Shrinkage in Bayesian Hierarchical Count Regression

W. Christopher Carleton

Shrinkage in Bayesian Hierarchical Models

Paths, Libraries, and Parameters

```
datapath <- "../Data/"

library(nimble)
```

```
## nimble version 0.11.1 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit http://R-nimble.org.
```

```
##
## Attaching package: 'nimble'
```

```
## The following object is masked from 'package:stats':
##
##     simulate
```

```
# vars for the counts
nbins <- 100
beta <- 0.04
x <- 1:nbins
lambda <- exp(beta * x)

#parameters for mcmc
niter <- 500000
burnin <- 0.2
retain <- floor(niter * burnin):niter
nretain <- length(retain)
```

Sample Count Sequences

```
sample_counts <- function(nil, n, lambda){
    counts <- rpois(n = nbins,
                lambda = lambda)
    return(counts)
}

K <- 200

Y <- sapply(1:K,
```

```
            sample_counts,
            n = nbins,
            lambda = lambda)
X <- matrix(rep(x,K), ncol = K)

savepath <- paste(datapath,
                  "count_sequences.RData",
                  sep = "")
```

```
poisCode <- nimbleCode({
   ###top-level regression
   B0 ~ dnorm(0, 100)
   B1 ~ dnorm(0, 100)
   sigB0 ~ dunif(1e-10, 100)
   sigB1 ~ dunif(1e-10, 100)
   for (k in 1:K) {
      ###low-level regression
      b0[k] ~ dnorm(mean = B0, sd = sigB0)
      b1[k] ~ dnorm(mean = B1, sd = sigB1)
      for (n in 1:N){
        r[n, k] <- exp(b0[k] + X[n, k] * b1[k])
        Y[n, k] ~ dpois(lambda = r[n, k])
      }
   }
})
```

```
K <- 5

poisData <- list(Y = Y[, 1:K],
                 X = X[, 1:K])

poisConsts <- list(N = nbins,
                 K = K)

poisInits <- list(B0 = 0,
                 B1 = 0,
                 b0 = rep(0, K),
                 b1 = rep(0, K),
                 sigB0 = 0.0001,
                 sigB1 = 0.0001)

poisModel <- nimbleModel(code = poisCode,
                         data = poisData,
                         inits = poisInits,
                         constants = poisConsts)

#compile nimble model to C++ code-much faster runtime
C_poisModel <- compileNimble(poisModel, showCompilerOutput = FALSE)

#configure the MCMC
poisModel_conf <- configureMCMC(poisModel)
```

```
## ===== Monitors =====
## thin = 1: B0, B1, sigB0, sigB1
```

```
## ===== Samplers =====
## RW sampler (12)
##   - sigB0
##   - sigB1
##   - b0[]  (5 elements)
##   - b1[]  (5 elements)
## conjugate sampler (2)
##   - B0
##   - B1
```

```
poisModel_conf$monitors <- c("B0", "B1", "sigB0", "sigB1")
poisModel_conf$addMonitors2(c("b0", "b1"))
```

```
## thin = 1: B0, B1, sigB0, sigB1
## thin2 = 1: b0, b1
```

```
#samplers
poisModel_conf$removeSamplers(c("B0", "B1", "sigB0", "sigB1", "b0", "b1"))
poisModel_conf$addSampler(target = c("B0", "B1", "sigB0", "sigB1"),
                          type = "AF_slice")
for(k in 1:K){
   poisModel_conf$addSampler(target = c(paste("b0[", k, "]", sep = ""),
                                        paste("b1[", k, "]", sep = "")),
                          type = "AF_slice")
}

#thinning to conserve memory when the samples are saved below
poisModel_conf$setThin(1)
```

```
## thin = 1: B0, B1, sigB0, sigB1
## thin2 = 1: b0, b1
```

```
poisModel_conf$setThin2(1)
```

```
## thin = 1: B0, B1, sigB0, sigB1
## thin2 = 1: b0, b1
```

```
#build MCMC
poisModelMCMC <- buildMCMC(poisModel_conf)

#compile MCMC to C++-much faster
C_poisModelMCMC <- compileNimble(poisModelMCMC, project = poisModel)

#set seed for replicability
set.seed(1)

#save the MCMC chain (monitored variables) as a matrix
nimble_samples_5 <- runMCMC(C_poisModelMCMC, niter = niter)
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

Plotting

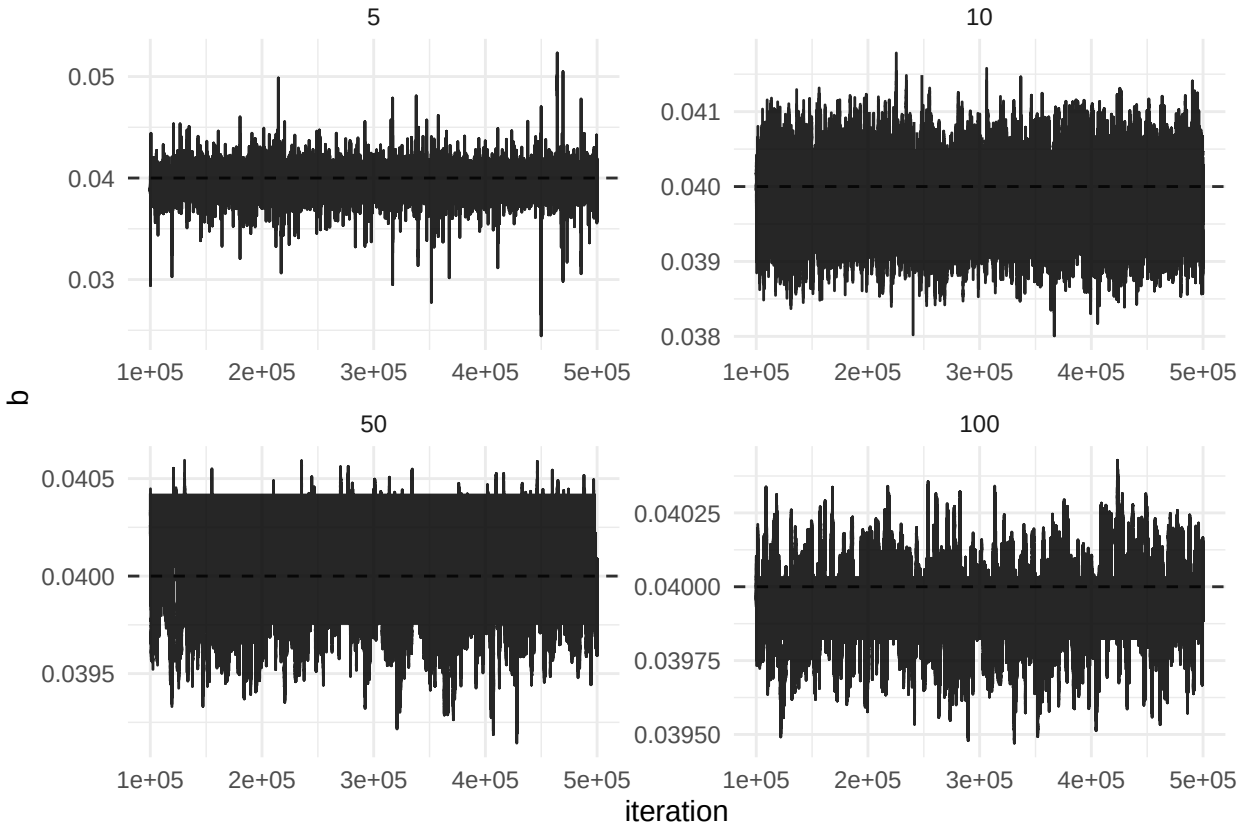
```
library(ggplot2)
library(ggpubr)
```

