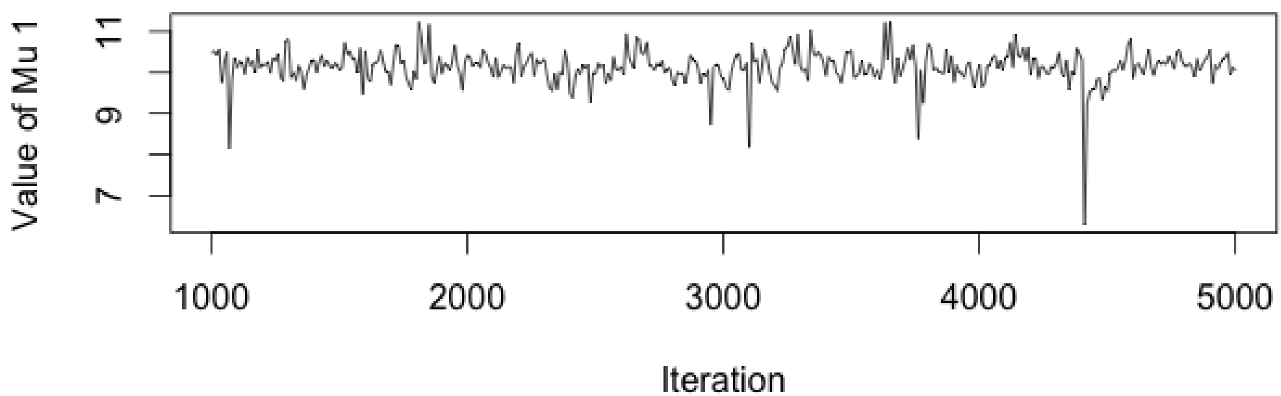
StatMod2 - Hierarchical Models and Shrinkage - Exercises 4

