Online sellers revisited worksheet

Joachim Vandekerckhove

Set up environment

```
library(rjags)
library(coda)
library(ggplot2)
library(dplyr)
library(readr)
```

Data

Load the data and necessary colors.

```
k <- c(10, 48, 186,  75, 1)
n <- c(10, 50, 200, 100, 2)

nSellers <- length(k)

sellerColors <- c("#003366", "#B00000", "#FFD700",
                  "#8FBC8F", "#778899")
```

JAGS Models

Create the JAGS model files for independent, aggregate, and hierarchical models and run MCMC sampling for each model.

```
# Define parameters for MCMC
nChains  <-    8
nBurnin  <- 1000
nSamples <- 5000
nThin    <-    1

samplesList <- list()
```

```
independent_model <- "
model{
  for (i in 1:nSellers){
    k[i] ~ dbin(theta[i], n[i])
    kPostpred[i] ~ dbin(theta[i], n[i])
    theta[i] ~ dunif(0, 1)
  }
}
"
modelFile <- "onlineSellersIndependent.jags"
writeLines(independent_model, con = modelFile)
```

```
data_list <- list(k = k, n = n, nSellers = nSellers)

inits <- function() list(theta = runif(nSellers))

params <- c("theta", "kPostpred")
```

```
jags_model <- jags.model(modelFile,
                         data     = data_list,
                         inits    = inits,
                         n.chains = nChains,
                         n.adapt  = nBurnin)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 10
##    Total graph size: 23
##
## Initializing model
```

```
update(jags_model, n.iter = nBurnin)
samples <- coda.samples(jags_model,
                        variable.names = params,
                        n.iter         = nSamples,
                        thin           = nThin)

samplesList[[modelFile]] <- samples
```

```
aggregate_model <- "
model{
  for (i in 1:nSellers){
    k[i] ~ dbin(theta, n[i])
    kPostpred[i] ~ dbin(theta, n[i])
  }
  theta ~ dunif(0, 1)
}
"
modelFile <- "onlineSellersAggregate.jags"
writeLines(aggregate_model, con = modelFile)
```

```
data_list <- list(k = k, n = n, nSellers = nSellers)

inits <- function() list(theta = runif(1))

params <- c("theta", "kPostpred")
```

```r
jags_model <- jags.model(modelFile,
                         data     = data_list,
                         inits    = inits,
                         n.chains = nChains,
                         n.adapt  = nBurnin)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 6
##    Total graph size: 19
##
## Initializing model
```

```
update(jags_model, n.iter = nBurnin)
samples <- coda.samples(jags_model,
                        variable.names = params,
                        n.iter         = nSamples,
                        thin           = nThin)

samplesList[[modelFile]] <- samples
```

```
hierarchical_model <- "
model{
  for (i in 1:nSellers){
    k[i] ~ dbin(theta[i], n[i])
    kPostpred[i] ~ dbin(theta[i], n[i])
    theta[i] ~ dnorm(mu, 1/sigma^2)T(0, 1)
  }
  mu ~ dunif(0, 1)
  sigma ~ dunif(0, 1)
}
"
modelFile <- "onlineSellersHierarchical.jags"
writeLines(hierarchical_model, con = modelFile)
```

```
data_list <- list(k = k, n = n, nSellers = nSellers)

inits <- function() list(theta = runif(nSellers),
                         mu    = runif(1),
                         sigma = runif(1))

params <- c("theta", "kPostpred", "mu", "sigma")
```

```
jags_model <- jags.model(modelFile,
                         data     = data_list,
                         inits    = inits,
                         n.chains = nChains,
                         n.adapt  = nBurnin)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 5
##    Unobserved stochastic nodes: 12
##    Total graph size: 28
##
## Initializing model
```

```
update(jags_model, n.iter = nBurnin)
samples <- coda.samples(jags_model,
                        variable.names = params,
                        n.iter         = nSamples,
                        thin           = nThin)

samplesList[[modelFile]] <- samples
```

Inspect Results

Print the multivariate Gelman diagnostic for each model.

```
mpsrf <- list()
modelList <- c("onlineSellersIndependent.jags",
               "onlineSellersAggregate.jags",
               "onlineSellersHierarchical.jags")
for (modelName in modelList) {
  samples <- samplesList[[modelName]]
  gelman_diag <- gelman.diag(samples,
                             multivariate=TRUE)
  mpsrf[[modelName]] <- gelman_diag$mpsrf
}
gelman_df <- data.frame(Model = names(mpsrf),
                        MPSRF = unlist(mpsrf))
```

