Methods

Marshall Honaker

First, let's import the necessary packages:

```
library(fields)
```

```
## Loading required package: spam

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.5-1 (2019-12-12) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```
library(RandomFields)
```

```
## Loading required package: sp

## Loading required package: RandomFieldsUtils

##
## Attaching package: 'RandomFields'

## The following object is masked from 'package:RandomFieldsUtils':
##
##     RFoptions
```

```
library(mvtnorm)
library(ggplot2)
library(matrixcalc)
```

Wu and Narisetty (2020) outline a score based likelihood approach to Bayesian multiple quantile regression. Here we provide R code for a modified version of this approach adapted for spatial data. Annotations will be made/provided to explain the ways in which the approach taken by Wu & Narisetty was modified to accomdodate spatially dependent data.

To begin, we will define so of the more basic, necessary functions. First, we will define the score function.

Basic Functions & Methods

In the aforementioned paper, the score function is given by: $s_\tau(\beta) = \sum_{i=1}^{n} x_i \psi_\tau(y_i - x_i^T \beta)$, where $\psi_\tau(u) = \tau - I_{\{u<0\}}(u)$.

```
get_score<- function(y, X, tau, beta){
  n <- nrow(X)
  temp <- t(X[1,,drop=FALSE])*ifelse(y[1] - X[1,,drop=FALSE]%*%as.matrix(beta) < 0, tau  - 1, tau)[1,1]
  for(i in 2:n){
    temp <- temp + t(X[i,,drop=FALSE])*ifelse(y[i] - X[i,,drop=FALSE]%*%as.matrix(beta) < 0, tau  - 1, 
  }
  return(temp)
}
```

I'm also going to define a function to obtain the spatial covaraince matrix. This isn't necessary, but it will make the code later on more readable.

```
get_spatial_covar_mat <- function(locs){  ## outputs nxn spatial covariance matrix
  dist <- rdist(locs)
  return(exp(-dist/1.5))
}
```

Next, we will write a method to obtain a spatially adapted version of the working likelihood function proposed in the paper. The function proposed in the paper is given by $L(Y|X,\beta) = C \exp\left(-\frac{1}{2n} s_\tau(\beta)^T W s_\tau(\beta)\right)$ where W is a p x p positive definite weight matrix. To account for the spatial variability/dependence within the data, we will instead use following quantity inspired by the Mahalanobis Distance: $(X-\hat{\mu})^T \Sigma^{-1}(X-\hat{\mu})$, where Σ^{-1} is the inverse of the spatial covariance matrix.

However, it is important to note that while the weight matrix proposed by the paper is positive definite, the quantity we use to replace it is only positive semi-definite. We have also omitted the exponentiation of the kernel of the likelihood function. (This yields sensible results when working with the Importance Sampling algorithm defined later on, but yields extremely large values which is very strange for a likelihood function.)

```
get_likelihood <- function(y, X, tau, beta, locs, C){
  ## Outputs 1x1 scalar that is the likelihood
  score <- get_score(y, X, tau, beta)
  X_centered <- X - colMeans(X)
  coef <- 1/(2*length(y))
  kernel <- (coef*(t(score)%*%t(X_centered)%*%solve(get_spatial_covar_mat(locs))%*%X_centered%*%score)[
  return(C*kernel)
}
```

Test Basic Methods

Now that we have defined the basic functions, let's test them to make sure they work.

```
response <- rnorm(500)
feature_mat <- cbind(rnorm(500), rnorm(500))
t <- .5
B <- c(1, 1)
l <- cbind(runif(500, 0, 10), runif(500, 0, 10))

get_score(response, feature_mat, t, B)
```

```
##            [,1]
## [1,] -113.3614
## [2,] -105.7253
```

```r
get_likelihood(response, feature_mat, t, B, l, 1)
```

```
## [1] 84739.43
```

Everything seems to be working alright but we should keep an eye on the likelihood function. It seems to work for now, but the magnitude of the values produced is very strange for a likelihood function.

