The Horseshoe model can be described as follows:

$$\begin{aligned} y_{ig} &\overset{ind}{\sim} N(x_i\theta_g, \sigma^2) \\ \theta_g &\overset{ind}{\sim} N(0, \lambda_g^2\tau^2) \\ \lambda_g^2 &\sim Ca^+(0,1) \\ \tau^2 &\sim Ca^+(0,1) \\ \pi(\sigma^2) &\propto 1/\sigma^2 \end{aligned}$$

1. The goal for this sampler is to make it in as few steps as possible. Here are the steps used:
   (a) **let** $i = 1$
   (b) **let** $\theta_{1,0}, \cdots, \theta_{n_g,0}, \lambda_{1,0}, \cdots, \lambda_{n_g,0}, \sigma_0^2, \tau_0^2$ be given
   (c) **while** $i <$ `ITER`
      i. **For** $g = 1, \cdots, G$, sample $\theta_g \sim N(\mu_g, \gamma_g^2)$
      ii. **For** $g = 1, \cdots, G$, sample λ_g^2 via a Metropolis-Hastings step the posterior from using the standard non-informative prior as its proposal i.e. for each λ_g^2:
         A. **Draw** $\lambda_g^{2'} \sim IG\left(\frac{1}{2}, \frac{\theta_g^2}{2\tau^2}\right)$
         B. **Draw** u from U(0,1)
         C. **If** $u < \frac{q(\lambda_g^{2'})}{q(\lambda_g^2)} \frac{p(\lambda_g^2)}{p(\lambda_g^{2'})}$ **then**
            let $\lambda_{g,i}^2 = \lambda_g^{2'}$ **else** $\lambda_{g,i}^2 = \lambda_{g,i-1}^2$
   (d) **Sample** $\sigma^2 \sim IG\left(\frac{n}{2}, \frac{1}{2}\sum_{g=1}^{G}\sum_{i=1}^{n_g}(y_{ig} - \theta_g)^2\right)$
   (e) **Sample** τ^2 via a Metropolis-Hastings step the posterior from using the standard non-informative prior as its proposal i.e.
      i. **Draw** $\tau^{2'} \sim IG\left(\frac{G}{2}, \sum_{g=1}^{G}\frac{\theta_g^2}{2\lambda_g^2}\right)$
      ii. **Draw** u from U(0,1)
      iii. **If** $u < \frac{q(\tau^{2'})}{q(\tau^2)} \frac{p(\tau^2)}{p(\tau^{2'})}$ **then**
         let $\tau_i^2 = \tau^{2'}$ **else** $\tau_i^2 = \tau_{i-1}^2$

   Full Conditionals: θ:

$$\begin{aligned} \theta_g|\cdots &\overset{ind}{\sim} N(\mu_g, \gamma_g^2) \\ \gamma_g^2 &= \left[\frac{1}{\lambda_i^2\tau_i^2} + \frac{n_g}{\sigma^2}\right]^{-1} \\ \mu_g &= \gamma_g^2\left[0 * \lambda_i^{-2}\tau_i^{-2} + \bar{y}_g n_g \sigma^{-2}\right] \\ &= \gamma_g^2\left[\bar{y}_g n_g \sigma^{-2}\right] \end{aligned}$$

σ^2:

$$\sigma^2|\cdots \sim IG\left(\frac{n}{2}, \frac{1}{2}\sum_{g=1}^{G}\sum_{i=1}^{n_g}(y_{ig}-\theta_g)^2\right)$$

λ_g^2 is sampled via a Metropolis-Hastings step since it's conditional posterior is unknown. For the proposal, we will use the posterior from using the standard non-informative prior($p(\lambda_g^2) \propto 1/\lambda_g^2$):

$$\lambda_g^2|\cdots \sim IG\left(\frac{1}{2}, \frac{\theta_g^2}{2\tau^2}\right)$$
$$q(\lambda_g^2|\cdots) \propto \frac{1}{\sqrt{\lambda_g^2}}\frac{1}{1+\lambda_g^2}exp\left(-\frac{\theta_g^2}{2\lambda_g^2\tau^2}\right)$$

