Assignment 3

Md Kamrul Hasan Khan

Due 5PM on Monday March 9, 2020

```r
set.seed(77777)
#if (!require("devtools")) install.packages('devtools')
#if (!require("spBayes")) install.packages('spBayes')
#if (!require("rspatial")) devtools::install_github('rspatial/rspatial')
library(sp)
library(tidyverse)
library(ggplot2)
library(mvnfast)
library(spBayes)
library(geoR)
library(splines)
library(mgcv)
```

Homework must be submitted as a single pdf file which is the output of the RMarkdown file. Please save the file as `Last-Name-assignment-3.pdf` and email it to jrtipton@uark.edu. For example, the student with the last name of Fox would submit the file `Fox-assignment-3.pdf`.

The objective of this assignment is to gain experience with MCMC methods for spatial models and to understand the properties of basis-function approaches to modeling spatial data.

Summary of Data

For this example, we will use the *meuse* dataset. The

Problem 1

Write an MCMC sampler for the model

$$\mathbf{y} \sim N\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I} + \tau^2\mathbf{R}(\phi)\right)$$

where $\boldsymbol{\beta}$ is assigned a $N(\mathbf{0}, 10 \times \mathbf{I})$ prior, σ^2 and τ^2 are assigned independent inverse-gamma$(1,1)$ priors and the spatial range parameter ϕ is assigned a uniform$(0,1000)$ prior. Assume the matrix $\mathbf{R}(\phi)$ is a covariance matrix for an exponential covariance function given range parameter ϕ

Make sure you write out the full conditional distributions (either using LaTeXor by hand and including a photo of your derivation) for each of the parameters before writing any MCMC code.

Solution:

Let $\tau^2 = \delta^2\sigma^2$. Now the hierarchical structure becomes:

$$\begin{aligned}
\mathbf{y} &\sim N\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2(\mathbf{I} + \delta^2\mathbf{R}(\phi))\right) \\
\boldsymbol{\beta} &\sim N(\mathbf{0}, 10 \times \mathbf{I}) \\
\sigma^2 &\sim IG(1,1) \\
\delta^2 &\sim IG(1,1) \\
\phi &\sim Uinf(0, 1000)
\end{aligned}$$

Posterior Distributions:

$$\boldsymbol{\beta}|- \quad \sim \quad N(\Sigma_\beta\, \mu_\beta, \Sigma_\beta); \;\; \sigma^2|- \sim IG(a_1, b_1)$$

where

$$\begin{aligned}
\mu_\beta &= \frac{\mathbf{X}^T(\mathbf{I} + \delta^2\mathbf{R}(\phi))^{-1}\mathbf{y}}{\sigma^2} \text{ and } \Sigma_\beta^{-1} = \frac{\mathbf{X}^T(\mathbf{I} + \delta^2\mathbf{R}(\phi))^{-1}\mathbf{X}}{\sigma^2} + \frac{\mathbf{I}}{10}, \\
a_1 &= 1 + \frac{n}{2} \text{ and } b_1 = 1 + \frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{I} + \delta^2\mathbf{R}(\phi))^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2}.
\end{aligned}$$

$$\begin{aligned}
\pi(\delta^2|-) &\propto (\det(\mathbf{I} + \delta^2\mathbf{R}(\phi)))^{-1/2} \exp\left\{ -\frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{I} + \delta^2\mathbf{R}(\phi))^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2\sigma^2}\right\} (\delta^2)^{-2} \exp\left\{-\frac{1}{\delta^2}\right\}, \\
\pi(\phi|-) &\propto (\det(\mathbf{I} + \delta^2\mathbf{R}(\phi)))^{-1/2} \exp\left\{ -\frac{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{I} + \delta^2\mathbf{R}(\phi))^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}{2\sigma^2}\right\}.
\end{aligned}$$

Since the posterior distributions of δ^2 and ϕ are not standard, they are sampled using Metropolish-Hanstings algorithm.

Problem 2

Fit the data from the *meuse* dataset using the your hand-coded MCMC model using the same priors as above.

```r
data("meuse")
glimpse(meuse)
```

```
## Rows: 155
## Columns: 14
## $ x       <dbl> 181072, 181025, 181165, 181298, 181307, 181390, 181165, 181027~
## $ y       <dbl> 333611, 333558, 333537, 333484, 333330, 333260, 333370, 333363~
## $ cadmium <dbl> 11.7, 8.6, 6.5, 2.6, 2.8, 3.0, 3.2, 2.8, 2.4, 1.6, 1.4, 1.8, 1~
## $ copper  <dbl> 85, 81, 68, 81, 48, 61, 31, 29, 37, 24, 25, 25, 93, 31, 27, 86~
## $ lead    <dbl> 299, 277, 199, 116, 117, 137, 132, 150, 133, 80, 86, 97, 285, ~
## $ zinc    <dbl> 1022, 1141, 640, 257, 269, 281, 346, 406, 347, 183, 189, 251, ~
## $ elev    <dbl> 7.909, 6.983, 7.800, 7.655, 7.480, 7.791, 8.217, 8.490, 8.668,~
## $ dist    <dbl> 0.00135803, 0.01222430, 0.10302900, 0.19009400, 0.27709000, 0.~
## $ om      <dbl> 13.6, 14.0, 13.0, 8.0, 8.7, 7.8, 9.2, 9.5, 10.6, 6.3, 6.4, 9.0~
## $ ffreq   <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ soil    <fct> 1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ lime    <fct> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,~
## $ landuse <fct> Ah, Ah, Ah, Ga, Ah, Ga, Ah, Ab, Ab, W, Fh, Ag, W, Ah, Ah, W, W~
## $ dist.m  <dbl> 50, 30, 150, 270, 380, 470, 240, 120, 240, 420, 400, 300, 20, ~
```