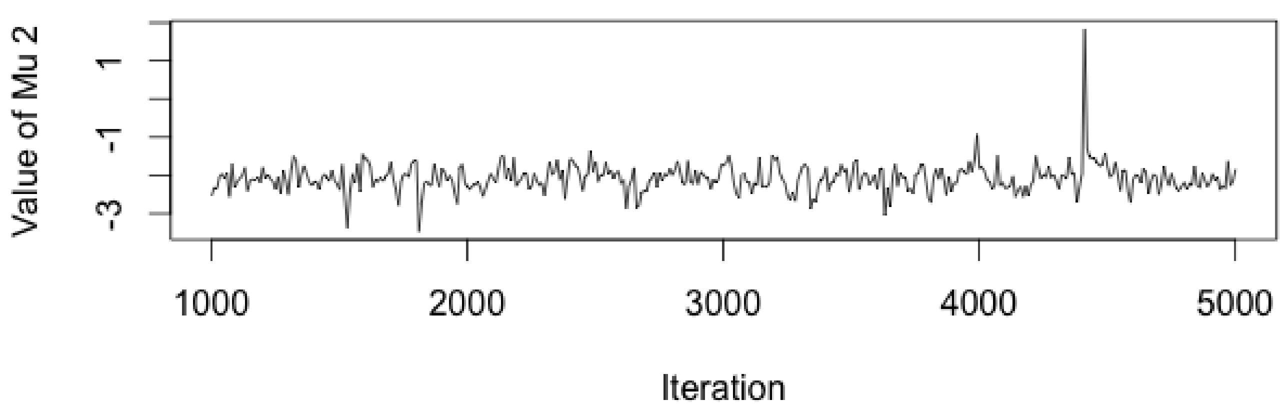
Maurice Diesendruck

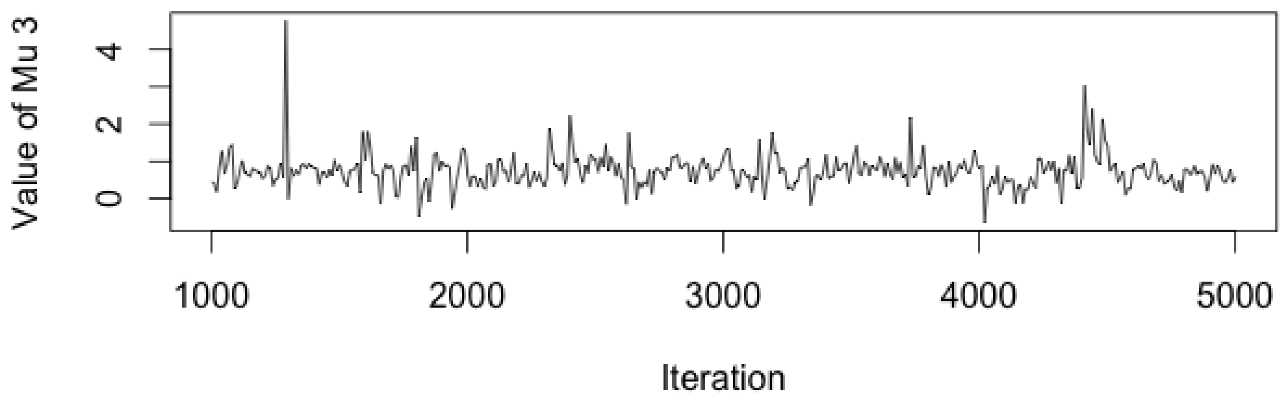
April 14, 2015

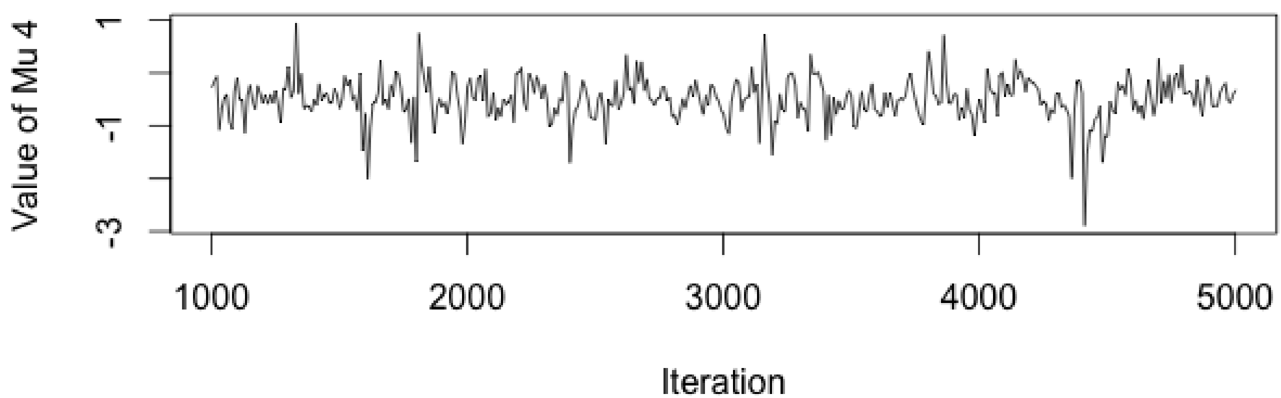
