$$\mathbf{X}_{i,d} = \mathbf{x}_i^{d-1}, \quad d = 2, \dots, (D+1)$$

$$\mathbf{X}_{i,1} = 1$$

$$\beta_d \sim \mathcal{N}(0, 1000), \quad d = 1, \dots, (D+1)$$

$$\gamma_k \sim \mathcal{N}(0, \sigma_2^2), \quad k = 1, \dots, K$$

$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \dots, 2$$

44.1. Form

44.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
K <- 10 #Number of knots
D <- 2 #Degree of polynomial
y <- log(demonsnacks$Calories)
x <- demonsnacks[,7]
k <- as.vector(quantile(x, probs=(1:K / (K+1))))
mon.names <- "LP"
parm.names <- parm.names(list(beta=rep(0,D+1), gamma=rep(0,K),
     log.sigma=rep(0,2)))
MyData <- list(D=D, K=K, N=N, mon.names=mon.names,
     parm.names=parm.names, k=k, x=x, y=y)
```

44.3. Initial Values

```
Initial.Values <- c(rep(0,D+1), rep(0,K), log(c(1,1)))
```

44.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[grep("beta", Data$parm.names)]
     gamma <- parm[grep("gamma", Data$parm.names)]
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     gamma.prior <- sum(dnorm(gamma, 0, sigma[2], log=TRUE))
     sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
     ### Log-Likelihood
     X <- matrix(Data$x, Data$N, Data$D)
     for (d in 2:Data$D) X[,d] <- X[,d]^d
     X <- cbind(1,X)
     S <- matrix(Data$x, Data$N, Data$K) -
          matrix(Data$k, Data$N, Data$K, byrow=TRUE)
     S <- ifelse(S > 0, S, 0)
     S <- S^Data$D
     mu <- tcrossprod(beta, X) + tcrossprod(gamma, S)
     LL <- sum(dnorm(Data$y, mu, sigma[1], log=TRUE))
     ### Log-Posterior
     LP <- LL + beta.prior + gamma.prior + sigma.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=mu, parm=parm)
```

```
    return(Modelout)
    }
```

45. Poisson Regression

45.1. Form

$$\mathbf{y} \sim \mathcal{P}(\lambda)$$

$$\lambda = \exp(\mathbf{X}\beta)$$

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

45.2. 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(tcrossprod(beta, 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)
```

45.3. Initial Values

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

45.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[1:Data$J]
    ### Log(Prior Densities)
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    lambda <- exp(tcrossprod(beta, Data$X))
    LL <- sum(dpois(Data$y, lambda, log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP,
         yhat=lambda, parm=parm)
```

```
    return(Modelout)
    }
```

46. Polynomial Regression

In this univariate example, the degree of the polynomial is specified as D. For a more robust extension to estimating nonlinear relationships between $\mathbf{y}$ and $\mathbf{x}$, see penalized spline regression in section 44.

46.1. Form

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

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

$$\mathbf{X}_{i,d} = \mathbf{x}_i^{d-1}, \quad d = 1, \dots, (D+1)$$

$$\mathbf{X}_{i,1} = 1$$

$$\beta_d \sim \mathcal{N}(0, 1000), \quad d = 1, \dots, (D+1)$$

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

46.2. Data

```
data(demonsnacks)
N <- NROW(demonsnacks)
D <- 2 #Degree of polynomial
y <- log(demonsnacks$Calories)
x <- demonsnacks[,7]
mon.names <- "LP"
parm.names <- parm.names(list(beta=rep(0,D+1), log.sigma=0))
MyData <- list(D=D, N=N, mon.names=mon.names, parm.names=parm.names, x=x,
    y=y)
```

46.3. Initial Values

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

46.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- parm[grep("beta", Data$parm.names)]
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
```

```
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
### Log-Likelihood
X <- matrix(Data$x, Data$N, Data$D)
for (d in 2:Data$D) {X[,d] <- X[,d]^d}
X <- cbind(1,X)
mu <- tcrossprod(beta, X)
LL <- sum(dnorm(Data$y, mu, sigma, log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + sigma.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=mu, parm=parm)
return(Modelout)
}
```

47. Revision, Normal

This example provides both an analytic solution and numerical approximation of the revision of a normal distribution. Given a normal prior distribution (α) and data distribution (β), the posterior (γ) is the revised normal distribution. This is an introductory example of Bayesian inference, and allows the user to experiment numerical approximation, such as with MCMC in LaplacesDemon. Note that, regardless of the data sample size N in this example, Laplace Approximation is inappropriate due to asymptotics since the data (β) is perceived by the algorithm as a single datum rather than a collection of data. MCMC, on the other hand, is biased only by the effective number of samples taken of the posterior.