For the response variable, use the log-concentration of zinc (*log(zinc)*) and for the covariates use distance (*dist*) along with an intercept term.

```r
set.seed(77777)

library(mvtnorm)
```

```
##
## Attaching package: 'mvtnorm'

## The following objects are masked from 'package:mvnfast':
##
##     dmvt, rmvt
```

```r
library(sp)
library(DAAG)
```

```
## Warning: package 'DAAG' was built under R version 4.0.5

## Loading required package: lattice
```

```r
library(mvnfast)
#####################################################################################
##### Functions

rmult <- function(p) { sum(ceiling(runif(1) - cumsum(p) )) + 1 }

inv_and_logdet.sym <- function(A)
{
  B <- chol(A)
  result <- list()
  result[[1]] <- B
  result[[2]] <- chol2inv(B)
  result[[3]] <- 2*sum(log(diag(B)))
  return(result)
}

#####################################################################################

data(meuse)

meuse$log_zinc <- log(meuse$zinc)

n <- nrow(meuse)  ## how many data points we have

y <- meuse$log_zinc

X <- cbind(rep(1,n), meuse$dist)

p <- ncol(X)

location <- cbind(meuse$x, meuse$y)
```

```
d <- as.matrix(dist(location, diag = TRUE, upper = TRUE))

#phi_choose <- seq(0.01, 1000, length.out = 15)

#phi <- sample(phi_choose, 1)

phi <- runif(1, 0, 1000)

R_phi <- exp(-d/phi)

## set priors

## beta ~ MVN(0, 10 * I)

beta_prior_mean <- matrix(0, p, 1)
beta_prior_var <- 10

## prior for sigma2 is IG(a1,b1)

a10 <- 1
b10 <- 1

## prior for delta2 is IG(a2,b2)

a20 <- 1
b20 <- 1

## initial values of beta and sigma2

sigma2 <- var(y)

delta2 <- 1

inv_logdet <- inv_and_logdet.sym(diag(n) + delta2*R_phi)

var_inv <- inv_logdet[[2]]
var_logdet <- inv_logdet[[3]]

var_MH_delta2 <- 1

accept_delta2 <- 0

var_MH_phi <- 500

accept_phi <- 0

## how many times you want to run the mcmc

nit <- 5000      ## number of initial runs that you want to discard
nmc <- 5000      ## number of runs that you want to do
nthin <- 10    ## you want t thin the sample at what interval to avoid correlation ?

## during the mcmc, you need to store the simulated samples of mu and sigma,
```

```r
## so define storage variables

beta_store_problem_2 <- matrix(0, nmc/nthin, p)
sigma2_store_problem_2 <- c()
delta2_store_problem_2 <- c()
phi_store_problem_2 <- c()
n_log_likelihood_store_problem_2 <- c()

## start the mcmc loop
print(date())
```

```
## [1] "Sun Nov 14 23:50:55 2021"
```

```r
for (iter in 1:(nit+nmc))
{

    ####################################################################################

    ## posterior distribution of beta is normal

    beta_post_dispersion <- inv_and_logdet.sym(t(X)%*%var_inv%*%X/sigma2
                                               + diag(p)/beta_prior_var)[[2]]

    beta_post_mean <-  crossprod( t(beta_post_dispersion),
                     ( (colSums(X*colSums(var_inv*y))/sigma2) +
                         (beta_prior_mean/beta_prior_var) ) )

    ## then simulate sample of beta

    beta <- c(rmvn (1, c(beta_post_mean), beta_post_dispersion))

    X_beta <- c(X%*%beta)

    residual <- y - X_beta

    #####################################################################################

    ## piosterior distribution of sigma2 is inverse gamma

    sum_S2 <- sum(residual*colSums(var_inv*residual) )

    sigma2_post_shape <- a10 + (n/2)

    sigma2_post_rate <- b10  + sum_S2/ 2

    ## then simulate sample of sigma2

    sigma2 <- 1/rgamma (1, shape = sigma2_post_shape, rate = sigma2_post_rate)

    ######################################################################################

    ## undpating delta2 using MH
```

```
delta2_prop = rlnorm (1, log(delta2), var_MH_delta2)

inv_logdet_prop_delta2 <- inv_and_logdet.sym(diag(n) + delta2_prop*R_phi)

var_inv_prop_delta2 <- inv_logdet_prop_delta2[[2]]
var_logdet_prop_delta2 <- inv_logdet_prop_delta2[[3]]

sum_S2_prop_delta2 <- sum(residual*colSums(var_inv_prop_delta2*residual) )

pa1_delta2 = a20* (log(delta2) - log(delta2_prop))

pa2_delta2 = (1/delta2 - 1/delta2_prop)*b20

pa3_delta2 = 0.5*( var_logdet - var_logdet_prop_delta2 )

pa4_delta2 = (sum_S2 - sum_S2_prop_delta2)/(2*sigma2)

pa_delta2 = min(0, pa1_delta2 + pa2_delta2 + pa3_delta2 + pa4_delta2)

if (rexp(1) >  - pa_delta2)
    {
    delta2 <- delta2_prop
    var_inv <- var_inv_prop_delta2
    var_logdet <- var_logdet_prop_delta2
    sum_S2 <- sum_S2_prop_delta2
    accept_delta2 <- accept_delta2 + 1
    }

#####################################################################################

## posterior distribution of phi

phi_prop <- rnorm(1, phi, var_MH_phi)

if (phi_prop > 0 & phi_prop < 1000)
{
R_phi_prop <- exp(-d/phi_prop)

inv_logdet_prop_phi <- inv_and_logdet.sym(diag(n) + delta2*R_phi_prop)

var_inv_prop_phi <- inv_logdet_prop_phi[[2]]
var_logdet_prop_phi <- inv_logdet_prop_phi[[3]]

sum_S2_prop_phi <- sum(residual*colSums(var_inv_prop_phi*residual) )

pa1_phi <- 0.5*(var_logdet - var_logdet_prop_phi)

pa2_phi <- (sum_S2 - sum_S2_prop_phi)/(2*sigma2)

pa_phi <- min(0, pa1_phi + pa2_phi)


if (rexp(1) >  - pa_phi)
```

```r
        {
        phi <- phi_prop
        R_phi <- R_phi_prop
        var_inv <- var_inv_prop_phi
        var_logdet <- var_logdet_prop_phi
        sum_S2 <- sum_S2_prop_phi
        accept_phi <- accept_phi + 1
        }
    } else {
    phi <- phi
    }
    ## Now store the samples in the store vector you already created

    if (iter > nit & (iter - nit)%%nthin==0)
    {
        beta_store_problem_2[(iter - nit)/nthin, ] <- t(beta)

        sigma2_store_problem_2[(iter - nit)/nthin] <- sigma2

        delta2_store_problem_2[(iter - nit)/nthin] <- delta2

        phi_store_problem_2[(iter - nit)/nthin] <- phi

        n_log_likelihood_store_problem_2[(iter - nit)/nthin] <- var_logdet/2 +
              n*log(sigma2)/2 + sum_S2/(2*sigma2)

    }

    if(iter == 1000) print(date())

}
```

```
## [1] "Sun Nov 14 23:51:00 2021"
```

```r
print(date())
```

```
## [1] "Sun Nov 14 23:51:45 2021"
```

```r
(accept_ratio_delta2 = accept_delta2 / (nit+nmc) )
```

```
## [1] 0.2929
```

```r
(accept_ratio_phi = accept_phi / (nit+nmc) )
```

```
## [1] 0.302
```

```r
apply(beta_store_problem_2, 2, median)
```

```
## [1]  6.586075 -2.678850
```

```r
median(sigma2_store_problem_2)
```

```
## [1] 0.1161252
```