Ada Importance Sampling for a Single Quantile Level

To approximate the posterior distribution of the model parameters, Wu & Narisetty implement an Importance Sampling (IS) procedure described on pgs. 13-16. The following code implements this IS algorithm. It runs, but we still need to check the accuracy and efficiency of the results.

```r
get_updated_params <- function(y, X, tau, beta, Sigma, locs, draw){
  ## outputs list with elements mu (a p-length vector of estimated means) and S (the estimated pxp cova
  n <- length(y)
  p <- ncol(X)

  w <- get_likelihood(y, X, tau, beta, locs, 1)*(1/(2*n)^p)/dmvnorm(draw, beta, Sigma)

  mu_hat <- apply(w*draw, 2, sum)/sum(w)

  S <- matrix(nrow = p, ncol = p)
  for(i in 1:p){
    for(j in 1:p){
      S[i,j] <- sum(w*draw[,i]*draw[,j])/sum(w) - mu_hat[i]*mu_hat[j]
    }
  }
  temp <- list(mu_hat, S)
  names(temp) <- c("mu", "S")
  return(temp)
}

get_effective_samp_size <- function(y, X, tau, beta, Sigma, locs, draw, M){
  ## Obtains Effective Sample Size
  p <- ncol(X)
  n <- length(y)
  w <- get_likelihood(y, X, tau, beta, locs, 1)*(1/(2*n)^p)/dmvnorm(draw, beta, Sigma)
  cv_sq <- ((M - 1)^(-1))*sum((w - mean(w))^2)/mean(w)^2
  return(M/(1 + cv_sq))
}

adaIS_singleQuantile <- function(y, X, tau, C, locs, M, num_reps){
  ## outputs a list with $mu (a p-length vector of estimated means) and $S (the estimated pxp covarianc
  p <- ncol(X) ## Step 1: Initialize starting values for IS algorithm
  n <- length(y) ## Step 1: Initialize starting values for IS algorithm
  beta <- coef(lm(y ~ X))[-1] ## Step 1: Initialize starting values for IS algorithm
  S <- cov(X) ## Step 1: Initialize starting values for IS algorithm

  params <- list(beta, S)  ## Put our initial parameter estimates/starting values into a list called pa
  names(params) <- c("mu", "S")

  for(i in 1:num_reps){ ## Step 4: Repeat steps 2 and 3 until we achieve the desired effective sample s
    draws <- rmvnorm(M, params$mu, params$S)  ## Step 2: Simulate M values from the proposal distributi
```

```
    params <- get_updated_params(y, X, tau, params$mu, params$S, locs, draws) ## Step 3: Update the par
  }
  return(params)
}
```

Test IS Algorithm

```
m <- 10000
S_0 <- cov(feature_mat)
initial_draw <- rmvnorm(m, B, S_0)

param_test <- get_updated_params(response, feature_mat, t, B, S_0, l, initial_draw)
param_test  ## Check updated parameter values
```

```
## $mu
## [1] 0.0232726 0.2846963
##
## $S
##           [,1]      [,2]
## [1,] 5.4052855 0.8506654
## [2,] 0.8506654 8.9217336
```

```
is.positive.definite(round(param_test$S, 7)) ## Make sure updated covariance matrix is positive definit
```

```
## [1] TRUE
```

```
posterior_param_test <- adaIS_singleQuantile(response, feature_mat, t, 1, l, m, 10)
posterior_param_test ## Check posterior parameter values
```

```
## $mu
##         X1         X2
## -124.75168  -80.74227
##
## $S
##           [,1]      [,2]
## [1,] 4350868.9 -799001.1
## [2,] -799001.1 1031522.8
```

```
is.positive.definite(round(posterior_param_test$S, 7)) ## Make sure posterior covaraince matrix is posi
```

```
## [1] TRUE
```

```
get_effective_samp_size(response, feature_mat, t, B, cov(feature_mat), l, initial_draw, m)
```

```
## [1] 6.293178
```

So everything seems to be working, though it does need a little tuning/refinement.

Simulation Study

Now let's put everything together to conduct a more thorough simulation study.

```
## First, let's simulate the locations
n <- 1000
locs <- as.data.frame(cbind(runif(n, 0, 100), runif(n, 0, 100)))
colnames(locs) <- c("long", "lat")
```