```
### Analytic Solution
prior.mu <- 0
prior.sigma <- 10
N <- 10
data.mu <- 1
data.sigma <- 2
posterior.mu <- (prior.sigma^-2 * prior.mu + N * data.sigma^-2 * data.mu) /
    (prior.sigma^-2 + N * data.sigma^-2)
posterior.sigma <- sqrt(1/(prior.sigma^-2 + data.sigma^-2))
posterior.mu
posterior.sigma
```

47.1. Form

$$\alpha \sim \mathcal{N}(0, 10)$$

$$\beta \sim \mathcal{N}(1, 2)$$

$$\gamma = \frac{\alpha_\sigma^{-2}\alpha + N\beta_\sigma^{-2}\beta}{\alpha_\sigma^{-2} + N\beta_\sigma^{-2}}$$

47.2. Data

```
N <- 10
mon.names <- c("LP","gamma")
parm.names <- c("alpha","beta")
MyData <- list(N=N, mon.names=mon.names, parm.names=parm.names)
```

47.3. Initial Values

```
Initial.Values <- c(0,0)
```

47.4. Model

```
Model <- function(parm, Data)
     {
     ### Hyperparameters
     alpha.mu <- 0
     alpha.sigma <- 10
     beta.mu <- 1
     beta.sigma <- 2
     ### Parameters
     alpha <- parm[1]
     beta <- parm[2]
     ### Log(Prior Density)
     alpha.prior <- dnorm(alpha, alpha.mu, alpha.sigma, log=TRUE)
     ### Log-Likelihood Density
     LL <- dnorm(beta, beta.mu, beta.sigma, log=TRUE)
     ### Posterior
     gamma <- (alpha.sigma^-2 * alpha + N * beta.sigma^-2 * beta) /
          (alpha.sigma^-2 + N * beta.sigma^-2)
     ### Log(Posterior Density)
     LP <- LL + alpha.prior
     Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,gamma), yhat=LL,
          parm=parm)
     return(Modelout)
     }
```

48. 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 30).

48.1. Form

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

48.2. 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(tcrossprod(beta, 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)
```

48.3. Initial Values

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

48.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     beta <- parm[1:Data$J]
     sigma <- exp(parm[Data$J+1])
     nu <- exp(parm[Data$J+2])
     ### Log(Prior Densities)
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
     sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
     nu.prior <- dhalfcauchy(nu, 25, log=TRUE)
     ### Log-Likelihood
     mu <- tcrossprod(beta, Data$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)
}
```

49. Seemingly Unrelated Regression (SUR)

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

49.1. Form

$$\mathbf{Y}_{t,k} \sim \mathcal{N}_K(\mu_{t,k}, \Sigma), \quad t = 1, \dots, T; \quad k = 1, \dots, K$$

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

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

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

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

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

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

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

49.2. 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)
```

49.3. Initial Values

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

49.4. 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)
    alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
    beta.prior <- sum(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)
    }
```

50. Simultaneous Equations

This example of simultaneous equations uses Klein's Model I (Kleine 1950) regarding economic fluctations in the United States in 1920-1941 (**N**=22). Usually, this example is modeled with 3-stage least sqaures (3SLS), excluding the uncertainty from multiple stages. By constraining each element in the instrumental variables matrix $\nu \in [-10, 10]$, this example estimates the model without resorting to stages. The dependent variable is matrix $\mathbf{Y}$, in which $\mathbf{Y}_{1,1:N}$ is **C** or Consumption, $\mathbf{Y}_{2,1:N}$ is **I** or Investment, and $\mathbf{Y}_{3,1:N}$ is **Wp** or Private Wages. Here is a data dictionary:

```
    A = Time Trend measured as years from 1931
    C = Consumption
    G = Government Nonwage Spending
    I = Investment
```

