````
---
title: "Appendix: Code"
output: html_notebook
---

```{r}

library(rstan)
library(bayesplot)
library(sp)
data(meuse)
library('plotly')
library(lattice)
source("~/Documents/BU/MA578/bslm.R")
```

```{r}
compute_distances <- function(x,y){
  n<- length(x)
  D = matrix(0,nrow=n,ncol=n)
  for(i in 1:n){
    for(j in 1:n){
        p1 = c(x[i],y[i])
        p2 = c(x[j],y[j])
        D[i,j] <- sqrt(sum((p1 - p2)^2))
    }
  }
  return(D)
}

D <- compute_distances(meuse$x,meuse$y)
p <- plot_ly(z=~D)
layout(p,title = "Distances Between Sites", xaxis=list(title =
'Site'),yaxis=list(title = 'Site'))

```

```{r}

p<-plot_ly(x= meuse$x,y=meuse$y,z = ~log(meuse$lead))
layout(p,title = "Log lead concentration as a function of location", scene =
list(xaxis=list(title = 'x location'),yaxis=list(title = 'y
location'),zaxis=list(title = 'Log lead')))

```

Fix tau2 to be 1 to understand impact of phi.
```{r}
````
````

```
phi=1.1
H <- 1*exp(-D/phi)
p <- plot_ly(z=~H)
layout(p,title = "Spatial Correlation for Phi = 1", xaxis=list(title =
'Site'),yaxis=list(title = 'Site'))

phi=300
H <- 1*exp(-D/phi)
p <- plot_ly(z=~H)
layout(p,title = "Spatial Correlation for Phi = 300", xaxis=list(title =
'Site'),yaxis=list(title = 'Site'))

phi=3000
H <- 1*exp(-D/phi)
p <- plot_ly(z=~H)
layout(p,title = "Spatial Correlation for Phi = 3000", xaxis=list(title =
'Site'),yaxis=list(title = 'Site'))
```

```{r}

xyplot(log(lead) ~ dist.m | soil, data = meuse,
       panel = function(...) {
         panel.xyplot(type = "p", ...)
         panel.lmline(lty = 2, ...)
       },xlab='Distance to River (meters)',ylab='Log Lead
Concentration',main='Lead Concentration vs Distance to River by Soil Type')
xyplot(log(lead) ~ dist.m , data = meuse,
       panel = function(...) {
         panel.xyplot(type = "p", ...)
         panel.lmline(lty = 2, ...)
       },xlab='Distance to River (meters)',ylab='Log Lead
Concentration',main='Lead Concentration vs Distance to River')

```


Set up design matrices
```{r}
y <- log(meuse$lead)
X <- model.matrix(~ -1 + soil + soil:dist.m, data = meuse)

J <- length(levels(meuse$soil))
n <- nrow(X)
p <- ncol(X) / J
site_j <- aggregate(. ~ soil, data = meuse, length)
soil_site <- data.frame(group = site_j$soil,
                            coef = c(rep("_intercept", J), rep("_distance",
J)))
X_beta <- model.matrix(~ -1 + coef , data = soil_site)
```

```

Metropolis-Gibbs sampling procedure.  Use 4 chains, 10000 simulations and warmup length of 2000.
```{r}
nchains <- 4
nsims <- 10000
warmup <-2000
gibbs_chain <- matrix(0,ncol = 12,nrow = nsims - warmup)
param_names <-  c("BetaA0","BetaB0", "BetaC0","BetaA1","BetaB1","BetaC1",
                           "Alpha1", "Alpha0", "Kappa2", "Sigma2", "Tau2",
"Phi")
colnames(gibbs_chain)<- param_names
nparams <- length(param_names)
sims <- mcmc_array(nsims - warmup, nchains, param_names)

# Initialize starting values for beta as MLE estimates for log(lead) ~
f(distance to river), not controlling for soil type.
n        <- length(y)
beta_fit <- lm(y ~ X - 1)
beta     <- coef(beta_fit)

# Assuming equal weight of deviance for fit of log lead concentration on error
in estimation and error from spatial component.
sigma2   <- deviance(beta_fit) / (2*n)
tau2     <- deviance(beta_fit) / (2*n)