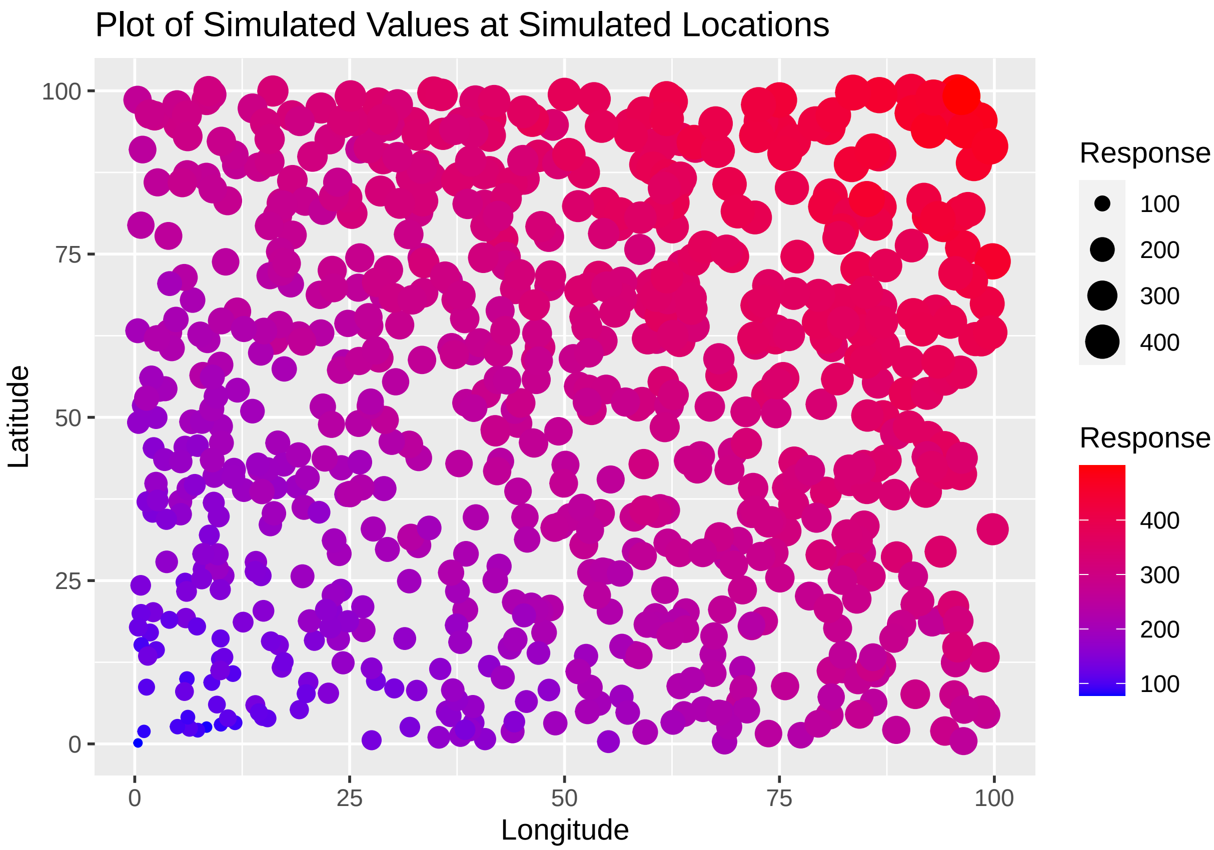
```r
## Next, we will simulate the spatially dependent values
sim_mod <- RMexp(var = 4, scale = 3) +
  RMnugget(var = .75) +
  RMtrend(mean = 1.5)
sim_vals <- RFsimulate(sim_mod, x = locs$long, y = locs$lat)
```

```
## New output format of RFsimulate: S4 object of class 'RFsp';
## for a bare, but faster array format use 'RFoptions(spConform=FALSE)'.
```

```r
## Now, we will simulate the values of the response and the covariates
beta <- c(2, 2, 5, 5, 5, 5)
X_sim <- cbind(rnorm(n, 10, 2), rexp(n, 5), rbeta(n, 5, 1), runif(n, 2.5, 7.5))
y_sim <- beta[1]*locs$long + beta[2]*locs$lat +
  beta[3]*X_sim[,1] + beta[4]*X_sim[,2] + beta[5]*X_sim[,3] + beta[6]*X_sim[,4]

## Next, we will split the data into training and test sets (75% of simulated values will be allocated
train_indices <- sample(1:n, round(.75*n), replace = FALSE)
X_train <- X_sim[train_indices,]
X_test <- X_sim[-train_indices,]
y_train <- y_sim[train_indices]
y_test <- y_sim[-train_indices]
locs_train <- locs[train_indices,]
locs_test <- locs[-train_indices,]

## Now, to plot the training data
sim_data <- as.data.frame(cbind(y_train, locs_train))
sim_plot <- ggplot(data = sim_data, aes(x = long, y = lat)) + geom_point(aes(col = y_train, size = y_tra
  labs(title = "Plot of Simulated Values at Simulated Locations",
       x = "Longitude",
       y = "Latitude",
       col = "Response",
       size = "Response") +
  scale_color_gradient(low = "blue", high = "red", name = "Response")
plot(sim_plot)
```




```
## Now that we have the simulated data, we can check out the performance of our functions for a variety
tau <- c(.01, .05, .1, .25, .5, .75, .9, .95, .99)
b <- beta[3:length(beta)]

score_mat <- matrix(nrow = length(b), ncol = length(tau))
for(i in 1:ncol(score_mat)){
  score_mat[,i] <- get_score(y_train, X_train, tau[i], b)
}
score_mat
```

```
##            [,1]       [,2]      [,3]       [,4]       [,5]      [,6]     [,7]
## [1,] 75.226646 376.133228 752.26646 1880.66614 3761.33228 5641.9984 6770.398
## [2,]  1.498511   7.492556  14.98511   37.46278   74.92556  112.3883  134.866
## [3,]  6.262412  31.312058  62.62412  156.56029  313.12058  469.6809  563.617
## [4,] 37.758348 188.791739 377.58348  943.95870 1887.91739 2831.8761 3398.251
##            [,8]      [,9]
## [1,] 7146.5313 7447.4379
## [2,]  142.3586  148.3526
## [3,]  594.9291  619.9787
## [4,] 3587.0430 3738.0764
```

```
likelihood_vec <- rep(NA, length(tau))
for(i in 1:length(tau)){
  likelihood_vec[i] <- get_likelihood(y_train, X_train, tau[i], b, locs_train, 1)
}
likelihood_vec
```

```
## [1]     108483     2712075    10848299    67801867   271207468   610216803   878712196
```

```
## [8]  979058959 1063241757
```

```r
draw <- rmvnorm(n, b, cov(X_train))
mu_mat <- matrix(nrow = length(b), ncol = length(tau))
S_list <- list()
param_list <- list(mu_mat, S_list)
names(param_list) <- c("mu", "S")
for(i in 1:length(tau)){
  x <- get_updated_params(y_train, X_train, tau[i], b, cov(X_train), locs_train, draw)
  param_list$mu[,i] <- x$mu
  param_list$S[[i]] <- x$S
  draw <- rmvnorm(n, x$mu, x$S)
}
param_list
```

```
## $mu
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 4.757638  1.991110  1.980434  1.969521  1.962239  1.955648  1.945760