```
K = Capital Stock
P = Private (Corporate) Profits
T = Indirect Business Taxes Plus Neg Exports
Wg = Government Wage Bill
Wp = Private Wages
X = Equilibrium Demand (GNP)
```

See Kleine (1950) for more information.

50.1. Form

$$\mathbf{Y} \sim \mathcal{N}(\mu, \Sigma)$$
$$\mu_{1,1} = \alpha_1 + \alpha_2 \nu_{1,1} + \alpha_4 \nu_{2,1}$$
$$\mu_{1,i} = \alpha_1 + \alpha_2 \nu_{1,i} + \alpha_3 \mathbf{P}_{i-1} + \alpha_4 \nu_{2,i}, \quad i = 2, \dots, N$$
$$\mu_{2,1} = \beta_1 + \beta_2 \nu_{1,1} + \beta_4 \mathbf{K}_1$$
$$\mu_{2,i} = \beta_1 + \beta_2 \nu_{1,i} + \beta_3 \mathbf{P}_{i-1} + \beta_4 \mathbf{K}_i, \quad i = 2, \dots, N$$
$$\mu_{3,1} = \gamma_1 + \gamma_2 \nu_{3,1} + \gamma_4 \mathbf{A}_1$$
$$\mu_{3,i} = \gamma_1 + \gamma_2 \nu_{3,i} + \gamma_3 \mathbf{X}_{i-1} + \gamma_4 \mathbf{A}_i, \quad i = 2, \dots, N$$
$$\mathbf{Z}_{j,i} \sim \mathcal{N}(\nu_{j,i}, \sigma_j^2), \quad j = 1, \dots, 3$$
$$\nu_{j,1} = \pi_{j,1} + \pi_{j,3} \mathbf{K}_1 + \pi_{j,5} \mathbf{A}_1 + \pi_{j,6} \mathbf{T}_1 + \pi_{j,7} \mathbf{G}_1, \quad j = 1, \dots, 3$$
$$\nu_{j,i} = \pi_{j,1} + \pi_{j,2} \mathbf{P}_{i-1} + \pi_{j,3} \mathbf{K}_i + \pi_{j,4} \mathbf{X}_{i-1} + \pi_{j,5} \mathbf{A}_i + \pi_{j,6} \mathbf{T}_i + \pi \mathbf{G}_i, \quad i = 1, \dots, N, \quad j = 1, \dots, 3$$
$$\alpha_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, 4$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, 4$$
$$\gamma_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, 4$$
$$\pi_{j,i} \sim \mathcal{N}(0, 1000) \in [-10, 10], \quad j = 1, \dots, 3, \quad i = 1, \dots, N$$
$$\sigma_j \sim \mathcal{HC}(25), \quad j = 1, \dots, 3$$
$$\Omega \sim \mathcal{W}(N, \mathbf{S}), \quad \mathbf{S} = \mathbf{I}_3$$
$$\Sigma = \Omega^{-1}$$

50.2. Data

```
N <- 22
A <- c(-11,-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10)
C <- c(39.8,41.9,45,49.2,50.6,52.6,55.1,56.2,57.3,57.8,55,50.9,45.6,46.5,
     48.7,51.3,57.7,58.7,57.5,61.6,65,69.7)
G <- c(2.4,3.9,3.2,2.8,3.5,3.3,3.3,4,4.2,4.1,5.2,5.9,4.9,3.7,4,4.4,2.9,4.3,
     5.3,6.6,7.4,13.8)
I <- c(2.7,-0.2,1.9,5.2,3,5.1,5.6,4.2,3,5.1,1,-3.4,-6.2,-5.1,-3,-1.3,2.1,2,
     -1.9,1.3,3.3,4.9)
```

```
K <- c(180.1,182.8,182.6,184.5,189.7,192.7,197.8,203.4,207.6,210.6,215.7,
     216.7,213.3,207.1,202,199,197.7,199.8,201.8,199.9,201.2,204.5)
P <- c(12.7,12.4,16.9,18.4,19.4,20.1,19.6,19.8,21.1,21.7,15.6,11.4,7,11.2,
     12.3,14,17.6,17.3,15.3,19,21.1,23.5)
T <- c(3.4,7.7,3.9,4.7,3.8,5.5,7,6.7,4.2,4,7.7,7.5,8.3,5.4,6.8,7.2,8.3,6.7,
     7.4,8.9,9.6,11.6)
Wg <- c(2.2,2.7,2.9,2.9,3.1,3.2,3.3,3.6,3.7,4,4.2,4.8,5.3,5.6,6,6.1,7.4,
     6.7,7.7,7.8,8,8.5)
Wp <- c(28.8,25.5,29.3,34.1,33.9,35.4,37.4,37.9,39.2,41.3,37.9,34.5,29,28.5,
     30.6,33.2,36.8,41,38.2,41.6,45,53.3)
X <- c(44.9,45.6,50.1,57.2,57.1,61,64,64.4,64.5,67,61.2,53.4,44.3,45.1,
     49.7,54.4,62.7,65,60.9,69.5,75.7,88.4)
year <- c(1920,1921,1922,1923,1924,1925,1926,1927,1928,1929,1930,1931,1932,
     1933,1934,1935,1936,1937,1938,1939,1940,1941)
Y <- matrix(c(C,I,Wp),3,N, byrow=TRUE)
Z <- matrix(c(P, Wp+Wg, X), 3, N, byrow=TRUE)
S <- diag(NROW(Y))
mon.names <- "LP"
parm.names <- parm.names(list(alpha=rep(0,4), beta=rep(0,4),
     gamma=rep(0,4), pi=matrix(0,3,7), log.sigma=rep(0,3),
     Omega=diag(3)), uppertri=c(0,0,0,0,0,1))
MyData <- list(A=A, C=C, G=G, I=I, K=K, N=N, P=P, S=S, T=T, Wg=Wg, Wp=Wp,
     X=X, Y=Y, Z=Z, mon.names=mon.names, parm.names=parm.names)
```

50.3. Initial Values

