StatMod2 - Hierarchical Models and Shrinkage - Exercises 4

Maurice Diesendruck

April 8, 2015

2 Price Elasticity of Demand

2.1 Model

Here, I used a linear model in the log-log form, and computed Gibbs sampling using JAGS. Exact likelihood and priors are shown below:

```
# CONTENTS OF MODEL FILE
model {
for (j in 1:N) {
  LOG.Q[j] ~ dnorm(mean[j], sigsq)
  mean[j] <- beta[1] + beta[2]*log.P[j] + beta[3]*disp[j] +
             beta[4]*cross[j]
}
beta[1] ~ dnorm(0,0.001) # Diffuse priors
beta[2] ~ dnorm(0,0.001)
beta[3] ~ dnorm(0,0.001)
beta[4] ~ dnorm(0,0.001)
sigsq ~ dgamma(1, 1)
}
```

2.2 Gibbs Sampler

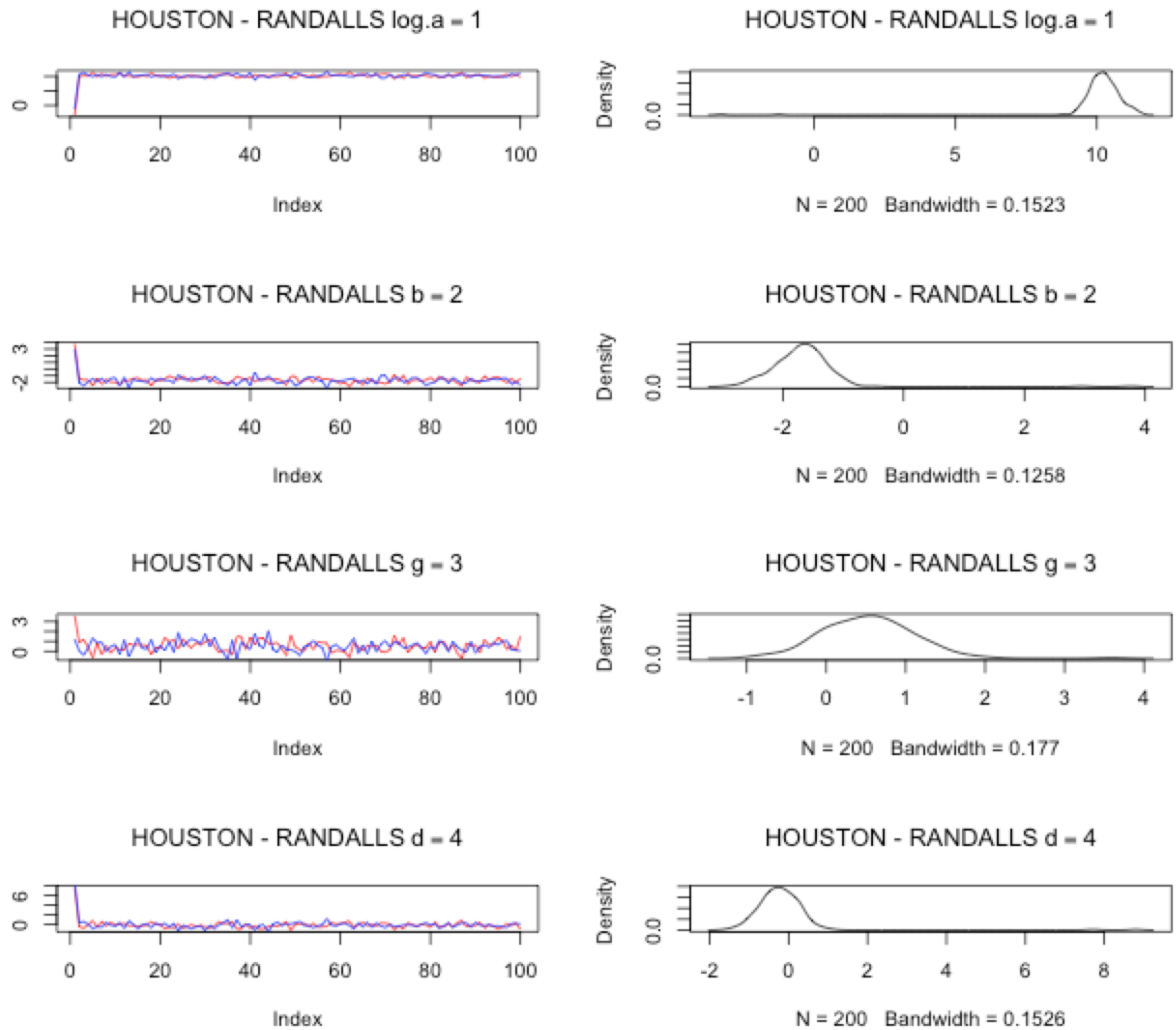
The following Gibbs sampler (using JAGS), produces chains for variables $log\alpha_i$, β_i, γ_i, and δ_i. In the code, these four variables are called $log.a$, b, g, and d.

```
# StatMod2 - Cheese - JAGS Version

library(rjags)
library(boot)
library(VGAM)
library(MASS)

# 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")
```

```r
data$log.Q <- log(data$vol)
data$log.P <- log(data$price)
attach(data)
n <- dim(data)[1]
store.names <- unique(data$store)

# Execute Gibbs sampler for all stores.
Execute <- function() {
  num.stores <- length(store.names)
  # Make container for store name, and means of 4 vars: log.a, b, g, d.
  PARAMS.BY.STORE <- matrix(NA, nrow=num.stores, ncol=5)
  for (i in 1:num.stores) {
    name <- toString(store.names[i])
    store.data <- data[which(data$store==name),]
    PARAMS.BY.STORE[i,] <- c(name, Gibbs(store.data, name))
  }
  return (PARAMS.BY.STORE)
}

Gibbs <- function(store.data, name) {
  # Set up JAGS model over 3 variables.
  jagsmodel <- jags.model(file="cheese-jags-model.txt",
                          data=list(LOG.Q=store.data$log.Q,
                               log.P=store.data$log.P,
                               disp=store.data$disp,
                               cross=store.data$log.P*store.data$disp,
                               N=length(store.data$log.Q)),
                          n.chains=2)

  # Fit jagsmodel.
  jagsfit <- jags.samples(jagsmodel,
                         variable.names=c("beta"),
                         n.iter=100000, thin=1000)

  # Create traceplots for beta chains.
  par(mfrow=c(4, 2))
  ref <- c("log.a", "b", "g", "d")
  for (i in 1:4) {
    # jagsfit£var takes three arguments: parameter, row, chain.
    plot(jagsfit$beta[i,,1], type="l", col="red", ylab="",
         main=bquote(.(name) ~ .(ref[i]) == .(i)))
    lines(jagsfit$beta[i,,2], type="l", col="blue")
    plot(density(jagsfit$beta[i,,]), type="l",
         main=bquote(.(name) ~ .(ref[i]) == .(i)))
  }

  mean.params <- rowMeans(jagsfit$beta[,,1])
```