2 Price Elasticity of Demand

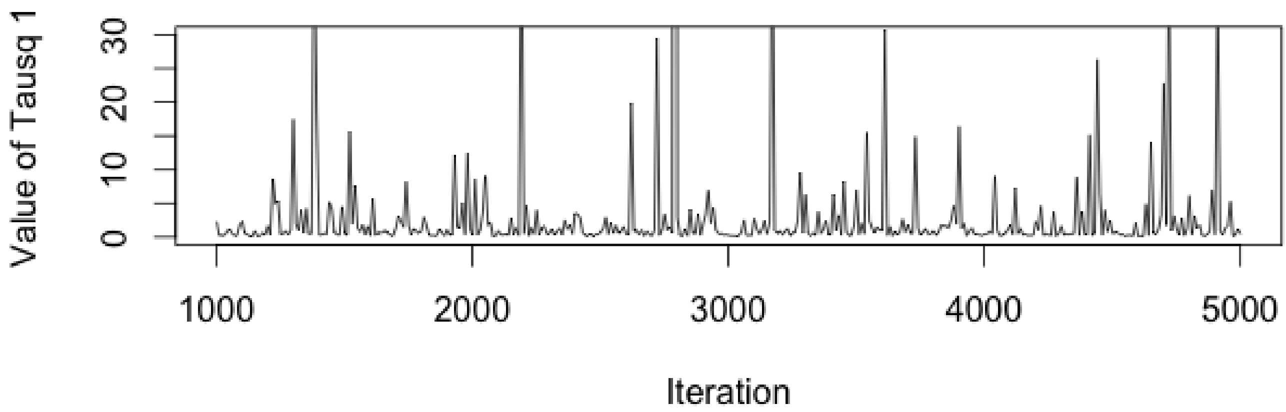
2.1 Bayesian Results for Mu's, TauSq's, and Beta's