```
Initial.Values <- c(rep(0,4), rep(0,4), rep(0,4), rep(0,3*7), rep(0,3),     S[upper.tri(S,
diag=TRUE)])
```

50.4. Model

```
Model <- function(parm, Data)
     {
     ### Parameters
     alpha <- parm[1:4]; beta <- parm[5:8]; gamma <- parm[9:12]
     pi <- matrix(interval(parm[grep("pi", Data$parm.names)],-10,10), 3, 7)
     parm[grep("pi", Data$parm.names)] <- as.vector(pi)
     sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
     Omega <- matrix(NA, 3, 3)
     Omega[upper.tri(Omega, diag=TRUE)] <- parm[grep("Omega",
          Data$parm.names)]
     Omega[lower.tri(Omega)] <- Omega[upper.tri(Omega)]
     Sigma <- solve(Omega)
     ### Log(Prior Densities)
     alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
     beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
```

```
gamma.prior <- sum(dnorm(gamma, 0, sqrt(1000), log=TRUE))
pi.prior <- sum(dnorm(pi, 0, sqrt(1000), log=TRUE))
sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
Omega.prior <- dwishart(Omega, NROW(Data$S), Data$S, log=TRUE)
### Log-Likelihood
mu <- nu <- matrix(0,3,Data$N)
for (i in 1:3) {
     nu[i,1] <- pi[i,1] + pi[i,3]*Data$K[1] + pi[i,5]*Data$A[1] +
          pi[i,6]*Data$T[1] + pi[i,7]*Data$G[1]
     nu[i,-1] <- pi[i,1] + pi[i,2]*Data$P[-Data$N] +
          pi[i,3]*Data$K[-1] + pi[i,4]*Data$X[-Data$N] +
          pi[i,5]*Data$A[-1] + pi[i,6]*Data$T[-1] +
          pi[i,7]*Data$G[-1]}
LL <- sum(dnorm(Data$Z, nu, matrix(sigma, 3, Data$N), log=TRUE))
mu[1,1] <- alpha[1] + alpha[2]*nu[1,1] + alpha[4]*nu[2,1]
mu[1,-1] <- alpha[1] + alpha[2]*nu[1,-1] +
     alpha[3]*Data$P[-Data$N] + alpha[4]*nu[2,-1]
mu[2,1] <- beta[1] + beta[2]*nu[1,1] + beta[4]*Data$K[1]
mu[2,-1] <- beta[1] + beta[2]*nu[1,-1] +
     beta[3]*Data$P[-Data$N] + beta[4]*Data$K[-1]
mu[3,1] <- gamma[1] + gamma[2]*nu[3,1] + gamma[4]*Data$A[1]
mu[3,-1] <- gamma[1] + gamma[2]*nu[3,-1] +
     gamma[3]*Data$X[-Data$N] + gamma[4]*Data$A[-1]
LL2 <- sum(dmvn(t(Data$Y), t(mu), Sigma, log=TRUE))
if(!is.nan(LL2)) LL <- LL + LL2
### Log-Posterior
LP <- LL + alpha.prior + beta.prior + gamma.prior + pi.prior +
     sigma.prior + Omega.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=LP, yhat=mu, parm=parm)
return(Modelout)
}
```

51. Space-Time, Dynamic

This approach to space-time or spatiotemporal modeling applies kriging to a stationary spatial component for points in space $s = 1, \dots, S$ first at time $t = 1$, where space is continuous and time is discrete. Vector ζ contains these spatial effects. Next, SSM (State-Space Model) or DLM (Dynamic Linear Model) components are applied to the spatial parameters (ϕ, κ, and λ) and regression effects (β). These parameters are allowed to vary dynamically with time $t = 2, \dots, T$, and the resulting spatial process is estimated for each of these time-periods. When time is discrete, a dyanmic space-time process can be applied. The matrix Θ contains the dynamically varying stationary spatial effects, or space-time effects. Spatial coordinates are given in longitude and latitude for $s = 1, \dots, S$ points in space and measurements are taken across discrete time-periods $t = 1, \dots, T$ for $\mathbf{Y}_{s,t}$. The dependent variable is also a function of design matrix $\mathbf{X}$ (which may also be dynamic, but is static in this example) and dynamic regression effects matrix $\beta_{1:J,1:T}$. For more information on kriging, see section 28. For more information on state-space or a DLM, see section 20. To extend this to a large spatial data set, consider incorporating the predictive process kriging example in section 29.

51.1. Form

$$\mathbf{Y}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \sigma_1^2), \quad s = 1, \dots, S, \quad t = 1, \dots, T$$
$$\mu_{s,t} = \mathbf{X}_{s,1:J}\beta_{1:J,t} + \Theta_{s,t}$$
$$\Theta_{s,t} = \frac{\Sigma_{s,s,t}}{\sum_{r=1}^{S} \Sigma_{r,s,t}} \Theta_{s,t-1}, \quad s = 1, \dots, S, \quad t = 2, \dots, T$$
$$\Theta_{s,1} = \zeta_s$$
$$\zeta \sim \mathcal{N}_S(0, \Sigma_{1:S,1:S,1})$$
$$\Sigma_{1:S,1:S,t} = \lambda_t^2 \exp(-\phi_t \mathbf{D})^{\kappa[t]}$$
$$\sigma_1 \sim \mathcal{HC}(25)$$
$$\beta_{j,1} \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, 2$$
$$\beta_{1,t} \sim \mathcal{N}(\beta_{1,t-1}, \sigma_2^2), \quad t = 2, \dots, T$$
$$\beta_{2,t} \sim \mathcal{N}(\beta_{2,t-1}, \sigma_3^2), \quad t = 2, \dots, T$$
$$\phi_1 \sim \mathcal{N}(0, 1000) \in [0, \infty]$$
$$\phi_t \sim \mathcal{N}(\phi_{t-1}, \sigma_4^2) \in [0, \infty], \quad t = 2, \dots, T$$
$$\kappa_1 \sim \mathcal{N}(0, 1000) \in [0, \infty]$$
$$\kappa_t \sim \mathcal{N}(\kappa_{t-1}, \sigma_5^2) \in [0, \infty], \quad t = 2, \dots, T$$
$$\lambda_1 \sim \mathcal{N}(0, 1000) \in [0, \infty]$$
$$\lambda_t \sim \mathcal{N}(\lambda_{t-1}, \sigma_6^2) \in [0, \infty], \quad t = 2, \dots, T$$

51.2. Data