## [2,] 5.166620  6.463510  6.463333  6.463174  6.463034  6.462935  6.462739
## [3,] 5.007925  5.639286  5.639781  5.640346  5.640717  5.641076  5.641589
## [4,] 6.858601 11.904621 11.909634 11.914034 11.917672 11.920338 11.925292
##           [,8]      [,9]
## [1,]  1.939269  1.931882
## [2,]  6.462630  6.462511
## [3,]  5.641885  5.642220
## [4,] 11.928451 11.932302
##
## $S
## $S[[1]]
##             [,1]        [,2]        [,3]        [,4]
## [1,]  5.68904348  0.46656787 -0.04288845 -1.79851520
## [2,]  0.46656787  0.22550445  0.01775603 -0.11209739
## [3,] -0.04288845  0.01775603  0.02523979  0.06999089
## [4,] -1.79851520 -0.11209739  0.06999089  7.54706330
##
## $S[[2]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.182369e-03  3.721403e-05 -1.058609e-04 -1.023306e-03
## [2,]  3.721403e-05  8.263177e-07 -1.653501e-06 -2.179668e-05
## [3,] -1.058609e-04 -1.653501e-06  5.780205e-06  4.314968e-05
## [4,] -1.023306e-03 -2.179668e-05  4.314968e-05  6.120057e-04
##
## $S[[3]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.439141e-03  4.147514e-05 -1.195028e-04 -1.128847e-03
## [2,]  4.147514e-05  8.952829e-07 -1.898542e-06 -2.335619e-05
## [3,] -1.195028e-04 -1.898542e-06  6.481019e-06  4.931200e-05
## [4,] -1.128847e-03 -2.335619e-05  4.931200e-05  6.467157e-04
##
## $S[[4]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.331964e-03  3.933909e-05 -1.143826e-04 -1.077306e-03
## [2,]  3.933909e-05  8.659186e-07 -1.769898e-06 -2.268374e-05
## [3,] -1.143826e-04 -1.769898e-06  6.291354e-06  4.591860e-05
## [4,] -1.077306e-03 -2.268374e-05  4.591860e-05  6.339241e-04
```

```
##
## $S[[5]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.372941e-03  4.056252e-05 -1.168263e-04 -1.104902e-03
## [2,]  4.056252e-05  9.028765e-07 -1.820936e-06 -2.363733e-05
## [3,] -1.168263e-04 -1.820936e-06  6.477191e-06  4.679329e-05
## [4,] -1.104902e-03 -2.363733e-05  4.679329e-05  6.602476e-04
##
## $S[[6]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0023482598  3.858480e-05 -1.165043e-04 -1.065811e-03
## [2,]  0.0000385848  8.452391e-07 -1.743153e-06 -2.222039e-05
## [3,] -0.0001165043 -1.743153e-06  6.482049e-06  4.549226e-05
## [4,] -0.0010658112 -2.222039e-05  4.549226e-05  6.272129e-04
##
## $S[[7]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0021351647  3.456760e-05 -1.060257e-04 -9.504017e-04
## [2,]  0.0000345676  7.688382e-07 -1.548474e-06 -2.014467e-05
## [3,] -0.0001060257 -1.548474e-06  5.977001e-06  3.959220e-05
## [4,] -0.0009504017 -2.014467e-05  3.959220e-05  5.726354e-04
##
## $S[[8]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  2.053212e-03  3.334292e-05 -1.011458e-04 -9.191105e-04
## [2,]  3.334292e-05  7.645959e-07 -1.462496e-06 -2.001623e-05
## [3,] -1.011458e-04 -1.462496e-06  5.706808e-06  3.728420e-05
## [4,] -9.191105e-04 -2.001623e-05  3.728420e-05  5.697935e-04
##
## $S[[9]]
##               [,1]          [,2]          [,3]          [,4]
## [1,]  0.0021133275  3.407050e-05 -1.046872e-04 -9.418707e-04
## [2,]  0.0000340705  7.723440e-07 -1.528043e-06 -2.007296e-05
## [3,] -0.0001046872 -1.528043e-06  5.900520e-06  3.917930e-05
## [4,] -0.0009418707 -2.007296e-05  3.917930e-05  5.675747e-04
```

```r
M <- 10000
draw <- rmvnorm(n, b, cov(X_train))
num_iters <- 10
post_mu_mat <- matrix(nrow = length(b), ncol = length(tau))
post_S_list <- list()
post_ess <- rep(NA, length(tau))
post_param_list <- list(post_mu_mat, post_S_list, post_ess)
names(post_param_list) <- c("mu", "S", "ESS")
for(i in 1:length(tau)){