# Set parameters of inverse gamma for tau2
alpha_tau <- 10
beta_tau  <- 4


#  Compute alpha MLE estimates and use deviance to get inital starting values
for kappa.
alpha_fit <- lm(beta ~ X_beta - 1)
kappa2    <- deviance(alpha_fit) / nrow(X_beta)

# Set parameters of inverse gamma for kappa
alpha_kappa <- 10
beta_kappa  <- 4

# Set parameters of inverse gamma for alpha phi
alpha_phi <- 10
phi       <- 1/rgamma(1,alpha_phi,alpha_phi)

# Flat prior on alpha: normal distribution with 0 mean and infinite variance.
alpha_prior <- list(mean=rep(0,2),precision=rep(0,2))
```
```

Function for H(phi)
```{r}
H <- function(phi){
  return(exp(-D/phi))
}
```

Function for random walk on phi.
```{r}
phi_jumping <- function(phi_mn){
  phi <- rnorm(1,mean=phi_mn, sd=0.1)
  return(phi)
}
```

Function for log posterior.
```{r}

logposterior  <-  function(phi,sigma2,delta,beta,kappa2,tau2,alpha) {
  logpost <-0
  logposterior<- logpost - log(sqrt(sigma2^n)) + (-1/(2*sigma2))*crossprod(y-X
%*% beta-delta) - log(sqrt(det(tau2*(H(phi))))) - (1/2) * t(delta) %*%
solve(tau2 * H(phi)) %*% delta -
    log(sigma2)  - (alpha_phi+1)*log(phi) - (alpha_phi/phi) -
log(sqrt(kappa2^6)) - (1/(2*kappa2)) * crossprod(beta- X_beta %*% alpha)
   - (alpha_tau+1)*log(tau2) - (beta_tau/tau2) - (alpha_kappa+1)*log(kappa2) -
(beta_kappa/kappa2)

  return(logposterior)
}
```

```{r}
for (chain in 1:nchains) {
  gibbs_chain <- matrix(0,ncol = 12,nrow = nsims - warmup)
  for (it in 1:nsims) {
    # 1. Sample alpha
    alpha <- bslm_sample1(beta, X_beta, sqrt(kappa2), alpha_prior)

    # 2. Sample delta
    delta <- rnorm(n,mean=rep(0,n),sd = chol((tau2) * H(phi)) )

    # 3. Sample beta
    beta  <- bslm_sample1(y - delta, X, sqrt(sigma2),
                          list(mean = X_beta %*% alpha, precision = 1 /
kappa2))

    # 4. Sample sigma^2
```

```
    sigma2  <- (n * (1/n * crossprod(y - X %*% beta - delta))/rchisq(1,n))
    sigma2  <- as.vector(sigma2)

    # 5. Sample tau^2
    nu <- alpha_tau + n/2
    s2 <- (2*beta_tau + t(delta) %*% solve(H(phi)) %*% delta)/(n+2*alpha_tau)
    tau2    <- (nu * (s2/rchisq(1,nu)))
    tau2    <- as.vector(tau2)

    # 6. Sample kappa^2
    nAlpha <-6
    nu <- alpha_kappa + nAlpha/2
    s2 <- (2*beta_kappa+ crossprod(beta - X_beta %*% alpha))/
(nAlpha+2*alpha_kappa)
    kappa2 <- (nu * (s2/rchisq(1,nu)))
    kappa2 <- as.vector(kappa2)

    # 7. Sample phi using random walk
    phi_star <- phi_jumping(phi)


    logR <- logposterior(phi_star,sigma2,delta,beta,kappa2,tau2,alpha) -
logposterior(phi,sigma2,delta,beta,kappa2,tau2,alpha)
     if (logR >= 0 || log(runif(1)) <= logR) {
          phi <- phi_star
     }

    # Save iterations
    if(it > warmup){
      sims[it - warmup, chain, ] <- c(beta,alpha,kappa2,sigma2,tau2,phi)
    }

  }
}
```

```{r}
monitor(sims)
mcmc_trace(sims)
mcmc_acf(sims)
mcmc_dens_overlay(sims)
```

```{r}
v<-rbind(sims[,1,],sims[,2,],sims[,3,],sims[,4,])
t(apply(v,2, function(x) quantile(x, c(.025,.25,.5,.75,.975))))
```