```
S <- 20
T <- 10
longitude <- runif(S,0,100)
latitude <- runif(S,0,100)
D <- as.matrix(dist(cbind(longitude,latitude), diag=TRUE, upper=TRUE))
beta <- matrix(c(50,2), 2, T)
phi <- rep(1,T); kappa <- rep(1.5,T); lambda <- rep(10000,T)
for (t in 2:T) {
     beta[1,t-1] <- beta[1,t-1] + rnorm(1,0,1)
     beta[2,t-1] <- beta[2,t-1] + rnorm(1,0,0.1)
     phi[t] <- phi[t-1] + rnorm(1,0,0.1)
     if(phi[t] < 0.001) phi[t] <- 0.001
     kappa[t] <- kappa[t-1] + rnorm(1,0,0.1)
```

```
    lambda[t] <- lambda[t-1] + rnorm(1,0,1000)}
Sigma <- array(0, dim=c(S,S,T))
for (t in 1:T) {
    Sigma[ , ,t] <- lambda[t] * exp(-phi[t] * D)^kappa[t]}
zeta <- as.vector(apply(rmvn(1000, rep(0,S), Sigma[ , ,1]), 2, mean))
mu <- Theta <- matrix(zeta,S,T)
for (t in 2:T) {for (s in 1:S) {
    Theta[,t] <- sum(Sigma[,s,t] / sum(Sigma[,s,t]) * Theta[,t-1])}}
X <- matrix(runif(S*2,-2,2),S,2); X[,1] <- 1
for (t in 1:T) {mu[,t] <- as.vector(tcrossprod(beta[,t], X))}
Y <- mu + Theta + matrix(rnorm(S*T,0,0.1),S,T)
mon.names <- c("LP", parm.names(list(sigma=rep(0,6))))
parm.names <- parm.names(list(zeta=rep(0,S), beta=matrix(0,2,T),
    log.phi=rep(0,T), log.kappa=rep(0,T), log.lambda=rep(0,T),
    log.sigma=rep(0,6)))
MyData <- list(D=D, S=S, T=T, X=X, Y=Y, mon.names=mon.names,
    parm.names=parm.names)
```

51.3. Initial Values