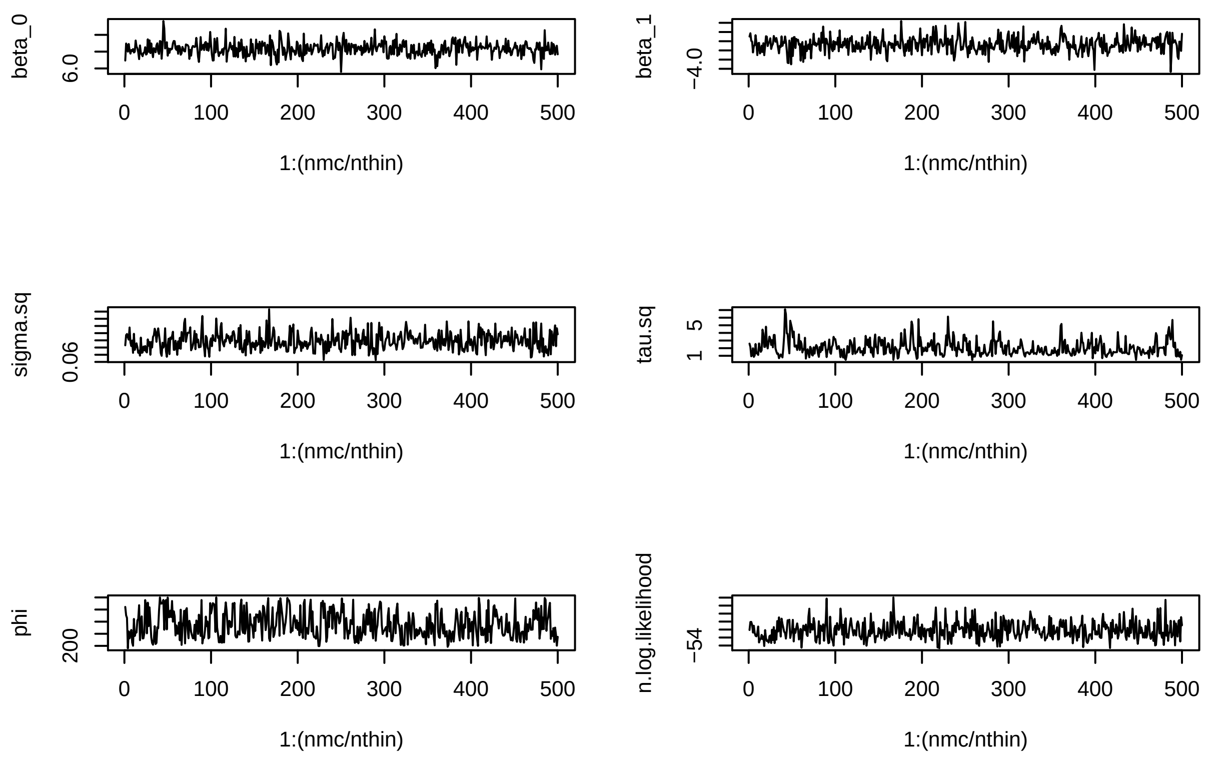
```r
median(delta2_store_problem_2)
```

```
## [1] 1.8273
```

```r
median(phi_store_problem_2)
```

```
## [1] 513.1683
```

```r
par(mfrow = c(3,2))
plot(1:(nmc/nthin), beta_store_problem_2[ , 1], "l", ylab = "beta_0")
plot(1:(nmc/nthin), beta_store_problem_2[ , 2], "l", ylab = "beta_1")
plot(1:(nmc/nthin), sigma2_store_problem_2, "l", ylab = "sigma.sq")
plot(1:(nmc/nthin), delta2_store_problem_2, "l", ylab = "tau.sq")
plot(1:(nmc/nthin), phi_store_problem_2, "l", ylab = "phi")
plot(1:(nmc/nthin), n_log_likelihood_store_problem_2, "l", ylab = "n.log.likelihood")
```



Solution:

Problem 3

Fit the data from the *meuse* dataset using the `spLM` function from the *spBayes* package. Use the same variables and priors as before.

