CheeseData

Rohit Arora

1. Setup

The data consists of quantities sold and the prices for 88 stores selling cheese. In addition we have a store specific display variable indicating if the cheese was on display. We start of with the following model

$$y_i = X_i\beta + Z_i\gamma_i + \epsilon_i$$

where the coefficients β correspond to the fixed effect across stores and γ_i correspond to the store specific random effects. To setup a fully bayesian model we assume the following set of priors on the parameters in the model

$$\begin{aligned}
y_i|\beta_i, \gamma_i, \sigma^2, \Sigma &\sim \mathcal{N}\left(X\beta + Z_i\gamma_i, \sigma^2 I\right) \\
\beta &\propto 1 \\
\gamma_i|\Sigma &\sim \mathcal{N}\left(0, \Sigma\right) \\
\Sigma &\sim \mathcal{IW}\left(d_0, C_0\right) \\
\sigma^2 &\propto \frac{1}{\sigma^2}
\end{aligned}$$

β is assumed to have a flat prior. γ_i are considered to have a Jeffrey's prior. Other prior's are standard.

2. Gibbs Updates

Below we specify the posterior updates for the parameters in our model. n_s is the number of stores, n_t is the number of observations per store and n is the total number of stores

- posterior for for fixed effect β

$$\begin{aligned}
p\left(\beta|\mathbf{y}, \gamma, \sigma^2, \Sigma; \Theta\right) &\propto p\left(\beta\right)\prod_{i=1}^{n_s} p\left(\mathbf{y}_i|\beta, \gamma_i, \sigma^2\right) \\
&\propto 1 \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n_s}\sum_{t=1}^{n_t}\left(y_{it} - x_{it}^\top\beta - z_{it}^\top\gamma_i\right)^2\right) \\
&= \exp\left(-\frac{1}{2}\sum_{i=1}^{n_s}\left(\mathbf{y}_i - X_i\beta - Z_i\gamma_i\right)^\top\left(\sigma^2 I\right)^{-1}\left(\mathbf{y}_i - X_i\beta - Z_i\gamma_i\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\beta^\top\left(\sigma^{-2}\sum_{i=1}^{n_s}X_i^\top X_i\right)\beta - 2\left(\sum_{i=1}^{n_s}\sigma^{-2}(\mathbf{y}_i - Z_i\gamma_i)^\top X_i\right)\beta\right)\right) \\
&\sim \mathcal{N}\left(\left(\sum_{i=1}^{n_s}X_i^\top X_i\right)^{-1}\left(\sum_{i=1}^{n_s}\left(\mathbf{y}_i - Z_i\gamma_i\right)^\top X_i\right), \left(\sigma^{-2}\sum_{i=1}^{n_s}X_i^\top X_i\right)^{-1}\right)
\end{aligned}$$