```
Initial.Values <- c(rep(0,S), rep(c(mean(Y),0),T), log(rep(1,T)),
    log(rep(1,T)), rep(1,T), log(rep(1,6)))
```

51.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    beta <- matrix(parm[grep("beta", Data$parm.names)], 2, Data$T)
    zeta <- parm[grep("zeta", Data$parm.names)]
    phi <- exp(parm[grep("log.phi", Data$parm.names)])
    kappa <- exp(parm[grep("log.kappa", Data$parm.names)])
    lambda <- exp(parm[grep("log.lambda", Data$parm.names)])
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    Sigma <- array(0, dim=c(Data$S, Data$S, Data$T))
    for (t in 1:Data$T) {
        Sigma[ , ,t] <- lambda[t]^2 * exp(-phi[t] * Data$D)^kappa[t]}
    ### Log(Prior Densities)
    beta.prior <- phi.prior <- kappa.prior <- lambda.prior <- rep(0,
        Data$T)
    beta.prior <- sum(dnorm(beta[,1], 0, sqrt(1000), log=TRUE))
    beta.prior <- beta.prior + sum(dnorm(beta[,-1], beta[,-Data$T],
        matrix(sigma[2:3], 2, Data$T-1), log=TRUE))
    zeta.prior <- dmvn(zeta, rep(0,Data$S), Sigma[ , ,1], log=TRUE)
    phi.prior[1] <- dtrunc(phi[1], "norm", a=0, b=Inf, mean=0,
        sd=sqrt(1000), log=TRUE)
```

```
    phi.prior[-1] <- dtrunc(phi[-1], "norm", a=0, b=Inf,
         mean=phi[-Data$T], sd=sigma[4], log=TRUE)
    kappa.prior[1] <- dtrunc(kappa[1], "norm", a=0, b=Inf, mean=0,
         sd=sqrt(1000), log=TRUE)
    kappa.prior[-1] <- dtrunc(kappa[-1], "norm", a=0, b=Inf,
         mean=kappa[-Data$T], sd=sigma[5], log=TRUE)
    lambda.prior[1] <- dtrunc(lambda[1], "norm", a=0, b=Inf, mean=0,
         sd=sqrt(1000), log=TRUE)
    lambda.prior[-1] <- dtrunc(lambda[-1], "norm", a=0, b=Inf,
         mean=lambda[-Data$T], sd=sigma[6], log=TRUE)
    sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
    ### Log-Likelihood
    mu <- Theta <- matrix(zeta, Data$S, Data$T)
    mu[,1] <- as.vector(tcrossprod(beta[,1], Data$X))
    for (t in 2:Data$T) {
         mu[,t] <- as.vector(tcrossprod(beta[,t], Data$X))
         for (s in 1:Data$S) {
              Theta[,t] <- Sigma[,s,t] / sum(Sigma[,s,t]) * Theta[,t-1]}}
    mu <- mu + Theta
    LL <- sum(dnorm(Data$Y, mu, sigma[1], log=TRUE))
    ### Log-Posterior
    LP <- LL + beta.prior + zeta.prior + sum(phi.prior) +
         sum(kappa.prior) + sum(lambda.prior) + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma),
         yhat=mu, parm=parm)
    return(Modelout)
}
```

52. 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 28. This example uses a nonseparable, stationary covariance function in which space and time are separable only when $\psi = 0$. To extend this to a large space-time data set, consider incorporating the predictive process kriging example in section 29.

52.1. Form

$$\mathbf{Y}_{s,t} \sim \mathcal{N}(\mu_{s,t}, \sigma_1^2), \quad s = 1, \ldots, S, \quad t = 1, \ldots, T$$
$$\mu = \mathbf{X}\beta + \Xi$$
$$\Xi \sim \mathcal{N}_{ST}(\Xi_\mu, \Sigma)$$

$$\Sigma = \sigma_2^2 \exp\left(-\frac{\mathbf{D}_S}{\phi_1}^\kappa - \frac{\mathbf{D}_T}{\phi_2}^\lambda - \psi\frac{\mathbf{D}_S}{\phi_1}^\kappa \frac{\mathbf{D}_T}{\phi_2}^\lambda\right)$$

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

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

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

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

$$\Xi_\mu = 0$$

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

52.2. 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(tcrossprod(beta, 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)
```

52.3. Initial Values

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

52.4. 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(tcrossprod(beta, 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)
}
```

53. 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 $s = 1, \dots, S$ points in space and measurements are taken across time-periods $t = 1, \dots, T$ 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 28. This example uses separable space-time covariances, which is more convenient but usually less appropriate than a nonseparable covariance function. To extend this to a large space-time data set, consider incorporating the predictive process kriging example in section 29.