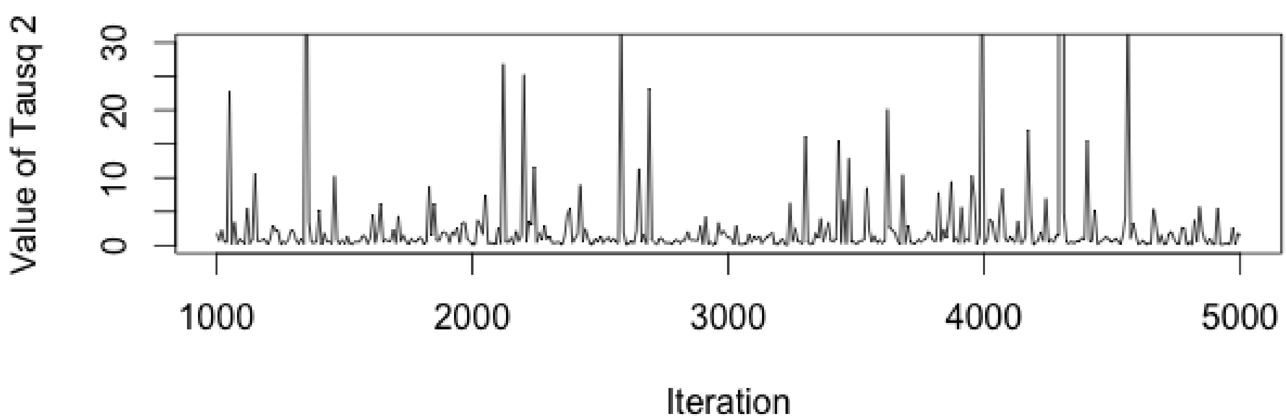


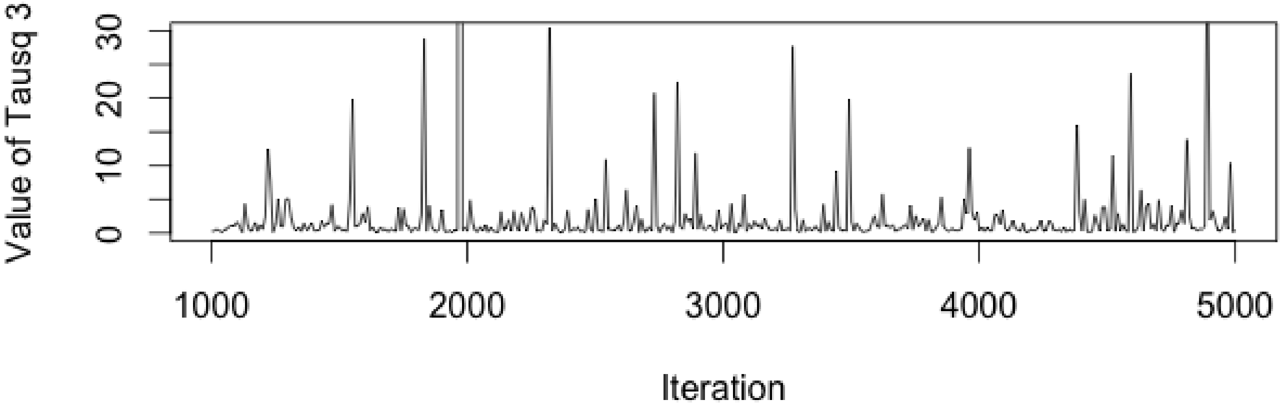



Value of Mu 3
4
2
0
1000
2000
3000
4000
5000
Iteration

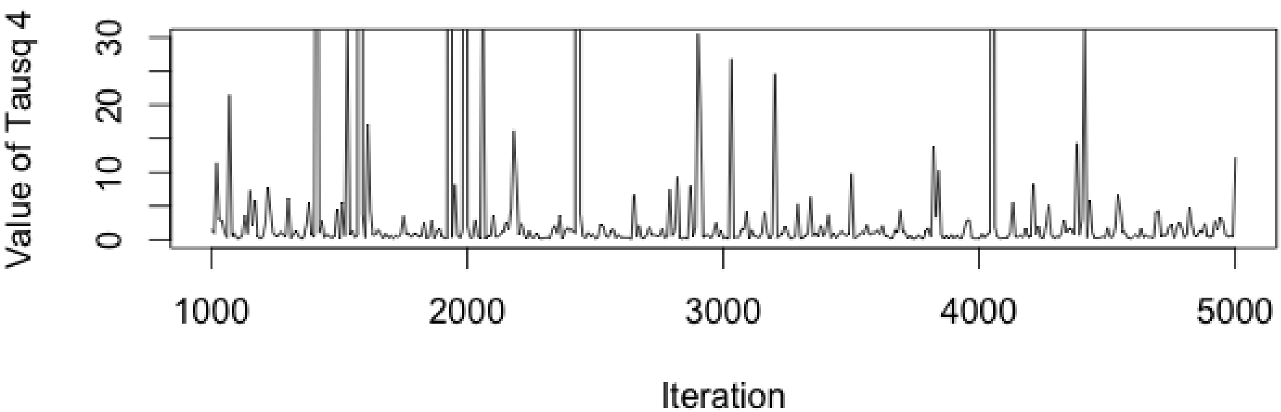

Value of Mu 4
1
-1
-3
1000
2000
3000
4000