Solution:

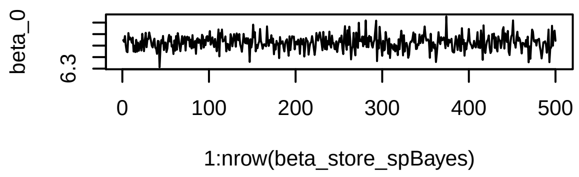
```
n.samples <- 10000
starting <- list("phi"=5, "sigma.sq"=50, "tau.sq"=1)
tuning <- list("phi"=0.1, "sigma.sq"=0.1, "tau.sq"=0.1)
priors.1 <- list("beta.Norm"=list(rep(0,p), diag(10,p)),
"phi.Unif"=c(0.1, 1000), "sigma.sq.IG"=c(1, 1),
"tau.sq.IG"=c(1, 1))
cov.model <- "exponential"
n.report <- 500
coords <- cbind(meuse$x, meuse$y)

fit_model_spBayes <- spLM(log_zinc~ dist, coords=coords, data = meuse, starting=starting,
tuning=tuning, priors=priors.1, cov.model=cov.model,
n.samples=n.samples, verbose=FALSE, n.report=n.report)

fit_model_spBayes <- spRecover(fit_model_spBayes, start=5001, verbose=FALSE, thin=10)

beta_store_spBayes <- fit_model_spBayes$p.beta.recover.samples
theta_store_spBayes <- fit_model_spBayes$p.theta.recover.samples

round(summary(fit_model_spBayes$p.beta.recover.samples)$quantiles[,c(3,1,5)],2)
```

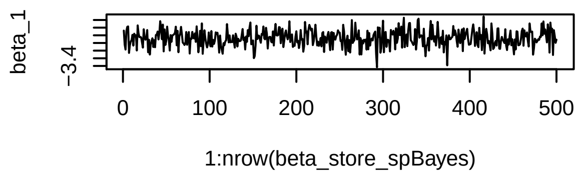
```
##               50%  2.5% 97.5%
## (Intercept)  6.53  6.39  6.66
## dist        -2.68 -3.09 -2.27
```

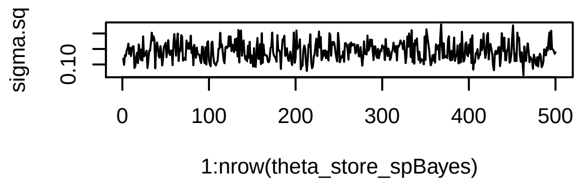
```
round(summary(fit_model_spBayes$p.theta.recover.samples)$quantiles[,c(3,1,5)],2)
```

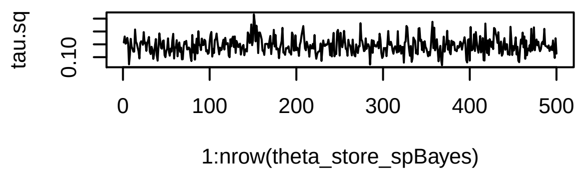
```
##             50%  2.5%  97.5%
## sigma.sq   0.14  0.09   0.20
## tau.sq     0.14  0.09   0.22
## phi      447.25 28.72 897.68
```

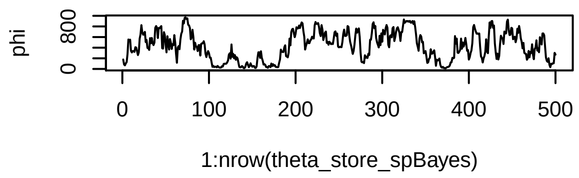
```
par(mfrow = c(3,2))
plot(1:nrow(beta_store_spBayes), beta_store_spBayes[,1], "l", ylab = "beta_0")
plot(1:nrow(beta_store_spBayes), beta_store_spBayes[,2], "l", ylab = "beta_1")
plot(1:nrow(theta_store_spBayes), theta_store_spBayes[,1], "l", ylab = "sigma.sq")
plot(1:nrow(theta_store_spBayes), theta_store_spBayes[,2], "l", ylab = "tau.sq")
plot(1:nrow(theta_store_spBayes), theta_store_spBayes[,3], "l", , ylab = "phi")
```











Problem 4

Fit the data from the *meuse* dataset using the `likfit` function from the *geoR* package.

Solution:

```
fit_model_geoR <- likfit(
data = meuse$log_zinc,
trend = ~ meuse$dist,
coords = cbind(meuse$x, meuse$y),
cov.model = "exponential",
ini.cov.pars = c(var(meuse$log_zinc), 10),
message = FALSE
)

beta_geoR <- fit_model_geoR$beta
beta_geoR
```

```
## intercept    covar1
##   6.59580  -2.81856
```

```
sigma2_geoR <- fit_model_geoR$nugget
sigma2_geoR
```

```
## [1] 0.03091324
```

```
tau2_geoR <- fit_model_geoR$sigmasq
tau2_geoR
```

```
## [1] 0.2297519
```

```
phi_geoR <- fit_model_geoR$phi
phi_geoR
```

```
## [1] 220.8658
```

Problem 5

Now, fit a model using log-concentration of zinc (*log(zinc)*) for the response variable and for the covariates use distance (*dist*) along with an intercept term. use a B-spline expansion expansion of the spatial variables `x` and `y` using the `bs` function from the `splines` library.