  temp <- adaIS_singleQuantile(y_train, X_train, tau[i], 1, locs_train, M, num_iters)
  post_param_list$mu[,i] <- temp$mu
  post_param_list$S[[i]] <- temp $S
  post_param_list$ESS[i] <- get_effective_samp_size(y_train, X_train, tau[i], b, cov(X_train), locs_tra
}
post_param_list
```

```
## $mu
##           [,1]       [,2]      [,3]       [,4]      [,5]        [,6]
## [1,]  240.99996 647.361510 -91.80860 -104.37047 360.64777 -1188.18298
```

```
## [2,]  -27.65949   3.026059  59.70562  -15.38500 -33.54343   -17.55109
## [3,]   26.44417  32.585519  84.04718   45.85093  50.45152    85.24936
## [4,] -183.24619 372.582349 407.60083   25.00017 109.82512  -636.26185
##             [,7]       [,8]      [,9]
## [1,] -353.495222 202.221398 -171.1666
## [2,]  -76.517404  -3.136762 -114.9209
## [3,]   -4.141158  33.648410  139.9208
## [4,]  -44.207387 -17.823285 -378.3938
##
## $S
## $S[[1]]
##             [,1]       [,2]        [,3]        [,4]
## [1,] 102280.6280  408.55060 -4660.57037 -75584.8579
## [2,]    408.5506   53.59516   -20.99961   -235.0363
## [3,]  -4660.5704  -20.99961   249.42160   3458.9593
## [4,] -75584.8579 -235.03632  3458.95931  56217.0175
##
## $S[[2]]
##            [,1]       [,2]       [,3]       [,4]
## [1,] 1583445.44 46935.5277 27853.8273 535469.043
## [2,]   46935.53  1627.0969   660.8243  16271.433
## [3,]   27853.83   660.8243  1383.4042   8782.752
## [4,]  535469.04 16271.4332  8782.7523 235468.892
##
## $S[[3]]
##             [,1]        [,2]      [,3]      [,4]
## [1,]  87568.7653 -12854.7017 -693.1696 5567.7550
## [2,] -12854.7017   2286.9636  231.3043  226.9055
## [3,]   -693.1696    231.3043  111.4614  774.6629
## [4,]   5567.7550    226.9055  774.6629 8752.3763
##
## $S[[4]]
##            [,1]      [,2]       [,3]      [,4]
## [1,] 487444.737 -1081.596 -23545.899 -86493.76
## [2,]  -1081.596  3220.434   2815.771  39394.88
## [3,] -23545.899  2815.771   4059.999  39596.21
## [4,] -86493.765 39394.877  39596.212 630118.28
##
## $S[[5]]
##             [,1]      [,2]       [,3]       [,4]
## [1,] 4064336.550 -5699.002 143354.827 -117574.21
## [2,]   -5699.002  3686.049  -1759.444   19027.16
## [3,]  143354.827 -1759.444   8668.907   15955.83
## [4,] -117574.206 19027.159  15955.826  624009.08
##
## $S[[6]]
##             [,1]       [,2]       [,3]       [,4]
## [1,] 1277281.550 -9465.2782 -78535.882 449223.519
## [2,]   -9465.278   550.1074   1035.208   4412.167
## [3,]  -78535.882  1035.2080   7504.343 -32300.460
## [4,]  449223.519  4412.1670 -32300.460 439647.492
##
## $S[[7]]
##           [,1]      [,2]      [,3]      [,4]
```

```
## [1,] 588414.42 61326.449 56007.632 456005.07
## [2,]  61326.45  7116.462  6471.551  47693.93
## [3,]  56007.63  6471.551  6580.209  45471.95
## [4,] 456005.07 47693.933 45471.951 469706.45
##
## $S[[8]]
##             [,1]      [,2]       [,3]        [,4]
## [1,]  3382583.85 210119.47  85862.666 -1134409.62
## [2,]   210119.47  18870.24   5757.460   -29427.26
## [3,]    85862.67   5757.46   4264.229   -37527.17
## [4,] -1134409.62 -29427.26 -37527.173   863389.32
##
## $S[[9]]
##           [,1]      [,2]       [,3]      [,4]
## [1,] 487502.35  76966.68  -65167.98  318492.9
## [2,]  76966.68  37445.54  -37090.94  144155.8
## [3,] -65167.98 -37090.94   37646.76 -142685.8
## [4,] 318492.85 144155.81 -142685.80  588153.3
##
##
## $ESS
## [1] 952.72 952.72 952.72 952.72 952.72 952.72 952.72 952.72 952.72
```