5000
Iteration


Value of Tausq 1
0
10
20
30
1000
2000
3000
4000
5000
Iteration

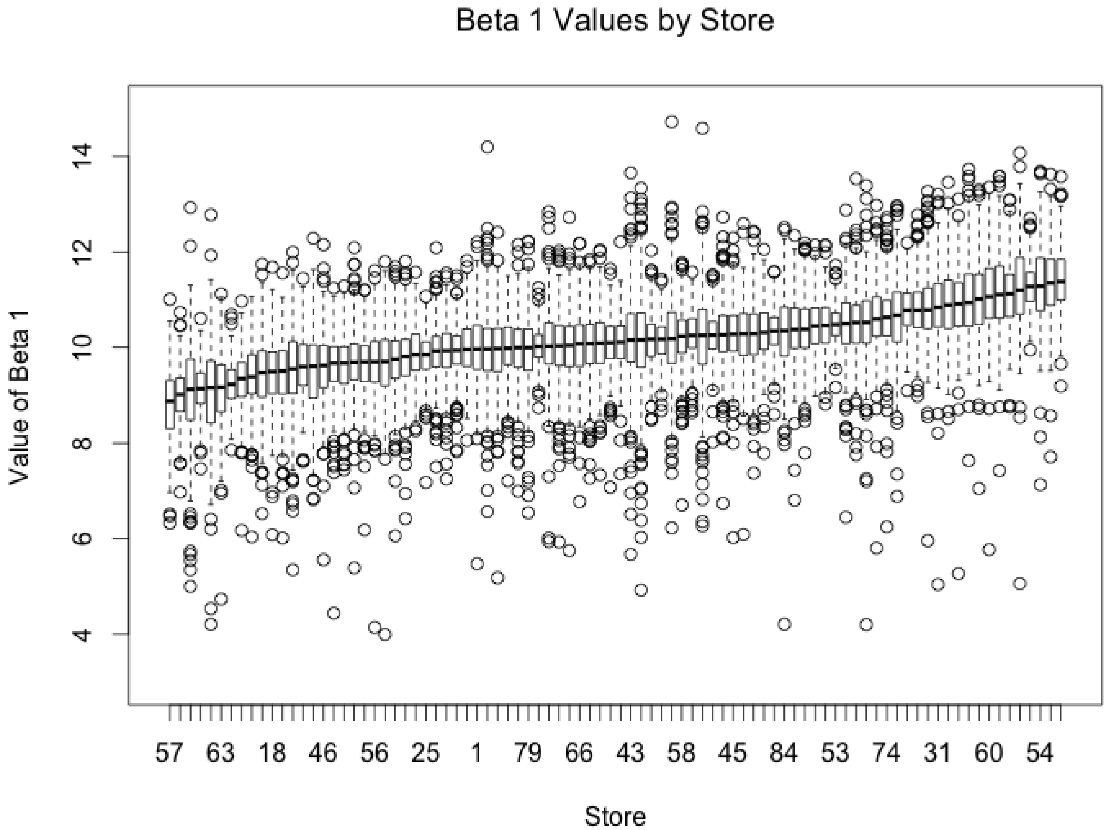

Value of Tausq 2
0
10
20
30
1000
2000
3000
4000
5000
Iteration

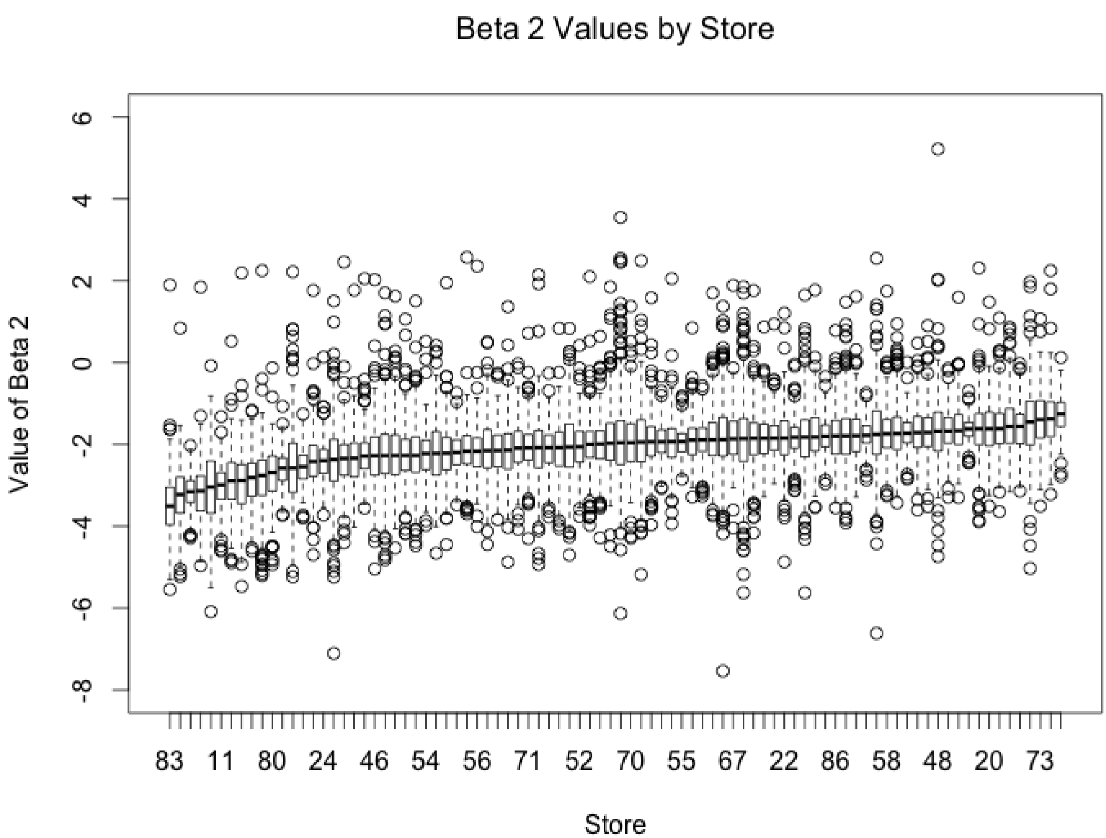