- posterior update for random effect γ_i

$$
\begin{aligned}
p\left(\gamma_i|\mathbf{y}_i,\sigma^2,\beta,\Sigma;\Theta\right) &\propto p\left(\mathbf{y}_i|\gamma_i,\beta,\sigma^2\right)p\left(\gamma_i|\Sigma\right)\\
\propto \exp\left(-\frac{1}{2\sigma^2}\sum_{t=1}^{n_i}\left(y_{it}-x_{it}^\top\beta-z_{it}^\top\gamma_i\right)^\top\left(y_{it}-x_{it}^\top\beta-z_{it}^\top\gamma_i\right)\right)&\exp\left(-\frac{1}{2}\gamma_i^\top\Sigma^{-1}\gamma_i\right)\\
=\exp\left(-\frac{1}{2\sigma^2}(\mathbf{y}_i-X_i\beta-Z_i\gamma_i)^\top\left(\mathbf{y}_i-X_i\beta-Z_i\gamma_i\right)\right)&\exp\left(-\frac{1}{2}\gamma_i^\top\Sigma^{-1}\gamma_i\right)\\
=\exp\left(-\frac{1}{2}\left((\mathbf{y}_i-X_i\beta-Z_i\gamma_i)^\top\left(\sigma^2I\right)^{-1}\left(\mathbf{y}_i-X_i\beta-Z_i\gamma_i\right)+\gamma_i^\top\Sigma^{-1}\gamma_i\right)\right)&\\
\propto \exp\left(-\frac{1}{2}\left(-2\sigma^{-2}(\mathbf{y}_i-X_i\beta)^\top Z_i\gamma_i+\sigma^{-2}\gamma_i^\top Z_i^\top Z_i\gamma_i+\gamma_i^\top\Sigma^{-1}\gamma_i\right)\right)&\\
=\exp\left(-\frac{1}{2}\left(\gamma_i^\top\left(\sigma^{-2}Z_i^\top Z_i+\Sigma^{-1}\right)\gamma_i-2\sigma^{-2}(\mathbf{y}_i-X_i\beta)^\top Z_i\gamma_i\right)\right)&\\
\sim\mathcal{N}\left(\left(\sigma^{-2}Z_i^\top Z_i+\Sigma^{-1}\right)^{-1}\sigma^{-2}(\mathbf{y}_i-X_i\beta)^\top Z_i,\left(\sigma^{-2}Z_i^\top Z_i+\Sigma^{-1}\right)^{-1}\right)&
\end{aligned}
$$

- posterior update for σ^2

$$
\begin{aligned}
p\left(\sigma^2|\mathbf{y},\gamma,\Sigma;\Theta\right)\propto p\left(\mathbf{y}|\gamma,\sigma^2\right)p\left(\sigma^2\right)&=p\left(\sigma^2\right)\prod_{i=1}^{n_s}p\left(\mathbf{y}_i|\gamma_i,\sigma^2\right)\\
\propto \sigma^{-2\frac{(n+1)}{2}}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n_s}\sum_{t=1}^{n_t}\left(y_{it}-z_{it}^\top\gamma_i\right)^\top\left(y_{it}-z_{it}^\top\gamma_i\right)\right)&\\
\sim \mathrm{I}\Gamma\left(\frac{n}{2},\frac{1}{2}\sum_{i=1}^{n_s}\sum_{t=1}^{n_t}\left(y_{it}-x_{it}^\top\beta-z_{it}^\top\gamma_i\right)^\top\left(y_{it}-x_{it}^\top\beta-z_{it}^\top\gamma_i\right)\right)&
\end{aligned}
$$