53.1. Form

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

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

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

$$\theta \sim \mathcal{N}_N(\theta_\mu, \Sigma_T)$$
$$\Sigma_T = \sigma_3^2 \exp(-\phi_2 \mathbf{D}_T)^\lambda$$
$$\zeta \sim \mathcal{N}_N(\zeta_\mu, \Sigma_S)$$
$$\Sigma_S = \sigma_2^2 \exp(-\phi_1 \mathbf{D}_S)^\kappa$$
$$\beta_j \sim \mathcal{N}(0, 1000), \quad j = 1, \dots, 2$$
$$\sigma_k \sim \mathcal{HC}(25), \quad k = 1, \dots, 3$$
$$\phi_k \sim \mathcal{U}(1, 5), \quad k = 1, \dots, 2$$
$$\zeta_\mu = 0$$
$$\theta_\mu = 0$$
$$\kappa = 1, \quad \lambda = 1$$

53.2. 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(tcrossprod(beta, X))
Y <- mu + zeta + Theta + matrix(rnorm(S*T,0,0.1),S,T)
mon.names <- c("LP","sigma[1]","sigma[2]","sigma[3]")
parm.names <- parm.names(list(zeta=rep(0,S), theta=rep(0,T),
     beta=rep(0,2), phi=rep(0,2), log.sigma=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)
```

53.3. Initial Values

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

53.4. Model

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

```
### Hyperparameters
zeta.mu <- rep(0,Data$S)
theta.mu <- rep(0,Data$T)
### Parameters
beta <- parm[grep("beta", Data$parm.names)]
zeta <- parm[grep("zeta", Data$parm.names)]
theta <- parm[grep("theta", Data$parm.names)]
kappa <- 1; lambda <- 1
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
phi <- interval(parm[grep("phi", Data$parm.names)], 1, 5)
parm[grep("phi", Data$parm.names)] <- phi
Sigma.S <- sigma[2]^2 * exp(-phi[1] * Data$D.S)^kappa
Sigma.T <- sigma[3]^2 * exp(-phi[2] * Data$D.T)^lambda
### Log(Prior Densities)
beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
zeta.prior <- dmvn(zeta, zeta.mu, Sigma.S, log=TRUE)
theta.prior <- dmvn(theta, theta.mu, Sigma.T, log=TRUE)
sigma.prior <- sum(dhalfcauchy(25, 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(tcrossprod(beta, Data$X)) + zeta + Theta
LL <- sum(dnorm(Data$Y, mu, sigma[1], log=TRUE))
### Log-Posterior
LP <- LL + beta.prior + zeta.prior + theta.prior + sigma.prior +
     phi.prior
Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,sigma),
     yhat=mu, parm=parm)
return(Modelout)
}
```

54. 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.

54.1. Form

$$\mathbf{y}_i \sim \mathcal{WEIB}(\gamma, \mu_i), \quad i = 1, \ldots, N$$

$$\mu = \exp(\mathbf{X}\beta)$$

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

$$\gamma \sim \mathcal{G}(1, 0.001)$$

54.2. 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(tcrossprod(beta, 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)
```

54.3. Initial Values

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

54.4. Model

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

55. 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 $\beta_\sigma = 0.1$, 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 variance 1000. 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.

55.1. Form

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

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

$$(\beta_j|\gamma_j) \sim (1-\gamma_j)\mathcal{N}(0, 0.01) + \gamma_j\mathcal{N}(0, \beta_\sigma^2) \quad j = 1, \dots, J$$

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

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

$$\delta_j \sim \mathcal{N}(0, 10) \in [-100, 100], \quad j = 1, \dots, J$$

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

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

55.3. Initial Values

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

55.4. Model

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

```
### Hyperparameters
beta.sigma <- exp(parm[grep("log.beta.sigma", Data$parm.names)])
delta <- interval(parm[grep("delta", Data$parm.names)],-100,100)
parm[grep("delta", Data$parm.names)] <- delta
### Parameters
beta <- parm[1:Data$J]
gamma <- round(invlogit(delta))
beta.sigma <- ifelse(gamma == 0, 0.1, beta.sigma)
sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
### Log(Hyperprior and Prior Densities)
beta.prior <- sum(dnorm(beta, 0, beta.sigma, log=TRUE))
beta.sigma.prior <- sum(dhalfcauchy(beta.sigma, 25, log=TRUE))
delta.prior <- sum(dtrunc(delta, "norm", a=-100, b=100,
     mean=logit(1/Data$J), sd=sqrt(1000), log=TRUE))
sigma.prior <- dhalfcauchy(sigma, 25, log=TRUE)
### Log-Likelihood
mu <- tcrossprod(beta, 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.sigma),
     sigma, gamma), yhat=mu, parm=parm)
return(Modelout)
}
```