Similarly, τ^2 is sampled via a Metropolis-Hastings step since it's conditional posterior is unknown. For the proposal, we will use the posterior from using the standard non-informative prior($p(\tau^2) \propto 1/\tau^2$):

$$\tau^2|\cdots \sim IG\left(\frac{G}{2}, \sum_{g=1}^{G}\frac{\theta_g^2}{2\lambda_g^2}\right)$$
$$q(\tau^2|\cdots) \propto \left(\frac{1}{\sqrt{\tau^2}}\right)^G\frac{1}{1+\tau^2}exp\left(-\frac{1}{2\tau^2}\sum_{g=1}^{G}\frac{\theta_g^2}{\lambda_g^2}\right)$$

This model described above I called the Informative Horseshoe since its proposals for τ and λ_g are fairly informative. I also ran a similar model using non-informative proposals for τ and λ_g. This will be referred to as the Vague Horseshoe.

2. Here is the samplers coded up in `R`.

```r
mcmc_horseshoe_inform <- function(n_reps, y, group, initial_values){
  require("invgamma")
  require("dplyr")

  # Setup the storage for posterior samples
  G <- length(unlist(initial_values["theta"])) # number of groups
  keep_theta  <- matrix(NA, ncol=G, nrow=n_reps)
  keep_lambda <- matrix(NA, ncol=G, nrow=n_reps)
  keep_sigma2 <- rep(NA, n_reps)
  keep_tau2   <- rep(NA, n_reps)

  # Set the initial values
  lambda20 <- as.numeric(unlist(initial_values["lambda"]))
  sigma20  <- as.numeric(initial_values["sigma2"])
  tau20    <- as.numeric(initial_values["tau2"])
  theta0   <- as.numeric(unlist(initial_values["theta"]))
```

```
# Setup summary stats
tmpdf <- data.frame(y, group)
n <- length(y)
n_g <- data.frame(tmpdf %>% group_by(group) %>% summarise(n_g = length(y)))
n_g <- as.numeric(n_g[,2])
ybar_g <- data.frame(tmpdf %>% group_by(group) %>% summarise(ybar_g = mean(y)))
ybar_g <- as.numeric(ybar_g[,2])

for(i in 1:n_reps){
  for(g in 1:G){
    ##### Sample Theta_g #####
    sigmaTheta <- ((1/lambda20[g]*tau20) + (n_g[g]/sigma20))^(-1)
    muTheta    <- sigmaTheta*(ybar_g[g]*n_g[g]/sigma20)
    theta      <- rnorm(1, muTheta, sigmaTheta)

    ###### Sample Lambda_g^2 with M-H Step #####
    al <- 1/2
    bl <- 0.5*(theta0[g]^2/tau20)
    # lambda2 <- rinvgamma(1, al, bl)
    lambda2 <- 1/rgamma(1, al, 1/bl)

    # Calc phi
    targetL <- function(newL, oldL, thg, t2){
      return(exp(0.5*(log(oldL) - log(newL)) + log(1+oldL^2) - log(1+newL^2) + (tl
      }

    propL   <- function(newL, oldL, th_g, t2){
      pl <- 1.5*(log(newL) - log(oldL)) + (th_g^2/(2*t2))/newL - (th_g^2/(2*t2))/
      return(exp(pl))
      }

    phiL <- targetL(lambda2, lambda20[g], theta0[g], tau20)#*propL(lambda2, lambd
    if(is.na(phiL > 0)){phiL = 0
      cat("i=",i,"g=", g,"theta=",theta0[g],"bl=",bl, "lambda=",lambda2,"lambda0="
                              "target=",targetL(lambda2, lambda20[g], theta0[g], tau

    uL <- runif(1,0,1)
    if(uL > phiL){lambda2 <- lambda20[g]}

    # Update Storage
    keep_lambda[i,g]  <- lambda2
    keep_theta[i,g]   <- theta
  }


  ###### Sample Sigma^2 #####
```

```
    tmpd <- data.frame(y, group, theta = theta0[group])
    SSE <- sum((tmpd$y-tmpd$theta)^2)
    a <- n/2
    b <- 0.5*SSE
    sigma2 <- rinvgamma(1,a,b)

    ###### Sample Tau^2 with M-H Step #####
    at <- G/2
    bt <- 0.5*(sum(theta0^2/lambda20))
    # tau2 <- rinvgamma(1, at, bt)
    tau2 <- 1/rgamma(1, at, 1/bt)

    # Calc phi
    targetT <- function(newT, oldT, th, lamb, GG){
      tl <- (GG/2)*(log(oldT) - log(newT)) + log(1+oldT^2) - log(1+newT^2) - 0.5*sum
      return(exp(tl))
      }

    propT <- function(newT, oldT, th, lamb,GG){
      pl <- (GG/2 + 1)*(log(newT) - log(oldT)) - (0.5*sum(th^2/lamb)/oldT) + (0.5*su