- posterior update for Σ

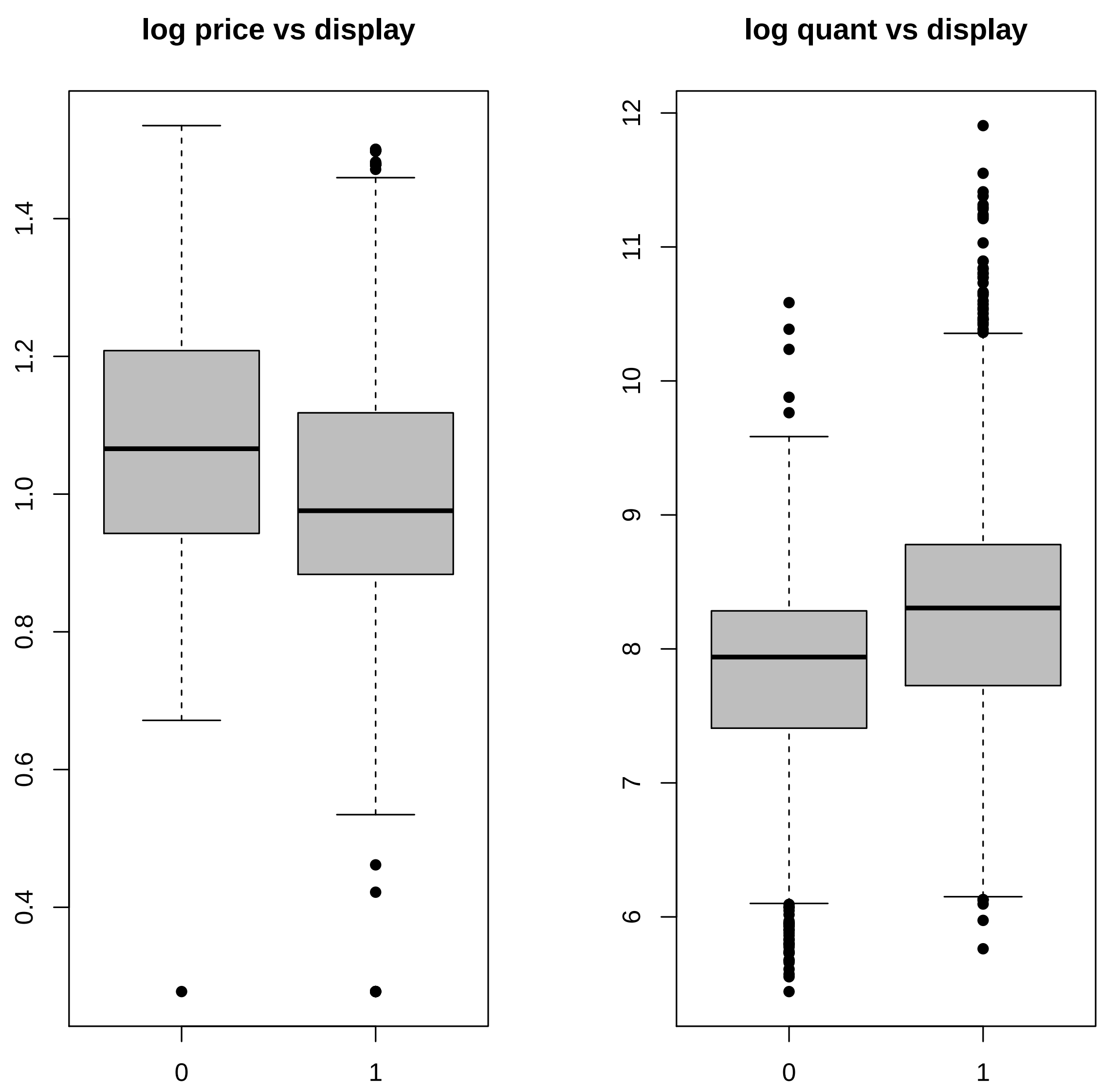
$$
p\left(\Sigma|\mathbf{y},\gamma,\sigma^2;\Theta\right)\propto\mathcal{IW}\left(d_0+n_s,C_0+\sum_{i=1}^{n_s}\gamma_i\gamma_i^\top\right)
$$