Inspect Results

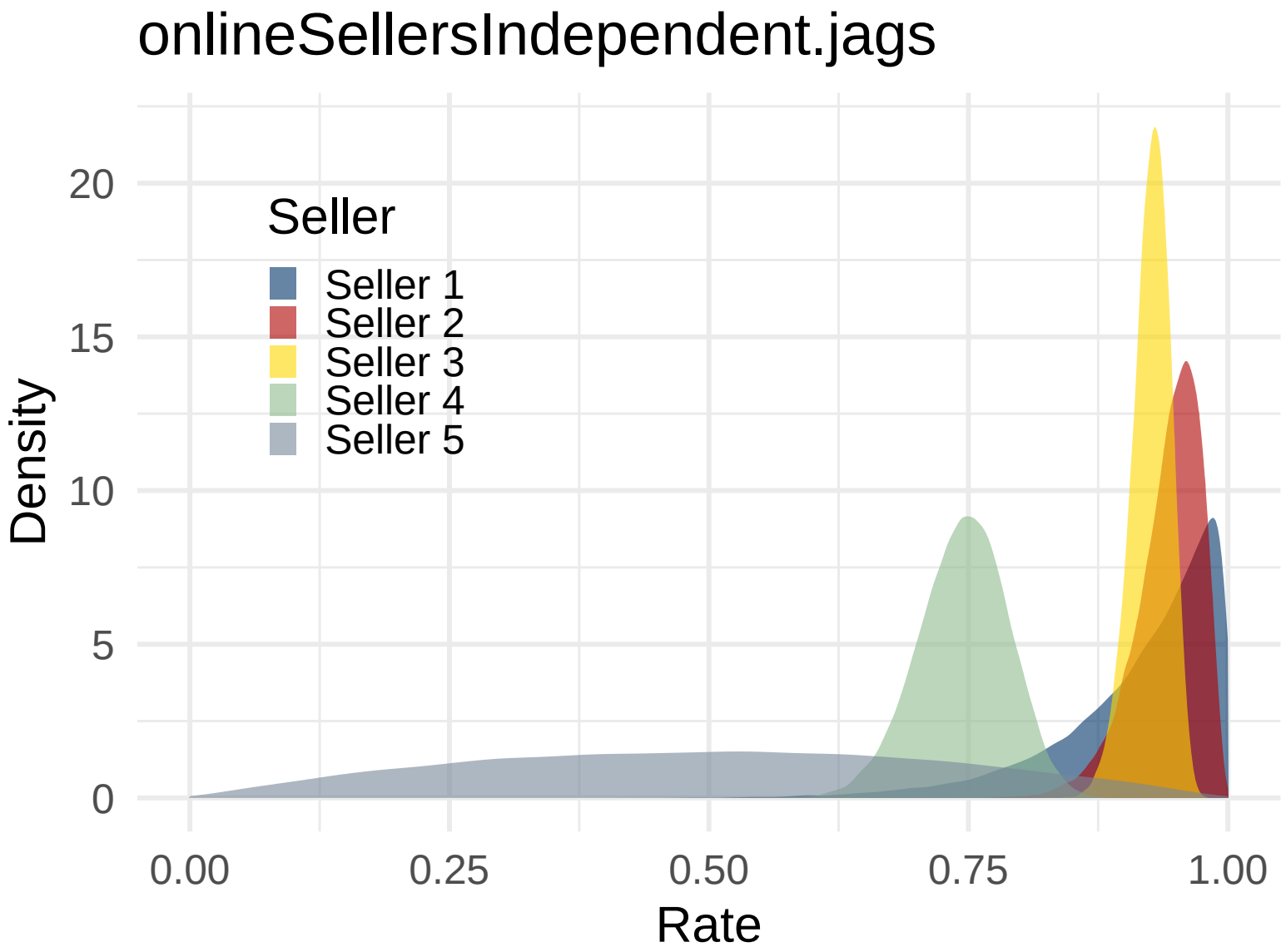
Print the multivariate Gelman diagnostic for each model.