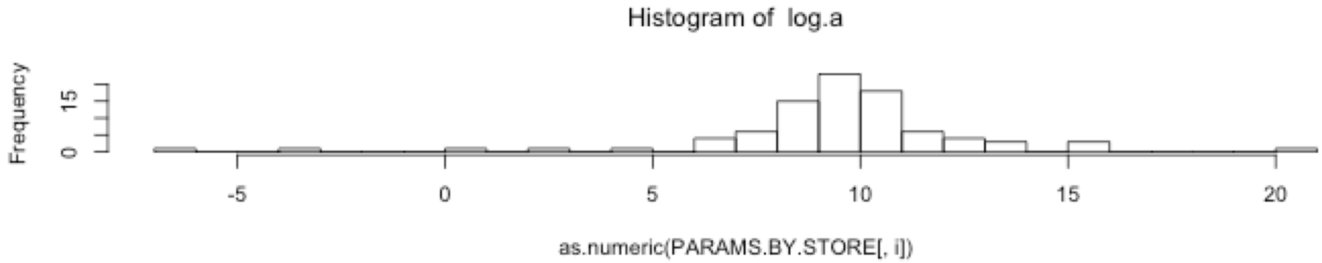
```
  return (mean.params)
}

PARAMS.BY.STORE <- Execute()
```

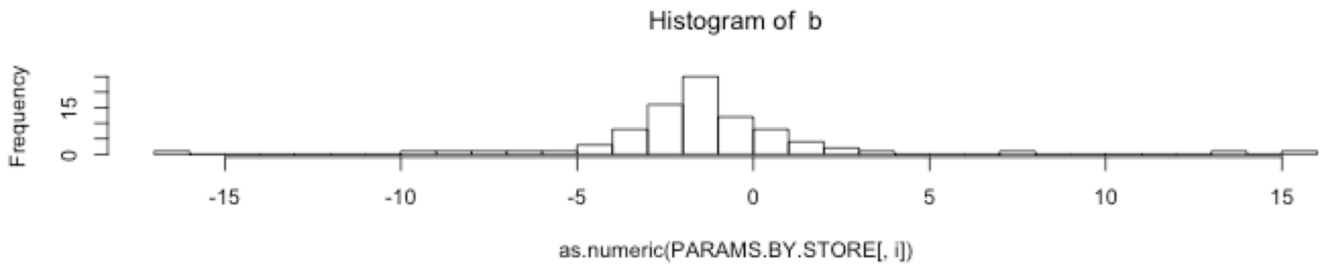
Here is an example of the output for one store:

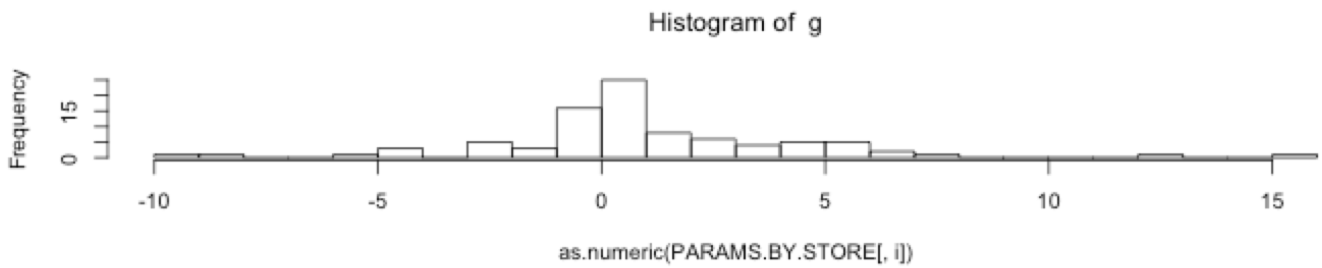


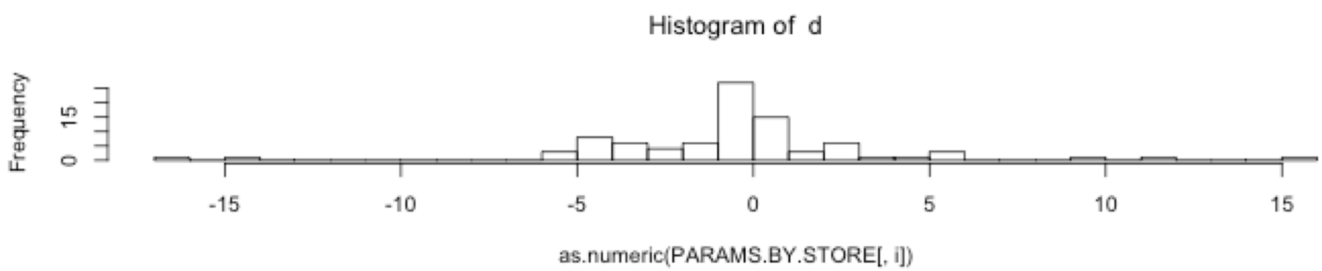
The variable *PARAMS.BY.STORE*, holds the chain means for each variable. A subset of this matrix is shown below:

```
"NEW YORK (NEW) - A & P"           "10.4892812337663" "-1.63881657781477"  "-0.280605857535447" "0.668721228527719"
"NEW YORK (NEW) - PATHMARK"        "6.7075209873317"  "0.835400624836997"  "6.46354378752612"   "-4.19724412764956"
"NEW YORK (NEW) - WALDBAUMS"       "10.0957040098413" "-1.80867944260936"  "1.21321149669627"   "-0.417223354485007"
"ROANOKE (NEW) - FOOD LION"        "9.60700679239987" "-1.77892466181445"  "-0.211543833589082" "0.420166311897755"
"ROANOKE (NEW) - KROGER CO"        "9.95425535627714" "-1.67798870689548"  "0.956855418268611"  "-0.948109553976121"
"DALLAS/FT. WORTH - WINN DIXIE"    "5.21235517793787" "1.06854357841277"   "6.11312180101043"   "-5.18876829458762"
```


Histogram of log.a
Frequency
0
15
-5
0
5
10
15
20
as.numeric(PARAMS.BY.STORE[, i])


Histogram of b
Frequency
0
15
-15
-10
-5
0
5
10
15
as.numeric(PARAMS.BY.STORE[, i])


Histogram of g
Frequency
0
15
-10
-5
0
5
10
15
as.numeric(PARAMS.BY.STORE[, i])


Histogram of d
Frequency
0
15
-15
-10
-5
0
5
10
15
as.numeric(PARAMS.BY.STORE[, i])