```
K_labels <- rep(c("5","10","50","100"), each = nretain)
K_labels <- factor(K_labels, levels=c("5","10","50","100"))
iterations <- rep(retain, 4)

beta_samples <- c(nimble_samples_5$samples[retain, 2],
                  nimble_samples_10$samples[retain, 2],
                  nimble_samples_50$samples[retain, 2],
                  nimble_samples_100$samples[retain, 2])

mcmc_samples_beta <- data.frame(model_run = K_labels,
                                iteration = iterations,
                                b = beta_samples)
```

```
ggplot(data = mcmc_samples_beta) +
    geom_hline(yintercept = beta,
                linetype = 2,
                alpha = 0.8) +
    geom_path(mapping = aes(y = b, x = iteration),
                alpha = 0.85) +
    facet_wrap(~ model_run, ncol = 2, scales = "free") +
    theme_minimal()
```




```
ggplot(data = mcmc_samples_beta) +
        geom_vline(xintercept = beta,
                linetype = 2,
                alpha = 0.8) +
```

```
        geom_density(mapping = aes(y = ..scaled.., x = b, fill = model_run),
                alpha = 0.5) +
        scale_x_continuous(limits = c(0.0375, 0.044)) +
        facet_wrap(~ model_run, ncol = 2, scales = "free") +
        theme_minimal()
```

```
## Warning: Removed 973 rows containing non-finite values (stat_density).
```