Solution:

```
fit_model_splines <- lm(log_zinc ~ dist + bs(x) + bs(y), data = meuse)

parameters_splines <- coef(fit_model_splines)

parameters_splines
```

```
## (Intercept)        dist      bs(x)1      bs(x)2      bs(x)3      bs(y)1
##  6.72259714 -2.39245760 -1.43094062  0.33556859 -1.67882818  0.70581291
##      bs(y)2      bs(y)3
## -0.06132254  1.14000700
```

Problem 6

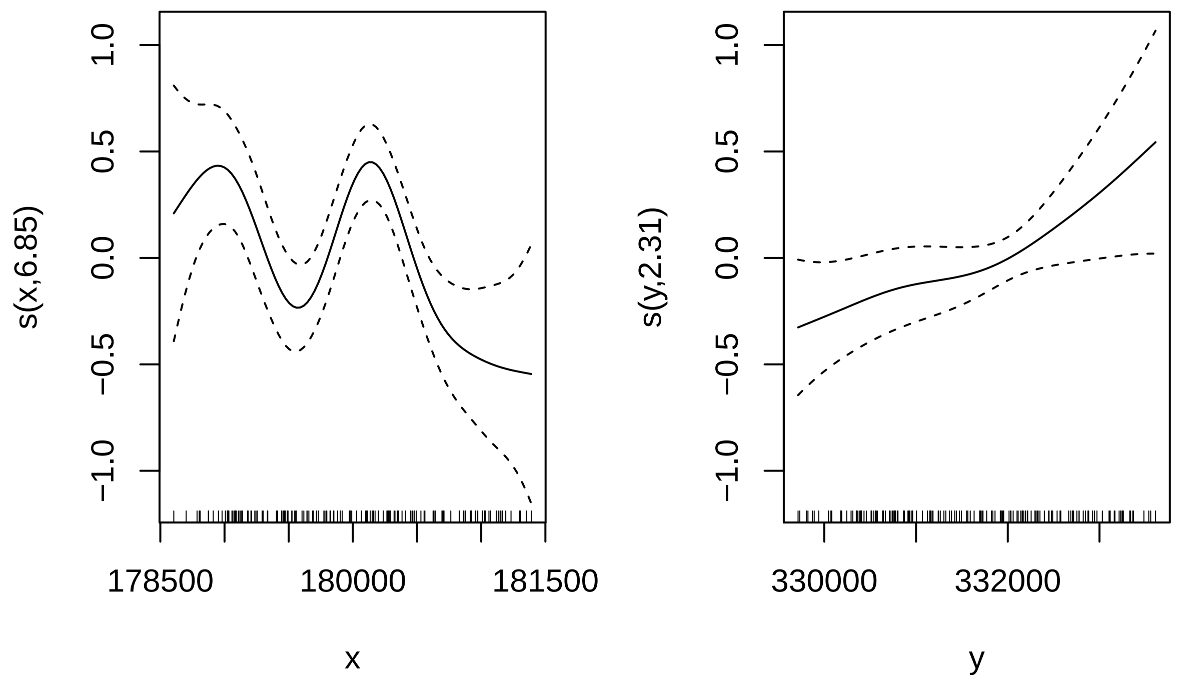
Now, fit a model using log-concentration of zinc (*log(zinc)*) for the response variable and for the covariates use distance (*dist*) along with an intercept term. Use the `gam` function in the `mgcv` library to generate a basis expansion of the spatial variables `x` and `y`.

Solution:

```
fit_model_mgcv <- gam(log_zinc ~ dist + s(x) + s(y), data = meuse, method = "REML" )

parameters_mgcv <- coef(fit_model_mgcv)

parameters_mgcv
```

```
##  (Intercept)         dist       s(x).1       s(x).2       s(x).3       s(x).4
##  6.550217231 -2.768310662 -0.236019009 -0.021409064  0.775095718  0.788367099
##       s(x).5       s(x).6       s(x).7       s(x).8       s(x).9       s(y).1
##  0.160346555  0.127754130 -0.026893446  0.834052295  0.307169428  0.114216688
##       s(y).2       s(y).3       s(y).4       s(y).5       s(y).6       s(y).7
##  0.010753485 -0.012071277  0.061897678  0.021004375 -0.041382662  0.004288046
##       s(y).8       s(y).9
## -0.176742289  0.302119778
```

```r
plot(fit_model_mgcv, page = 1)
```