      return(exp(pl))
      }


    phiT <- targetT(tau2, tau20, theta0, lambda20, G)##propT(tau2, tau20, theta0, l
    if(is.na(phiT > 0)){phiT = 0
     print(tau2)
     cat("TauT=",targetT(tau2, tau20, theta0, lambda20, G))}
    uT <- runif(1,0,1)
    if(uT > phiT){tau2 <- tau20}

    # Update Storage
    keep_sigma2[i]  <- sigma2
    keep_tau2[i]    <- tau2


    # Update current values
    lambda20 <- keep_lambda[i,]
    theta0   <- keep_theta[i,]
    sigma20  <- keep_sigma2[i]
    tau20    <- keep_tau2[i]
  }


return(list(
  theta  = keep_theta[,],
```

```
  lambda = keep_lambda[,],
  sigma2 = keep_sigma2,
  tau2   = keep_tau2
))
}

mcmc_horseshoe_vague <- function(n_reps, y, group, initial_values){
  require("MCMCpack")
  require("dplyr")

  # Setup the storage for posterior samples
  G <- length(unlist(initial_values["theta"])) # number of groups
  keep_theta  <- matrix(NA, ncol=G, nrow=n_reps)
  keep_lambda <- matrix(NA, ncol=G, nrow=n_reps)
  keep_sigma2 <- rep(NA, n_reps)
  keep_tau2   <- rep(NA, n_reps)

  # Set the initial values
  lambda20 <- as.numeric(unlist(initial_values["lambda"]))
  sigma20  <- as.numeric(initial_values["sigma2"])
  tau20    <- as.numeric(initial_values["tau2"])
  theta0   <- as.numeric(unlist(initial_values["theta"]))

  # Setup summary stats
  tmpdf <- data.frame(y, group)
  n <- length(y)
  n_g <- data.frame(tmpdf %>% group_by(group) %>% summarise(n_g = length(y)))
  n_g <- as.numeric(n_g[,2])
  ybar_g <- data.frame(tmpdf %>% group_by(group) %>% summarise(ybar_g = mean(y)))
  ybar_g <- as.numeric(ybar_g[,2])

  for(i in 1:n_reps){
    for(g in 1:G){
      ##### Sample Theta_g #####
      sigmaTheta <- ((1/lambda20[g]*tau20) + (n_g[g]/sigma20))^(-1)
      muTheta    <- sigmaTheta*(ybar_g[g]*n_g[g]/sigma20)
      theta      <- rnorm(1, muTheta, sigmaTheta)

      ###### Sample Lambda_g^2 with M-H Step #####
      al <- 1
      bl <- 2
      lambda2 <- rinvgamma(1, al, bl)

      # Calc phi
      targetL <- function(newL, oldL, th_g, t2){
        tl <- 0.5*(log(oldL) - log(newL)) + 2*(log(1+oldL) - log(1+newL)) + (th_g^2/
```

```
    return(exp(tl))
    }

  propL   <- function(newL, oldL, th_g, t2){
    pl <- (1.5)*(log(newL) - log(oldL)) - (20/oldL) + (20/newL)
    return(exp(pl))
    }

  phiL <- targetL(lambda2, lambda20[g], theta0[g], tau20)#*propL(lambda2, lambd

  uL <- runif(1,0,1)
  if(uL > phiL){lambda2 <- lambda20[g]}

  # Update Storage
  keep_lambda[i,g]  <- lambda2
  keep_theta[i,g]   <- theta
}


###### Sample Sigma^2 #####
tmpd <- data.frame(y, group, theta = theta0[group])
SSE <- sum((tmpd$y-tmpd$theta)^2)
a <- n/2
b <- 0.5*SSE
sigma2 <- rinvgamma(1,a,b)

###### Sample Tau^2 with M-H Step #####
at <- 1
bt <- 1
tau2 <- rinvgamma(1, at, bt)

# Calc phi
targetT <- function(newT, oldT, th, lamb, GG){
  tl <- (GG/2)*(log(oldT) - log(newT)) + 2*(log(1+oldT) - log(1+newT)) - (.5*su
  return(exp(tl))
  }

propT <- function(newT, oldT,GG){
  pl <- (GG/2 + 1)*(log(newT) - log(oldT)) - (20/oldT) + (20/newT)
  return(exp(pl))
  }

phiT <- targetT(tau2, tau20, theta0, lambda20, G)#*propT(tau2, tau20, G)

uT <- runif(1,0,1)
if(uT > phiT){tau2 <- tau20}
```

```
    # Update Storage
    keep_sigma2[i]  <- sigma2
    keep_tau2[i]    <- tau2


    # Update current values
    lambda20 <- keep_lambda[i,]
    theta0   <- keep_theta[i,]
    sigma20  <- keep_sigma2[i]
    tau20    <- keep_tau2[i]
  }


return(list(
  theta  = keep_theta[,],
  lambda = keep_lambda[,],
  sigma2 = keep_sigma2,
  tau2   = keep_tau2
))
}
```