56. Vector Autoregression, VAR(1)

56.1. Form

$$\mathbf{Y}_{t,j} \sim \mathcal{N}(\mu_{t,j}, \sigma_j^2), \quad t = 1, \ldots, T, \quad j = 1, \ldots, J$$

$$\mu_{t,j} = \alpha_j + \Phi_{1:J,j} \mathbf{Y}_{t-1,j}$$

$$\mathbf{y}_j^{new} = \alpha_j + \Phi_{1:J,j} \mathbf{Y}_{T,j}$$

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

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

$$\Phi_{i,k} \sim \mathcal{N}(0, 1000), \quad i = 1, \ldots, J, \quad k = 1, \ldots, J$$

56.2. Data

```
T <- 100
J <- 3
Y <- matrix(0,T,J)
```

```
for (j in 1:J) {for (t in 2:T) {
    Y[t,j] <- Y[t-1,j] + rnorm(1,0,0.1)}}
mon.names <- c("LP", parm.names(list(ynew=rep(0,J))))
parm.names <- parm.names(list(alpha=rep(0,J), Phi=matrix(0,J,J),
    log.sigma=rep(0,J)))
MyData <- list(J=J, T=T, Y=Y, mon.names=mon.names, parm.names=parm.names)
```

56.3. Initial Values

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

56.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1:Data$J]
    Phi <- matrix(parm[grep("Phi", Data$parm.names)], Data$J, Data$J)
    sigma <- exp(parm[grep("log.sigma", Data$parm.names)])
    ### Log(Prior Densities)
    alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
    Phi.prior <- sum(dnorm(Phi, 0, sqrt(1000), log=TRUE))
    sigma.prior <- sum(dhalfcauchy(sigma, 25, log=TRUE))
    ### Log-Likelihood
    mu <- matrix(alpha,Data$T,Data$J,byrow=TRUE)
        mu[-1,] <- mu[-1,] + t(tcrossprod(Phi,Data$Y[-Data$T,]))
    ynew <- alpha + as.vector(crossprod(Phi, Data$Y[Data$T,]))
    LL <- sum(dnorm(Data$Y, mu,
        matrix(sigma,Data$T,Data$J,byrow=TRUE), log=TRUE))
    ### Log-Posterior
    LP <- LL + alpha.prior + Phi.prior + sigma.prior
    Modelout <- list(LP=LP, Dev=-2*LL, Monitor=c(LP,ynew), yhat=mu,
        parm=parm)
    return(Modelout)
    }
```

57. Zero-Inflated Poisson (ZIP)

57.1. Form

$$\mathbf{y} \sim \mathcal{P}(\Lambda_{1:N,2})$$

$$\mathbf{z} \sim \mathcal{BERN}(\Lambda_{1:N,1})$$

$$\mathbf{z}_i = \begin{cases} 1 & \text{if } \mathbf{y}_i = 0 \\ 0 & \end{cases}$$

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

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

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

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

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

57.2. 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(tcrossprod(alpha, X1) + rnorm(N,0,0.1)))
mu <- as.vector(round(exp(tcrossprod(beta, 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)
```

57.3. Initial Values

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

57.4. Model

```
Model <- function(parm, Data)
    {
    ### Parameters
    alpha <- parm[1:Data$J1]
    beta <- parm[grep("beta", Data$parm.names)]
    ### Log(Prior Densities)
    alpha.prior <- sum(dnorm(alpha, 0, sqrt(1000), log=TRUE))
    beta.prior <- sum(dnorm(beta, 0, sqrt(1000), log=TRUE))
    ### Log-Likelihood
    Lambda <- matrix(NA, Data$N, 2)
    Lambda[,1] <- invlogit(tcrossprod(alpha, Data$X1))
```

```
Lambda[,2] <- exp(tcrossprod(beta, 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 + alpha.prior + 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.

Crainiceanu C, Ruppert D, Wand M (2005). "Bayesian Analysis for Penalized Spline Regression Using WinBUGS." *Journal of Statistical Software*, **14**(14), 1–24.

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 (2011). ***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 11.09.12, URL http://cran.r-project.org/web/packages/LaplacesDemon/index.html.

Kleine L (1950). *Economic Fluctuations in the United States 1921-1940*. John Wiley & Sons, New York, New York.

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