3. Hierarchical Model using LME4

```
library(lme4)
library(mosaic)
library(dplyr)
```

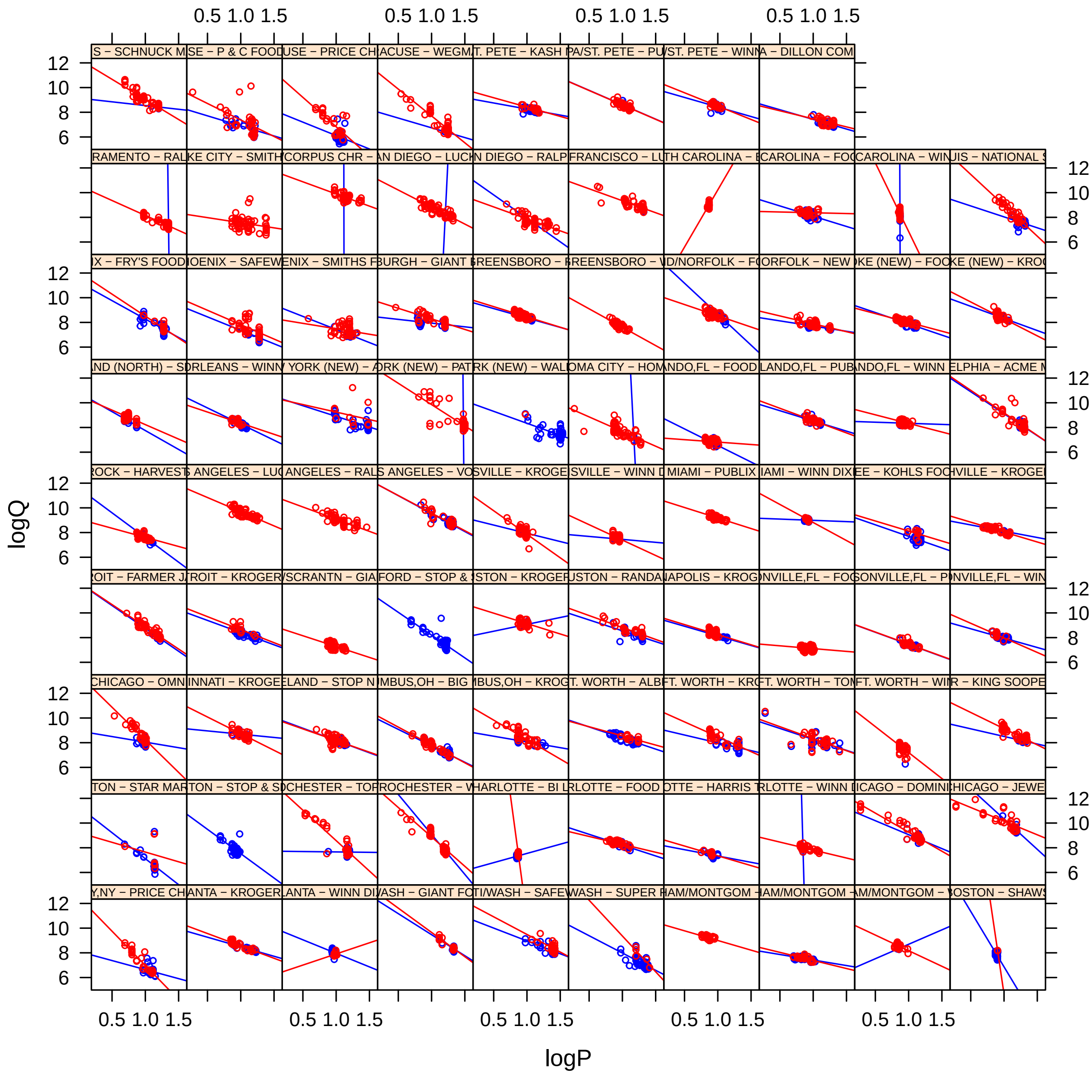
Below we will first examine a few plots to check if presence or absence of display, affects the price of cheese. Then we will fit a simple hierarchical model.

```r
data <- read.csv('cheese.csv')

modData <- data %>%
  mutate(logP = log(price), logQ = log(vol)) %>%
  dplyr::select(-one_of(c("price","vol"))) %>%
  as.data.frame()

par(mfrow = c(1,2))
boxplot(logP ~ disp, data = modData, col = 'gray', pch = 16, main="log price vs display")
boxplot(logQ ~ disp, data = modData, col = 'gray', pch = 16, main="log quant vs display")
```




```r
par(mfrow = c(1,1))
```

```r
xyplot(logQ ~ logP | store, data = modData, group = disp, col = c('blue','red'),
       col.line = c('blue','red'), cex = 0.5, type = c("r", "p"),
       par.strip.text=list(cex=.5))
```