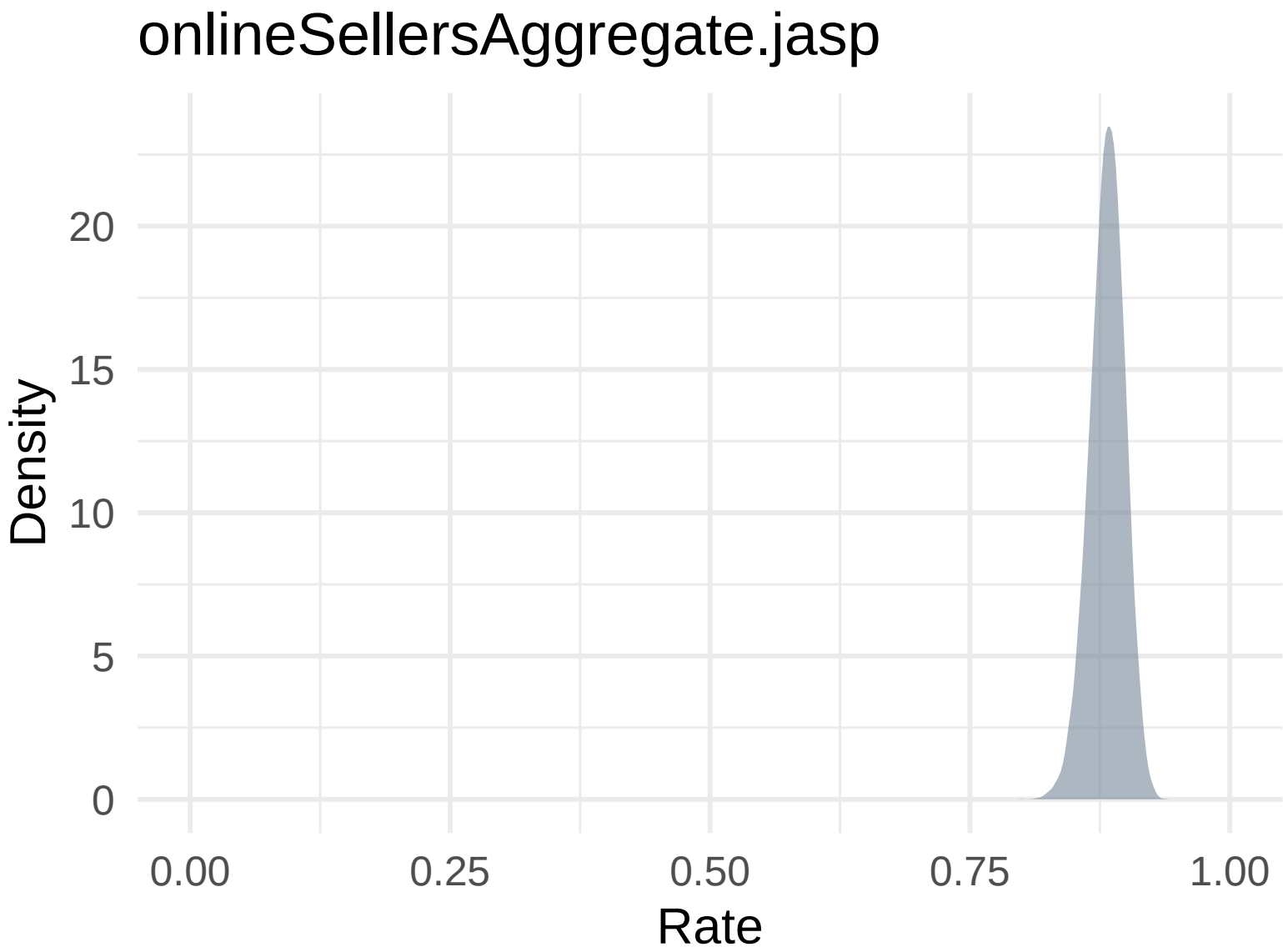
```
knitr::kable(gelman_df, row.names = FALSE,
             col.names = c("Model",
                           "Multivariate PSRF"))
```