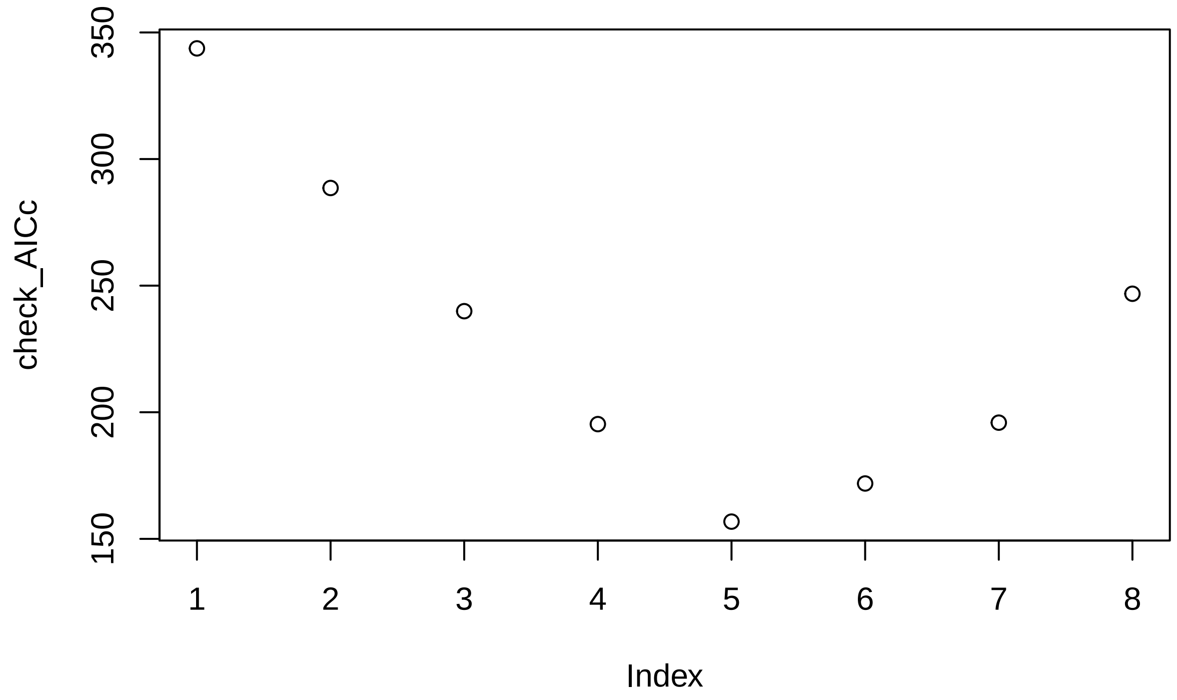


Problem 7

Generate a grid of knot locations and use these knot locations to generate a set of kernels to expand the spatial locations. Then, fit this model using the `lm` function in R using the constructed design matrix using the fixed effects and the kernels.

Solution:

```r
make_kernel_basis <- function(coords, knots, kernel = "gaussian", bandwith = 1, threshold = NULL)
  {
    if (!(kernel %in% c("gaussian", "exponential"))) {
        stop("only kernels available are gaussian and exponential")
    }
    D <- fields::rdist(as.matrix(coords), as.matrix(knots))
    X <- matrix(0, nrow(D), ncol(D))
    if (kernel == "exponential") {
        X <- exp (- D / bandwith)
    } else if (kernel == "gaussian") {
        X <- exp (- D^2 / bandwith)
    }

    ## add in a minimum distance threshold
    if (!is.null(threshold)) {
        X[D > threshold] <- 0
    }

    return(X)
}

check_AICc <- rep(0, 8)
for (j in 1:8) {
    n_knots <- j^2
    knots <- expand.grid(
        seq(min(meuse$x), max(meuse$x), length = sqrt(n_knots)),
        seq(min(meuse$y), max(meuse$y), length = sqrt(n_knots))
    )
    X <- make_kernel_basis(coords, knots, kernel = "exponential", bandwith = 600)
    colnames(X) <- paste0("X", 1:ncol(X))
    X <- data.frame(X)
    fit <- lm(meuse$log_zinc ~ ., data = X)
    check_AICc[j] <- AIC(fit) + (2 * (j^2 + 1)^2 + 2 * (j^2 + 1)) / (nrow(X) - j^2)
}
plot(check_AICc)
```



```
n_knots <- (which(check_AICc == min(check_AICc, na.rm = TRUE)))^2
knots <- expand.grid(
    seq(min(meuse$x), max(meuse$x), length = sqrt(n_knots)),
    seq(min(meuse$y), max(meuse$y), length = sqrt(n_knots))
)
X <- make_kernel_basis(coords, knots, kernel = "exponential", bandwith = 600)
colnames(X) <- paste0("X", 1:ncol(X))
X <- data.frame(X)
fit_model_kernel <- lm(meuse$log_zinc ~ ., data = X)
fit_model_kernel
```

```
##
## Call:
## lm(formula = meuse$log_zinc ~ ., data = X)
##
## Coefficients:
## (Intercept)           X1           X2           X3           X4           X5
##      3.0609       2.4601       2.8945     -14.0378      83.8807    -156.1201
##          X6           X7           X8           X9          X10          X11
##     -0.5406      -0.4130       1.0260      -0.1061      13.5325       8.0506
##         X12          X13          X14          X15          X16          X17
##     -1.2646       2.5074       0.4224       3.9792    -254.4395    -100.5041
##         X18          X19          X20          X21          X22          X23
##      9.1590       0.1092       1.0567    2597.4159    -716.3418      17.2557
##         X24          X25
##      1.2920       0.7701
```

Problem 8

Generate Kriging predictions from the models fitted in problems 2 and 3. Use the *meuse.grid data.frame* to generate predicts at locations given in this data – for the spatial models, use the Kriging formulas, for the B-spline and `mgcv` model, use the `predict()` function. Also generate prediction uncertainty estimates. Compare and contrast the differences/similarities in the model fits from problems 2-8.

```
data("meuse.grid")
glimpse(meuse.grid)
```

```
## Rows: 3,103
## Columns: 7
## $ x      <dbl> 181180, 181140, 181180, 181220, 181100, 181140, 181180, 181220,~
## $ y      <dbl> 333740, 333700, 333700, 333700, 333660, 333660, 333660, 333660,~
## $ part.a <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ part.b <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ dist   <dbl> 0.00000000, 0.00000000, 0.01222430, 0.04346780, 0.00000000, 0.0~
## $ soil   <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ ffreq  <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

```
## prediction from problem 2

X <- cbind(rep(1,n), meuse$dist)