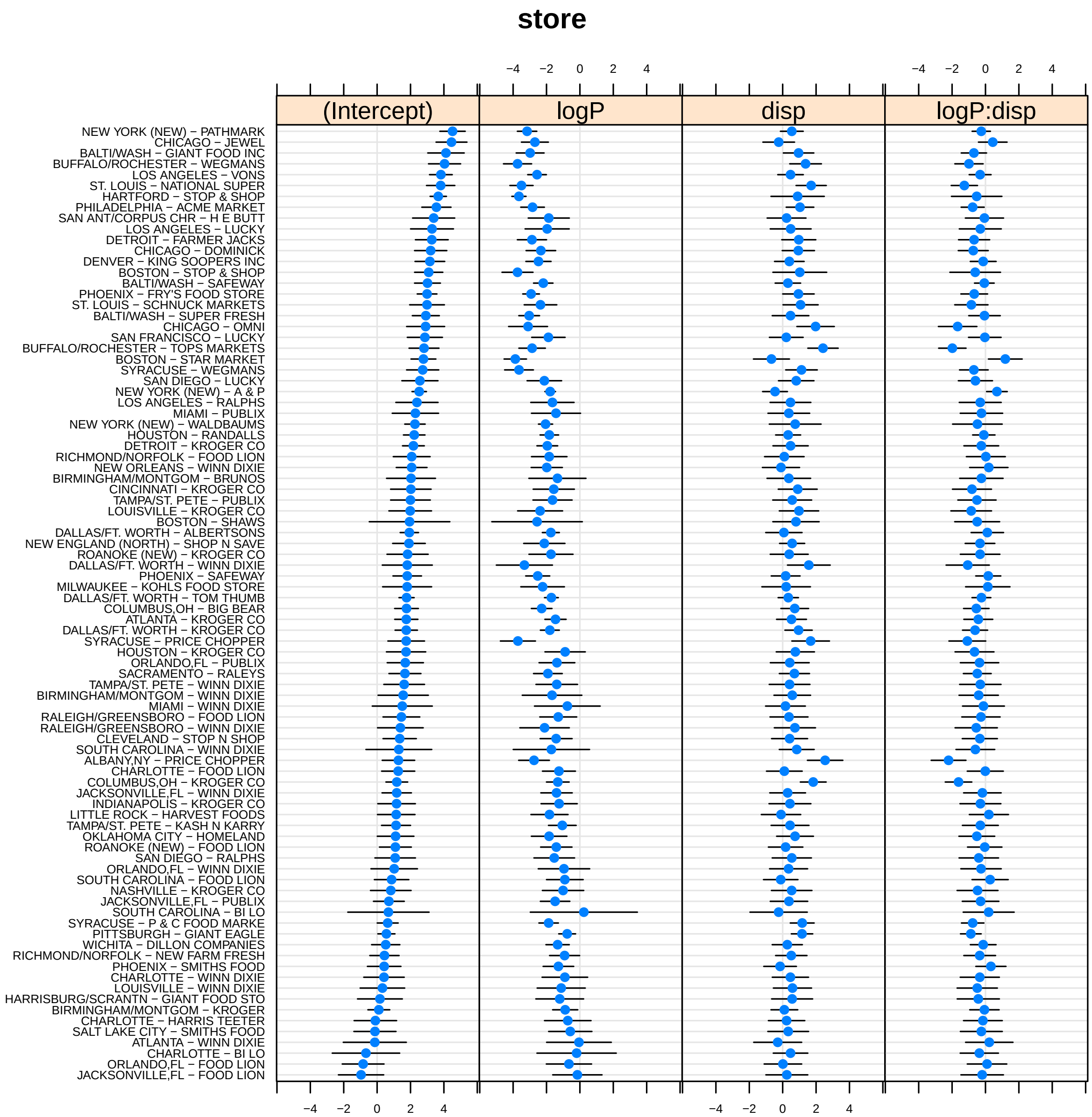

```r
hlm <- lmer(logQ ~  (1 + logP + disp + disp:logP | store), data = modData)
summary(hlm)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: logQ ~ (1 + logP + disp + disp:logP | store)
##    Data: modData
##
## REML criterion at convergence: 1811.9
##
```

```
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.0245 -0.4898 -0.0317  0.4348 12.2358
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  store    (Intercept) 5.0478   2.2467
##           logP        4.6658   2.1600   -0.94
##           disp        0.9634   0.9815    0.47 -0.55
##           logP:disp   0.7004   0.8369   -0.32  0.38 -0.97
##  Residual             0.0675   0.2598
## Number of obs: 5555, groups:  store, 88
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  8.18711    0.07794     105
```

```r
resid <- ranef(hlm, condVar = T)
dotplot(resid, scales=list(cex=c(.5,.5)),layout=c(4,1), main=T,
        main.title='Random Effects by by Store')
```

```
## $store
```