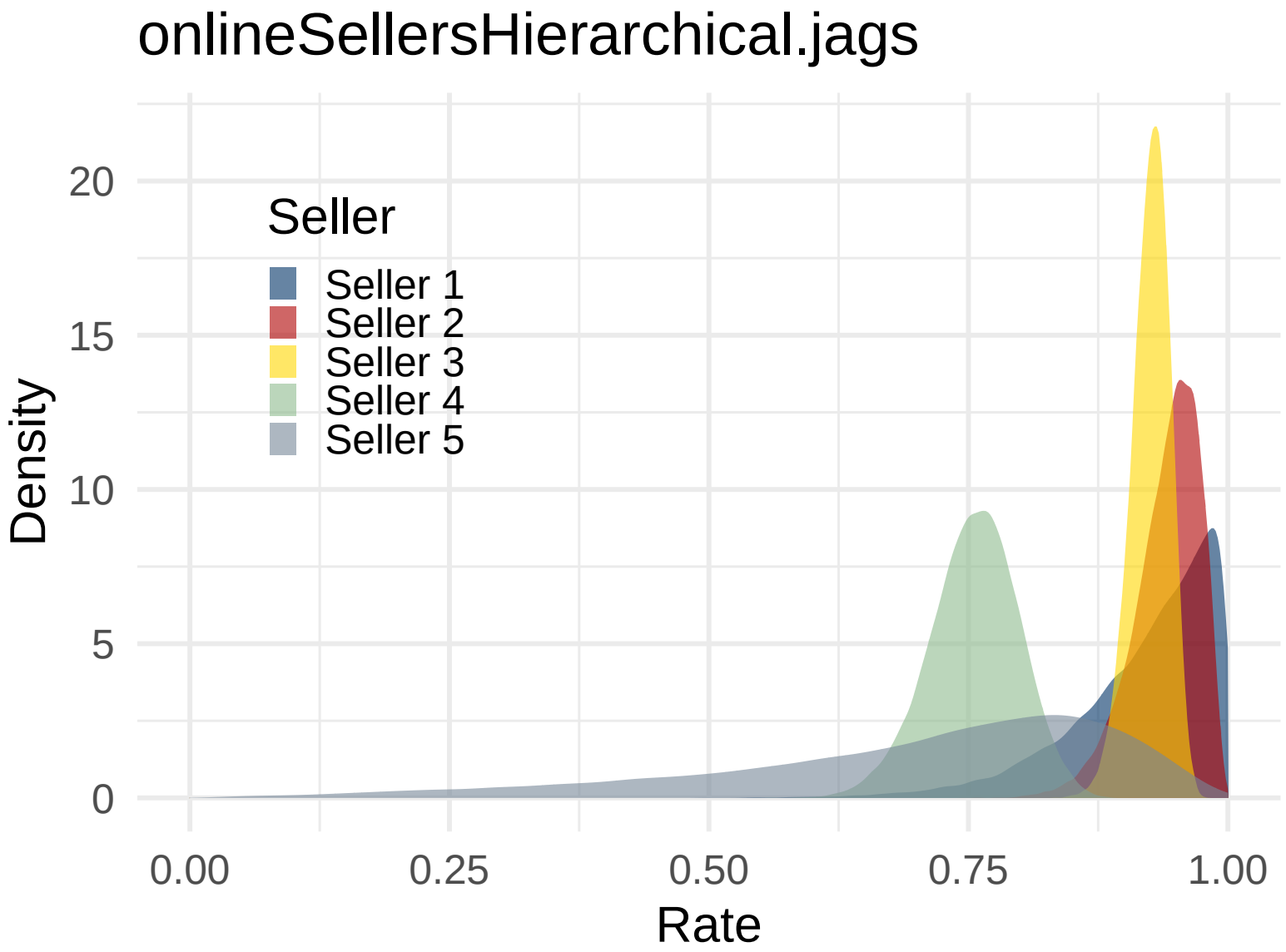
Model	Multivariate PSRF
onlineSellersIndependent.jags	1.001188
onlineSellersAggregate.jags	1.000179
onlineSellersHierarchical.jags	1.004616