Value of Tausq 3
0
10
20
30
1000
2000
3000
4000
5000
Iteration

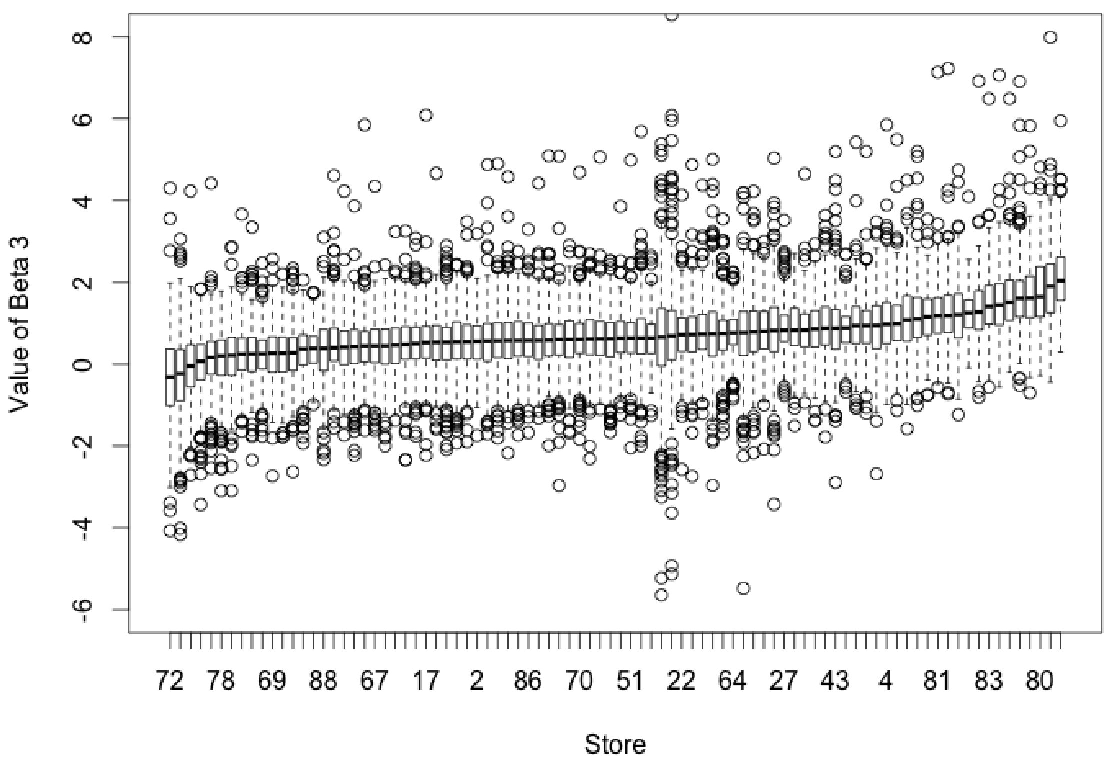


Value of Tausq 4
0
10
20
30
1000
2000
3000
4000
5000
Iteration

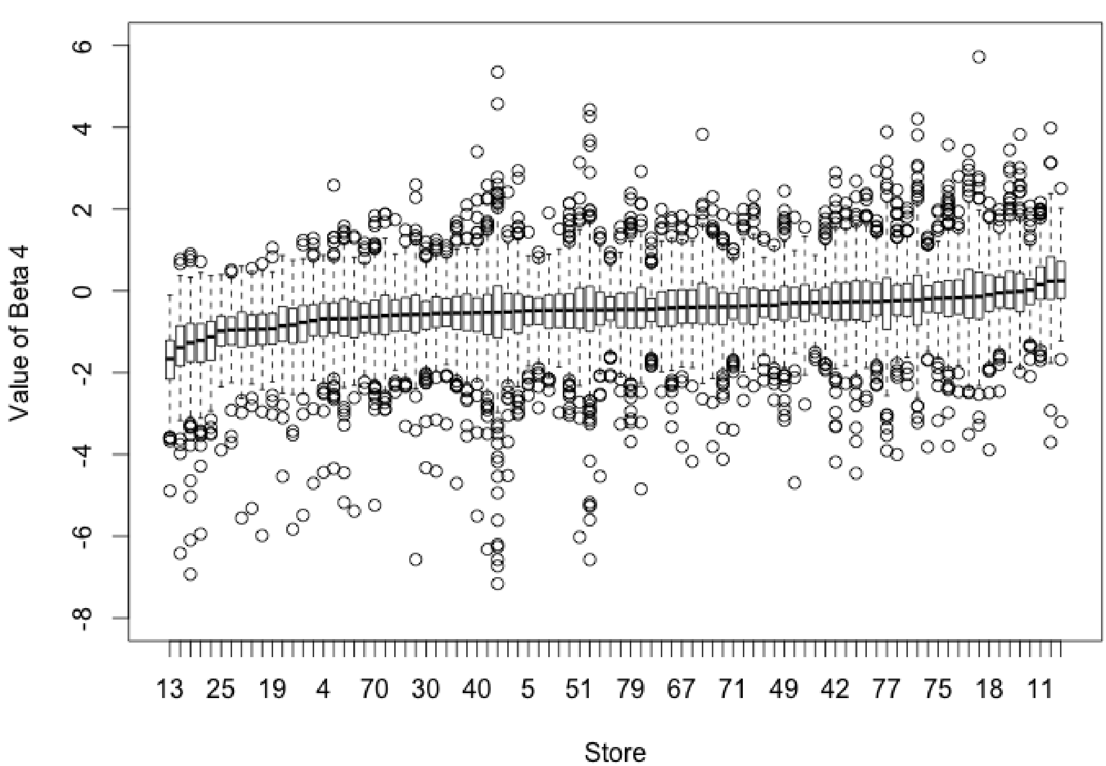

Beta 1 Values by Store
Value of Beta 1
4
6
8
10
12
14
57
63
18
46
56
25
1
79
66
43
58
45
84
53
74
31
60
54
Store


Beta 2 Values by Store
Value of Beta 2
-8
-6
-4
-2
0
2
4
6
83
11
80
24
46
54
56
71
52
70
55
67
22
86
58
48
20
73
Store