The necessity of using a hierarchical model is evident from boxplots and matrix plots. Some of the groups have very few observations corresponding to the presence/absence of advertising. Not pooling among the two groups can lead to biased estimates. Below we will try to load the data and fit a hierarchical bayesian model to the cheese dataset.

4. Gibbs Sampler

Below is the main function for the Gibbs Sampler. We start with β and then sequentially update in the following order γ, σ^2 and Σ.

```
library(dplyr)
library(mvtnorm)
```

```r
library(MCMCpack)
library(robustbase)
```

```r
gibbsCheese <- function(y, X, Z, allStores, d, C, iter= 1000, burn=2, thin=2) {

  n <- length(y); p <- ncol(Z); nstores <- nlevels(allStores);

  #setup the structure for the chain
  beta <- array(0, dim=c(ncol(X),iter)); gamma <- array(0,dim=c(nstores, ncol(Z), iter));
  Sigma <- array(0, dim=c(p, p, iter)); sig.sq <- rep(0,iter)

  #initialize the chain
  beta[,1] <- rep(0, p) ;gamma[,,1] <- rep(0, nstores*p);  Sigma[,,1] <- diag(p)
  sig.sq[1] <- 1

  pb <- txtProgressBar(min = 2, max = iter, style = 3, file = stderr())

  for (iter in 2:iter) {

    # Update fixed effect for beta
    beta.post.var <- solve(
      sig.sq[iter-1]^-1 * Reduce('+',
        lapply(1:nstores, function(ns) {
          bStore <- which(ns==as.integer(allStores));
          Xi <- X[bStore,,drop=FALSE]; crossprod(Xi)
        })
      ))

    beta.post.mean <- (beta.post.var/sig.sq[iter-1]) %*% t(Reduce('+',
      lapply(1:nstores, function(ns) {
        bStore <- which(ns==as.integer(allStores));
        Zi <-  Z[bStore,]; yi <- y[bStore]; Xi <- X[bStore,,drop=FALSE]
        t(yi - Zi %*% gamma[ns,,iter-1]) %*% Xi
      })
    ))

    beta[,iter] <- rmvnorm(1, beta.post.mean, beta.post.var)

    # Update random effect gamma for each store
    Sig.inv <- solve(Sigma[,,iter-1])

    gamma[,,iter] <-
      t(sapply(1:nstores, function(ns) {

        bStore <- which(ns==as.integer(allStores))
        Xi <- X[bStore,,drop=FALSE]; Zi <-  Z[bStore,]; yi <- y[bStore]

        gamma.post.var <- solve((sig.sq[iter-1])^-1 * crossprod(Zi) + Sig.inv)
        gamma.post.mean <-
          gamma.post.var %*%
          (sig.sq[iter-1]^-1 * crossprod(Zi, yi - Xi %*% beta[,iter,drop=FALSE]))

        rmvnorm(1, gamma.post.mean, gamma.post.var)
```