X_pred <- cbind(1, meuse.grid$dist)

location_pred <- cbind(meuse.grid$x, meuse.grid$y)

n <- nrow(location)

n_pred <- nrow(location_pred)

location_all <- rbind(location, location_pred)

d_full <- as.matrix(dist(location_all, diag = TRUE, upper = TRUE))

pred_store <- matrix(0, nmc/nthin, nrow(meuse.grid))

for (i in 1:(nmc/nthin))
{

  beta_store_problem_2[(iter - nit)/nthin, ] <- t(beta)

  sigma2_store_problem_2[(iter - nit)/nthin] <- sigma2

  delta2_store_problem_2[(iter - nit)/nthin] <- delta2

  phi_store_problem_2[(iter - nit)/nthin] <- phi


  beta <- beta_store_problem_2[i, ]
  sigma2 = sigma2_store_problem_2[i]
  delta2 <- delta2_store_problem_2[i]
  phi <- phi_store_problem_2[i]

  var_full_mat <- sigma2*(diag(n+n_pred) + delta2*exp(-d_full/phi) )

  var_mat_1_inv <- inv_and_logdet.sym(var_full_mat[1:n, 1:n])[[2]]

  var_mat_21 = t(var_full_mat[1:n,(n+1):(n+n_pred)])

  var_mat_2 = var_full_mat[(n+1):(n+n_pred),(n+1):(n+n_pred)]

  mean_pred = c(X_pred%*%beta) + c( crossprod(t(crossprod(t(var_mat_21),
             var_mat_1_inv)),(meuse$log_zinc - c(X%*%beta))))

  var_pred = var_mat_2 - crossprod(t(crossprod(t(var_mat_21), var_mat_1_inv)),t(var_mat_21))

  pred_store[i, ] = c(rmvn(1, mean_pred, var_pred))

}
```

```
meuse.grid$preds_mean_MCMC <- apply(pred_store, 2, mean)
meuse.grid$preds_var_MCMC <- apply(pred_store, 2, var)
```

```
## prediction using spBayes

coord_pred <- cbind(meuse.grid$x, meuse.grid$y)

preds_spBayes <- spPredict( fit_model_spBayes,
  start = 5001,
  thin = 10,
  pred.coords = coord_pred,
  pred.covars = cbind(1, meuse.grid$dist))
```

```
## ----------------------------------------
##  General model description
## ----------------------------------------
## Model fit with 155 observations.
##
## Prediction at 3103 locations.
##
## Number of covariates 2 (including intercept if specified).
##
## Using the exponential spatial correlation model.
##
## -------------------------------------------------
##      Sampling
## -------------------------------------------------
## Sampled: 100 of 500, 19.80%
## Sampled: 200 of 500, 39.80%
## Sampled: 300 of 500, 59.80%
## Sampled: 400 of 500, 79.80%
## Sampled: 500 of 500, 99.80%
```

```r
## Calculate the prediction means and variances
meuse.grid$preds_mean_spBayes <- apply(preds_spBayes$p.y.predictive.samples, 1, mean)
meuse.grid$preds_var_spBayes <- apply(preds_spBayes$p.y.predictive.samples, 1, var)

## calculate 95% credible intervals for the predictions
pred_summary_spBayes <- apply(preds_spBayes$p.y.predictive.samples, 1, function(x) {
  quantile(x, prob = c(0.025, 0.5, 0.975))
})
```

```r
### prediction using geoR

preds_geoR <- krige.conv(as.geodata(meuse[, c(1,2,15)]),
                         locations  = cbind(meuse.grid$x, meuse.grid$y),
                         krige = krige.control(obj.model = fit_model_geoR))
```

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```r
meuse.grid$preds_mean_geoR <- preds_geoR$predict
meuse.grid$preds_var_geoR <- preds_geoR$krige.var
```

```r
### prediction using spline

preds_splines <- predict(fit_model_splines, meuse.grid, se.fit = TRUE)
```

```
## Warning in bs(x, degree = 3L, knots = numeric(0), Boundary.knots = c(178605, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(y, degree = 3L, knots = numeric(0), Boundary.knots = c(329714, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

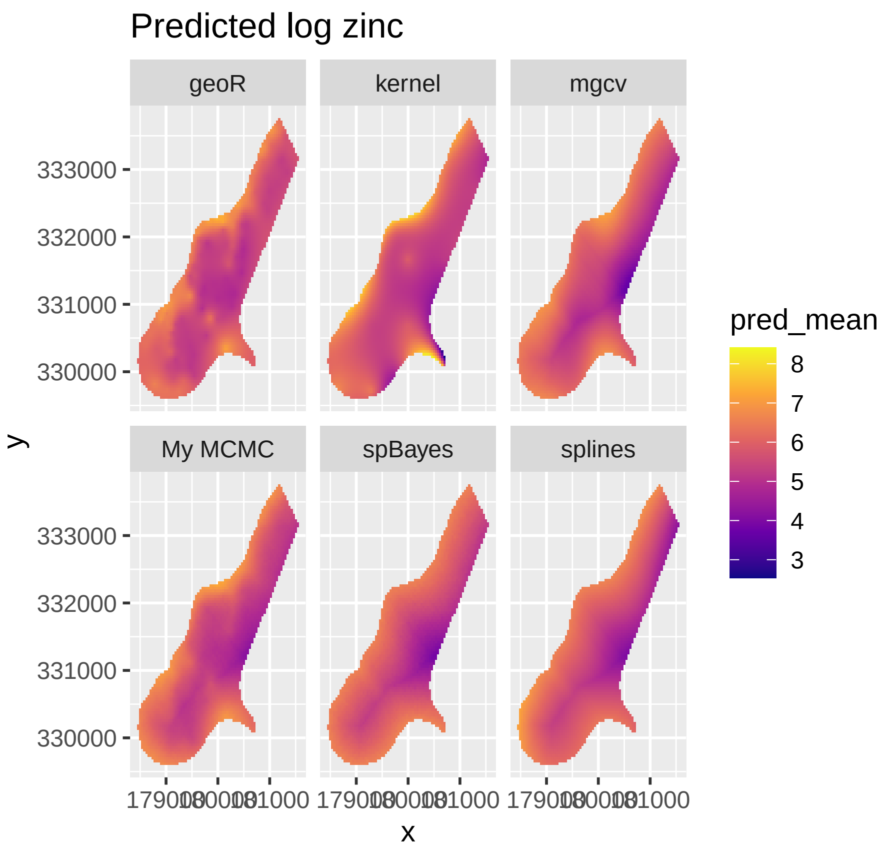
```r
meuse.grid$preds_mean_splines <- unname(preds_splines$fit)
meuse.grid$preds_var_splines <- unname((preds_splines$se.fit)^2)
```