Beta 3 Values by Store



Beta 4 Values by Store



2.2 Full R Code

```
# StatMod2 - Cheese - Basic Hierarchical Model Version

library(reshape)
library(ggplot2)

# Fetch and prepare data.
path <- paste("~/Google Drive/2. SPRING 2015/STAT MOD 2 - Prof Scott/",
              "hierarch-shrinkage/cheese", sep="")
setwd(path)
data <- read.csv("cheese.csv")
data$log.Q <- log(data$vol)
data$log.P <- log(data$price)
data$cross <- data$log.P*data$disp
data <- data[,c(1, 5, 6, 4, 7)]
data <- data[order(data[,1]),]
attach(data)

sigma2 <- 1/rgamma(1, 0.5, 0.5)
a0=0.5; b0=0.5; # Hyper params for Tau's, which are ~ InvGamma.
a1=0.5; b1=0.5;
a2=0.5; b2=0.5;
a3=0.5; b3=0.5;
m0=10; v0=1000; # Hyper params for Mu's, which are ~ Normal.
m1=-1; v1=1000;
m2=1; v2=1000;
m3=-1; v3=1000;

Gibbs <- function(data) {
  t0sq <- 1/rgamma(1, a0, b0) # Initialize Tau's.
  t1sq <- 1/rgamma(1, a1, b1)
  t2sq <- 1/rgamma(1, a2, b2)
  t3sq <- 1/rgamma(1, a3, b3)
  mu0 <- rnorm(1, m0, v0) # Initialize Mu's.
  mu1 <- rnorm(1, m1, v1)
  mu2 <- rnorm(1, m2, v2)
  mu3 <- rnorm(1, m3, v3)
  n.iter <- 2000

  # Prepare per-store data.
  store.names <- unique(data$store)
  num.stores <- length(store.names)

  # Create containers for chains and variables.
```

```r
  PARAMS.BY.STORE <- array(rep(NA, num.stores*4*n.iter),
                           c(num.stores, 4, n.iter))
  MU <- matrix(NA, nrow=n.iter+1, ncol=4)
  MU[1,] <- c(mu0, mu1, mu2, mu3)
  TAUSQ <- matrix(NA, nrow=n.iter+1, ncol=4)
  TAUSQ[1,] <- c(t0sq, t1sq, t2sq, t3sq)

  # Do Gibbs many times.

  for (iter in 1:n.iter) {

    # Sample Beta's (be0, be1, be2, be3) for each store.
    # Use Normal-Normal full conditional posterior.
    for (i in 1:num.stores) {
      name <- toString(store.names[i])
      store.data <- data[which(data$store==name),]
      ni <- dim(store.data)[1]
      y <- store.data$log.Q
      X <- as.matrix(cbind(rep(1, ni),
                           store.data[,c("log.P", "disp", "cross")]))
      XtX <- t(X)%*%X
      Xty <- t(X)%*%y
      latest.mu <- MU[iter,]
      diag.tausqs <- diag(c(t0sq, t1sq, t2sq, t3sq))
      b <- sample.beta(sigma2, diag.tausqs, XtX, Xty, latest.mu)
      PARAMS.BY.STORE[i,,iter] <- b
    }

    # Use Normal-Normal full conditional posterior to update Mu's, given Beta's.
    beta.means <- colMeans(PARAMS.BY.STORE[,,iter])
    mu0 <- sample.mu(m0, v0, num.stores, beta.means[1], t0sq)
    mu1 <- sample.mu(m1, v1, num.stores, beta.means[2], t1sq)
    mu2 <- sample.mu(m2, v2, num.stores, beta.means[3], t2sq)
    mu3 <- sample.mu(m3, v3, num.stores, beta.means[4], t3sq)
    MU[iter+1,] <- c(mu0, mu1, mu2, mu3)

    # Use Normal-InvGamma full conditional posterior to update Tau's, given Beta's.
    t0sq <- sample.tausq(beta.means[1], MU[iter,][1], a0, b0)
    t1sq <- sample.tausq(beta.means[2], MU[iter,][2], a1, b1)
    t2sq <- sample.tausq(beta.means[3], MU[iter,][3], a2, b2)
    t3sq <- sample.tausq(beta.means[4], MU[iter,][4], a3, b3)
    TAUSQ[iter+1,] <- c(t0sq, t1sq, t2sq, t3sq)
  }

  return (list(PARAMS.BY.STORE=PARAMS.BY.STORE, MU=MU, TAUSQ=TAUSQ))
}
```

```
sample.beta <- function(sigma2, diag.tausqs, XtX, Xty, latest.mu) {
  cov <- ginv(XtX/sigma2 + solve(diag.tausqs))
  mean <- cov%*%(Xty/sigma2 + solve(diag.tausqs)%*%latest.mu)
  beta <- mvrnorm(1, mu=mean, Sigma=cov)
  return (beta)
}
sample.mu <- function(mu.pr.mean, mu.pr.var, num.stores, beta.mean, beta.var) {
  var <- ginv(1/mu.pr.var + num.stores/beta.var)
  mean <- var*(mu.pr.mean/mu.pr.var + num.stores*beta.mean/beta.var)
  mu0 <- rnorm(1, mean=mean, sd=sqrt(var))
  return (mu0)
}
sample.tausq <- function(be0, mu0, pr.a, pr.b) {
  shape <- pr.a + 1/2
  rate <- 1/2*(be0-mu0)^2 + pr.b
  tausq <- rgamma(1, shape=shape, rate=rate)
  return (1/tausq)
}

results <- Gibbs(data)
P <- results$PARAMS.BY.STORE
MU <- results$MU
TAUSQ <- results$TAUSQ

# Show traceplots for Mu's and Tau's.
par(mfrow=c(2, 2))
its <- 1:dim(MU)[1]
its <- its[seq(1001, dim(MU)[1], 10)]
count <- dim(MU)[2]
for (p in 1:count) {
  plot(its, MU[,p][seq(1001, dim(MU)[1], 10)], xlab="Iteration",
       ylab=bquote("Value of Mu"~.(p)), type="l")
}
for (p in 1:count) {
  plot(its, TAUSQ[,p][seq(1001, dim(MU)[1], 10)], xlab="Iteration",
       ylab=bquote("Value of Tausq"~.(p)), type="l",
       ylim=c(0, 30))
}

# Show boxplots for betas of all stores together.
par(mfrow=c(2,2))
for (p in 1:count) {
  data.to.plot <- t(P[,p,seq(1001, dim(MU)[1]-1, 10)])
  m <- melt(data.to.plot)[c(2, 3)]
  names(m) <- c("store", "value")
  bymedian <- with(m, reorder(m$store, m$value, median))
  boxplot(m$value ~ bymedian, data=m, xlab="Store",
```

```
          ylab=bquote("Value of Beta"~.(p)),
          main=bquote("Beta"~.(p)~"Values by Store"),
          ylim=c(3,15))
}
```