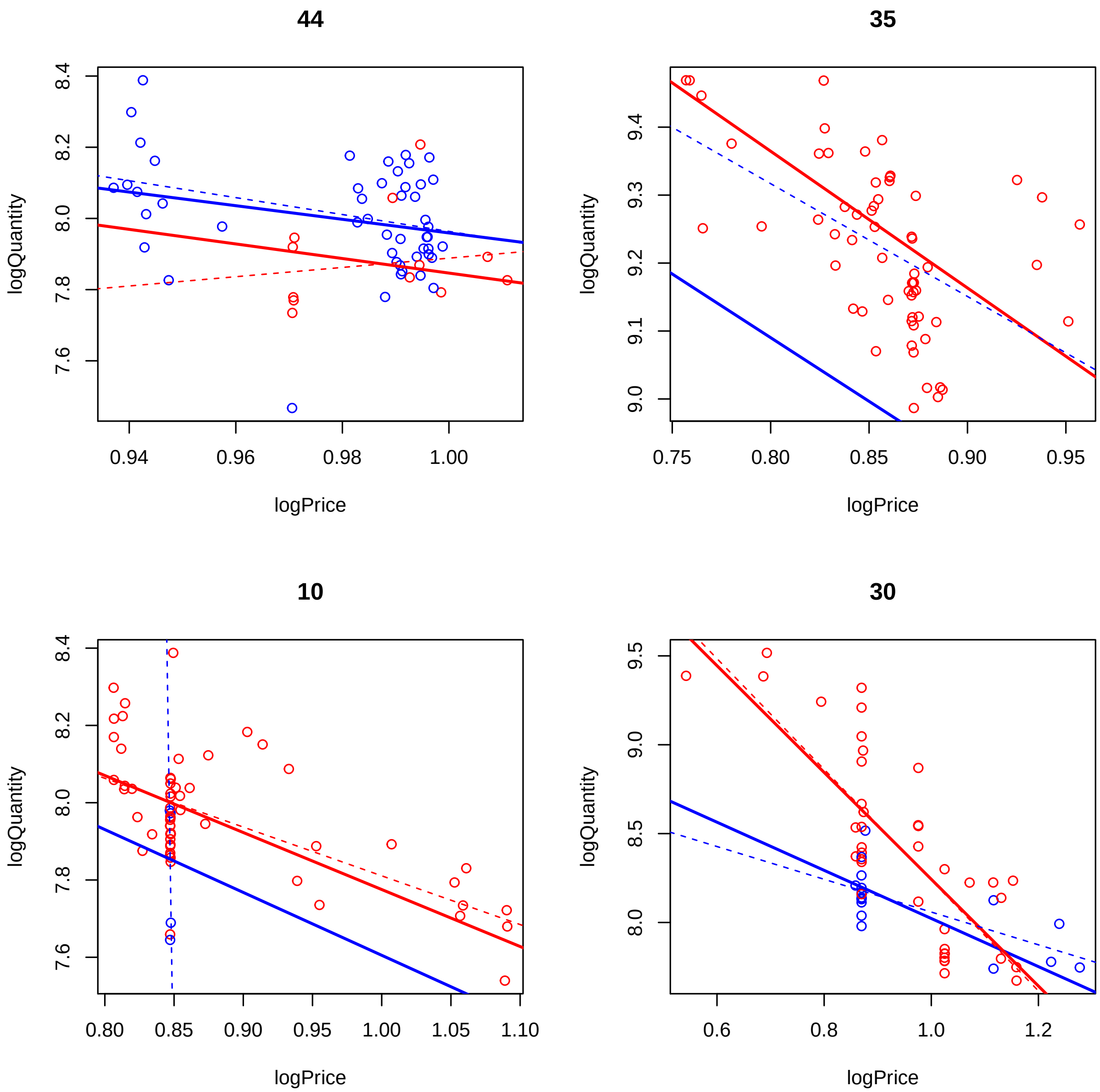
```r
    }))


    # Update sig.sq

    SS <- sum(
      sapply(1:nstores, function(ns) {
        bStore <- which(ns==as.integer(allStores));
        Zi <-  Z[bStore,]; yi <- y[bStore]; Xi <- X[bStore,,drop=FALSE]
        crossprod(yi - Zi %*% gamma[ns,,iter] - Xi %*% beta[,iter,drop=FALSE])
      })
    )

    sig.sq[iter] <- 1/rgamma(1, n/2, SS/2)

    #Update Sigma values for IW
    S <- Reduce('+',lapply(1:nstores, function(ns) {
      tcrossprod(gamma[ns,,iter])
    }))

    Cn = C + S; dn = d + nstores
    Sigma[,,iter] <- riwish(dn, Cn)

    setTxtProgressBar(pb, iter)
  }

  thinseq <- seq(1,iter - 1, by=thin)

  beta <- beta[,-burn]; gamma <- gamma[,,-burn];
  Sigma <- Sigma[,,-burn];   sig.sq <- sig.sq[-burn]

  beta <- beta[,thinseq]; gamma <- gamma[,,thinseq];
  Sigma <- Sigma[,,thinseq]; sig.sq <- sig.sq[thinseq]

  #posterior medians
  beta.post.median     <- apply(beta, 1, median)
  gamma.post.median    <- apply(gamma, c(1,2), median)
  sig.sq.post.median   <- median(sig.sq)
  Sigma.post.median    <- apply(Sigma, c(1,2), mean)

  list(beta.post = beta.post.median,
             gamma.post = gamma.post.median,
             sig.sq.post = sig.sq.post.median,
             Sigma.post = Sigma.post.median)
}
```

```r
data <- read.csv('cheese.csv')
data <- data %>%
  mutate(logP = log(price), logQ = log(vol), disp = disp) %>%
  as.data.frame() %>%
  dplyr::select(-one_of("vol","price"))

y  <- data$logQ; X <- Z <- model.matrix(logQ ~ 1 + logP + disp + logP:disp, data=data)
```

```r
p <- d <- ncol(Z); C = diag(p)
mcoutput <- gibbsCheese(y, X, Z, data$store, d, C)
```