3. For the data simulation, I set up my response and mean parameters like so:

$$\theta_1 \cdots \theta_{20} = 0$$
$$\theta_{21} \cdots \theta_{25} \neq 0$$
$$y \sim N(\theta, 3 * I)$$

And in R,

```
library("dplyr")
library("Rcpp")
sourceCpp("homework2.cpp")


# Simulate data
set.seed(1)
G = 25
theta = c(rep(0,G-5),35,10,15,15,20)
trueTheta = theta
d = data.frame(group = rep(1:G, each=15))
d$theta = theta[d$group]
d$y = rnorm(nrow(d), d$theta, 3)

# Get initial values
s <- d %>% group_by(group) %>%
```

```
  summarize(n = n(),
            mean = mean(y),
            var  = var(y))

initial_values = list(
  lambda = s$var,
  mu = mean(s$mean),
  theta = s$mean,
  sigma2 = mean(s$var),
  tau2 = var(s$mean))

# Set prior
prior = list(m = 0, C = 100, a = 1, b = 1, c = 1)

# Additional initial and prior values
initial_values$gamma = abs(s$mean) > 2*sqrt(initial_values$sigma2)
initial_values$pi = 1-mean(initial_values$gamma)
initial_values$psi = with(initial_values, ifelse(gamma, theta, rnorm(G, mu, sqrt(tau
initial_values$mu = 0
initial_values$tau2 = var(s$mean[initial_values$gamma])
if (is.na(initial_values$tau2)) initial_values$tau2 = 1
prior$a_pi = prior$b_pi = 1
initial_values$phi = rep(1,G)
prior$df = 3
```

We ran all the models for 10^4 iterations. The models all ran relatively quickly even though they were coded in R. They all preformed well in estimating the θ parameters. In Figure 1, the credible intervals of the groups plotted by each model. The intervals when beta is equal to 0 are fairly tight with the point mass prior basically equaling 0 in its samples. When beta is not zero the bands are still fairly tight with the exception of the vague Horseshoe model whose bands are fairly wide due to the nature of vague proposals in it's algorithm. The informative Horseshoe however did not sample the variance parameters very well while the vague Horseshoe model did. This might be remedied if a random walk was used in the proposals for τ and λ_g instead of the MH step.