```r
### prediction using kernel

X_pred <- make_kernel_basis(cbind(meuse.grid$x, meuse.grid$y),
                    knots, kernel = "exponential", bandwith = 600)

colnames(X_pred) <- paste0("X", 1:ncol(X_pred))

X_pred <- data.frame(X_pred)

preds_kernel <- predict(fit_model_kernel, newdata = X_pred, se.fit = TRUE)
meuse.grid$preds_mean_kernel <- preds_kernel$fit
meuse.grid$preds_var_kernel <- preds_kernel$se.fit^2
```

```r
#########################################################################################

### prediction using mgcv
```

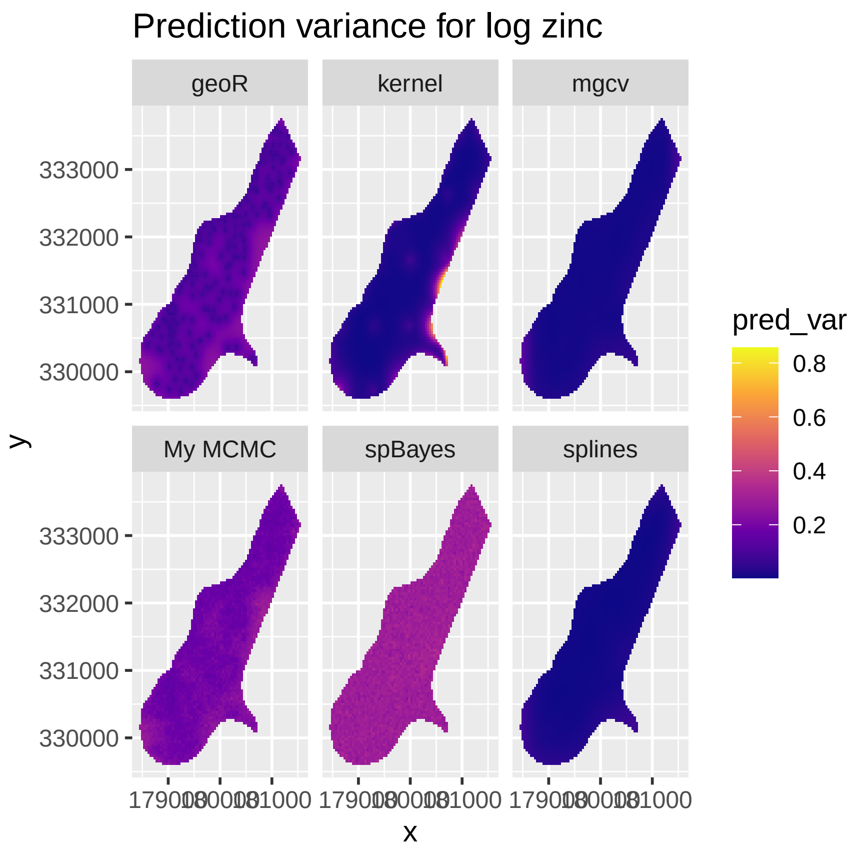
```r
preds_mgcv <- predict(fit_model_mgcv, meuse.grid, se.fit = TRUE)

meuse.grid$preds_mean_mgcv <- unname(preds_mgcv$fit)
meuse.grid$preds_var_mgcv <- unname((preds_mgcv$se.fit)^2)
```

```r
plot_data <- data.frame(x = rep(meuse.grid$x, 6),
                        y = rep(meuse.grid$y, 6),
                        pred_mean = c(meuse.grid$preds_mean_MCMC,
                                      meuse.grid$preds_mean_spBayes,
                                      meuse.grid$preds_mean_geoR,
                                      meuse.grid$preds_mean_splines,
                                      meuse.grid$preds_mean_kernel,
                                      meuse.grid$preds_mean_mgcv),
                        pred_var = c(meuse.grid$preds_var_MCMC,
                                      meuse.grid$preds_var_spBayes,
                                      meuse.grid$preds_var_geoR,
                                      meuse.grid$preds_var_splines,
                                      meuse.grid$preds_var_kernel,
                                      meuse.grid$preds_var_mgcv),
                        models = rep(c("My MCMC", "spBayes", "geoR", "splines",
                                       "kernel", "mgcv"), each = nrow(meuse.grid)))
ggplot(data = plot_data, aes(x = x, y = y, fill = pred_mean)) +
  geom_raster() +
  scale_fill_viridis_c(option = "plasma") +
  facet_wrap(~models, ncol = 3) +
  ggtitle("Predicted log zinc")  +
  coord_fixed(xlim = range(meuse.grid$x), ylim = range(meuse.grid$y), ratio = 1.3)
```



The above figures represent the predicted log zinc using 6 candidate models. All the plots are almost similar. I also want to notify that I don't believe the geoR is using "dist" covariate in the prediction.

```r
ggplot(data = plot_data, aes(x = x, y = y, fill = pred_var)) +
  geom_raster() +
  scale_fill_viridis_c(option = "plasma") +
  facet_wrap(~models, ncol = 3) +
  ggtitle("Prediction variance for log zinc")  +
  coord_fixed(xlim = range(meuse.grid$x), ylim = range(meuse.grid$y), ratio = 1.3)
```



The above figures depict the prediction variances of log zinc. These figure show that the prediction variance is highest in almost all data points using spBayes model and the largest variance arises in right middle part of the kernel model.