We will now examine the fit with a few stores. Stores 35 and 10 show some peculiar differences between ordinary regression and our hierarchical model.

```r
cols <- c('blue','red')
xgrid <- seq(min(X[,2]), max(X[,2]), length.out = 50)
stores <- as.integer(data$store)

par(mfrow = c(2,2))

for (i in c(3, 7, 18, 25)){

  plot(X[stores==i,2], y[stores == i], main = as.character(stores[i]),
       col = cols[X[stores==i, "disp"]+1], xlab = "logPrice", ylab="logQuantity")
  lines(xgrid, (mcoutput$beta.post[1] + mcoutput$gamma.post[i,1]) +
          xgrid * (mcoutput$beta.post[2] + mcoutput$gamma.post[i,2]), col = cols[1], lwd = 2)
  lines(xgrid, (mcoutput$beta.post[1] + mcoutput$beta.post[3] +
                  mcoutput$gamma.post[i,1]+ mcoutput$gamma.post[i,3]) +
          xgrid * (mcoutput$beta.post[2] + mcoutput$beta.post[4] +
                     mcoutput$gamma.post[i,2] + mcoutput$gamma.post[i,4]),
        col = cols[2], lwd = 2)

  fit = lmrob(y[stores == i] ~ -1 + X[stores==i,])
  betas.lmrob <- fit$coefficients
  lines(xgrid, (betas.lmrob[1]) + xgrid * (betas.lmrob[2]), col = cols[1], lwd = 1, lty = 2)
  lines(xgrid, (betas.lmrob[1]+ betas.lmrob[3]) + xgrid * (betas.lmrob[2] + betas.lmrob[4]),
        col = cols[2], lwd = 1, lty = 2)
}
```




```r
par(mfrow = c(1,1))
```