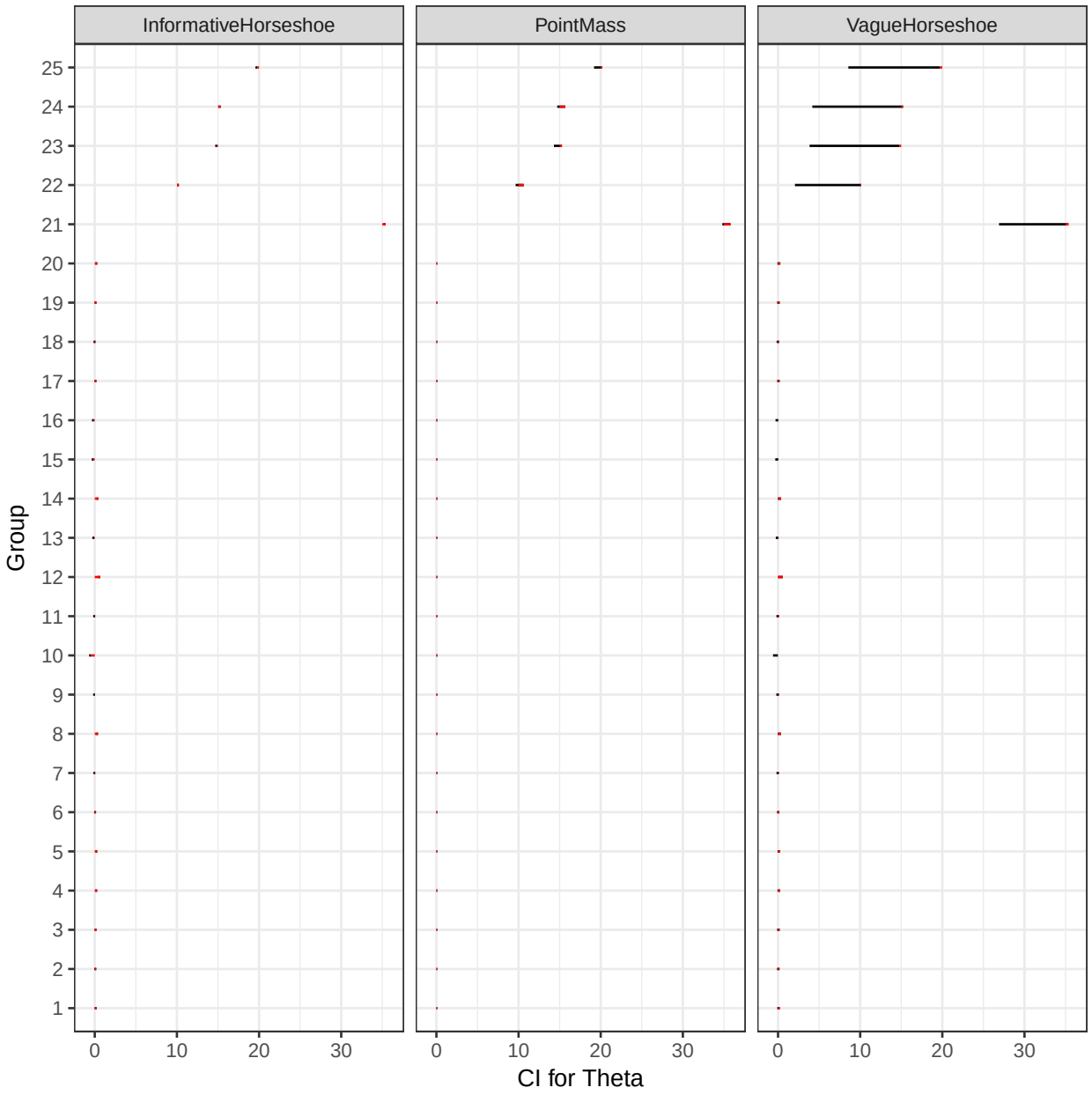

Figure 1: The credible intervals of the groups plotted by each model. The intervals when beta is equal to 0 are fairly tight with the point mass prior basically equaling 0 in its samples. When beta is not zero the bands are still fairly tight with the exception of the vague Horseshoe model whose bands are fairly wide due to the nature of vague proposals in it's algorithm.