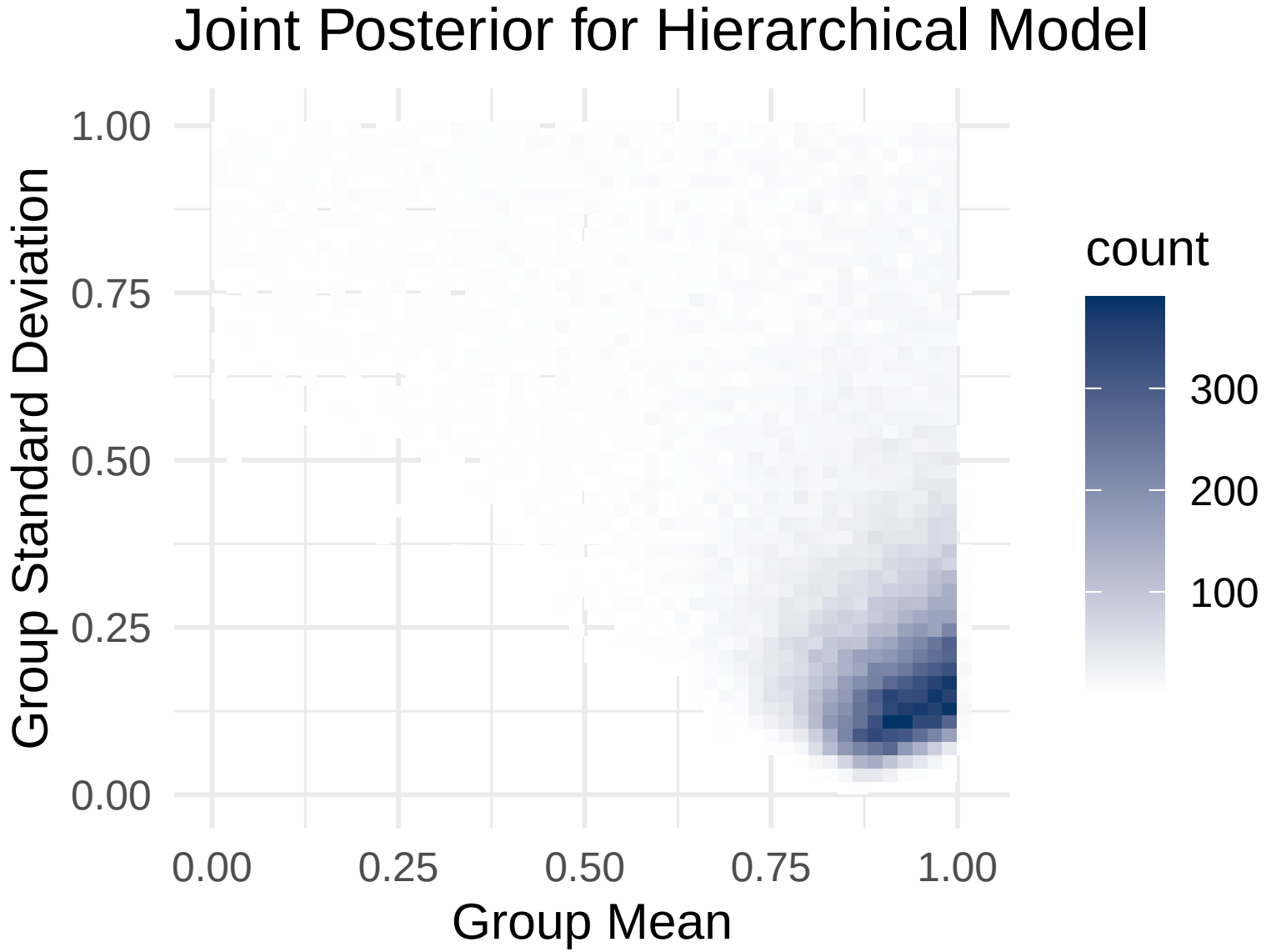
Analysis and Plots

Generate plots for posterior distributions and posterior predictive distributions.


onlineSellersIndependent.jags
Seller
Seller 1
Seller 2
Seller 3
Seller 4
Seller 5
Density
20
15
10
5
0
0.00
0.25
0.50
0.75
1.00
Rate


onlineSellersAggregate.jasp
Density
20
15
10
5
0
0.00
0.25
0.50
0.75
1.00
Rate


onlineSellersHierarchical.jags
Seller
Seller 1
Seller 2
Seller 3
Seller 4
Seller 5
Density
20
15
10
5
0
0.00
0.25
0.50
0.75
1.00
Rate


Joint Posterior for Hierarchical Model
Group Standard Deviation
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
Group Mean
count
300
200
100