---

output: pdf_document

What about finley's code?

```
# attempt to organize data in the same way
dat_small <- read_csv("../data/subsets/dat_small.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   subsection = col_character(),
##   section = col_character(),
##   province = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
finley <- dat_small %>%
  filter(province == "M333") %>%
  group_by(subsection) %>%
  summarize(mean_BIOLIVE_TPA = mean(BIOLIVE_TPA, na.rm = TRUE),
            var_BIOLIVE_TPA = var(BIOLIVE_TPA, na.rm = TRUE),
            mean_forprob = mean(forprob, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
library(sp)
library(coda)
library(MCMCpack)  ##inverse gamma distribution
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2020 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
##=============================================================================
## Load Image of data.frame called "dat" along with neighborhood matrix "R"
##=============================================================================
# load(file = "data.RData")

##-----------
## CONSTANTS
##-----------
##number of small areas (i.e., stands)
m <- 20
##length of each chain
G <- 6000

##----------------------------------
## Multivariate Normal Distribution
##----------------------------------
rmvn <- function(n, mu=0, V = matrix(1)){
  p <- length(mu)
  if(any(is.na(match(dim(V),p))))
    stop("Dimension problem!")
  D <- chol(V)
  t(matrix(rnorm(n*p), ncol=p)%*%D + rep(mu,rep(n,p)))
}

##-----------
## Data
##-----------
sigma.sq <- as.numeric(unlist(finley[, "var_BIOLIVE_TPA"]))
# smooth.sigma.sq <- as.numeric(dat[, "Smooth.Var.AGBM"])

y <- as.matrix(finley[, "mean_BIOLIVE_TPA"], nrow = m)

##Select covariates
X <- finley[, c("subsection", "mean_forprob")]
X <- as.data.frame(X[order(X$subsection, decreasing = FALSE),])
X <- as.matrix(X[,c("mean_forprob")], nrow = m)
##Create Design Matrix
X <- cbind(1, X)

##Starting values for beta from linear model
lm.beta <- lm(mean_BIOLIVE_TPA ~ mean_forprob, data = finley)
beta.start <- as.matrix(coef(lm.beta), nrow = 3)

##-------------------
## Source the models
##-------------------
##Three chains for FH Model
FH.model <- function(y, X, m, sigma.sq, G, beta, a0, b0, sigma.sq.v){
    ##Construct Matrices
    theta.mat <- matrix(nrow = m, ncol = G)
    beta.mat <- matrix(nrow = nrow(beta), ncol = G)
    g.mat <- matrix(nrow = m, ncol = G)
    sigma.sq.v.vec <- vector(mode = "numeric", length = G)

    g <- vector(mode = "numeric", length = m)
    mu <- vector(mode = "numeric", length = m)
    var.t <- vector(mode = "numeric", length = m)
    theta <- matrix(nrow = m, ncol = 1)

    for(i in 1:G){
        ## (1): Draw from Full conditional for theta
        for(j in 1:m){
            g[j] <- sigma.sq.v / (sigma.sq.v + sigma.sq[j])
            mu[j] <- g[j]%*%y[j] + (1 - g[j])%*%t(X[j,])%*%beta
            var.t[j] <- sigma.sq[j]%*%g[j]
            theta[j,1] <- rnorm(1, mu[j], sqrt(var.t[j]))
        }

        ## (2): Draw from Full conditional for beta
        mu.beta <- solve(t(X)%*%X)%*%t(X)%*%theta
        var.beta <- sigma.sq.v*solve(t(X)%*%X)
        beta <- mvrnorm(1, mu.beta, var.beta)

        ## (3): Draw from Full conditional for sigma.sq.v
        shape.v <- a0 + m/2
        scale.v <- b0 + (1/2)*t(theta - X%*%beta)%*%(theta - X%*%beta)
        sigma.sq.v <- rinvgamma(1, shape.v, scale.v)

        ##Parameters to monitor
        theta.mat[,i] <- theta
        beta.mat[,i] <- beta
        sigma.sq.v.vec[i] <- sigma.sq.v
        g.mat[,i] <- g
    }
    out <- list(theta.mat, beta.mat, sigma.sq.v.vec, g.mat)
}
set.seed(19)

FH.chain1 <- FH.model(y = y, X = X, m = m, sigma.sq = sigma.sq, G = G,
             beta = beta.start, a0 = 2, b0 = round(mean(sigma.sq)), sigma.sq.v = 10000)

FH.chain2 <- FH.model(y = y, X = X, m = m, sigma.sq = sigma.sq, G = G,
             beta = beta.start, a0 = 2, b0 = round(mean(sigma.sq)), sigma.sq.v = 1000)

FH.chain3 <- FH.model(y = y, X = X, m = m, sigma.sq = sigma.sq, G = G,
             beta = beta.start, a0 = 2, b0 = round(mean(sigma.sq)), sigma.sq.v = 100)


##-----------------------------------
## SUMMARIZE OUTPUT for theta and cv
##-----------------------------------
```

```
##warm-up
B <- 3000
##post warm-up and thinned samples
sub <- seq(B+1, G, by = 3)
##post warm-up and unthinned samples
sub2 <- seq(B+1, G, by = 1)

##---------------
##FH mcmc objects
##---------------
FH.tsamps.list <- mcmc.list(mcmc(t(FH.chain1[[1]][,sub])),
                            mcmc(t(FH.chain2[[1]][,sub])),
                            mcmc(t(FH.chain3[[1]][,sub])))
FH.tsamps <- rbind(FH.tsamps.list[[1]], FH.tsamps.list[[2]], FH.tsamps.list[[3]])

##posterior mean of theta
FH.theta.mean <- apply(FH.tsamps, 2, mean)

##posterior variance and cv of theta
FH.var.theta <- apply(FH.tsamps, 2, var)
FH.cv.theta <- sqrt(FH.var.theta) / FH.theta.mean
```