Issues so Far

In developing this novel approach to Bayesian spatial quantile regression, we have encountered several issues. We will list them here.

- **Likelihood Function**: The working likelihood function in the original paper specifies a positive definite weight matrix W. To accommodate the spatial dependencies in our data, we instead used a spatially adjusted quantity inspired by the Mahalanobis Distance: $(X - \hat{\mu})^T \Sigma^{-1} (X - \hat{\mu})$, where Σ^{-1} is the inverse of the spatially varying covariance matrix.
  - However, the quantity used to replace the sample Mahalanobis Distance is not positive definite (it is only positive semi-definite). The methods seem to work for the time being, but moving forward we should clarify what role this difference plays in the function and performance of the model.
  - Furthermore, the spatially adapted Mahalanobis distance produces extremely large quantities. In the original function, the quadratic form of the score function and the weight matrix was exponentiated with a negative coefficient (that is, the kernel of the likelihood function was given by: $\exp\left(-\frac{1}{2n} s_\tau(\beta)^T W s_\tau(\beta)\right)$. However, when we replace this with the extremely large quantities generated by our spatially adapted Mahalanobis distance, the likelihood becomes effectively zero. This wreaks havoc with the rest of the code (since a likelihood of zero makes it difficult to do any calculations whatsoever) so we removed the exponentiation and the negative sign (on the coefficient) from the likelihood. This seems to yield reasonable posterior parameter estiamtes, but we will need to do some more testing to make sure that these posterior predictions are in fact accurate.
  - How can we choose a value for C in the likelihood function? I've been using 1 so far, since that will let the code run nicely for the simulation, but when we apply it to the actual data, how can we select a particular value?
  - Lastly, I think this approach makes sense, but I'd like to make sure that it isn't biased or will produce misleading results. (By using the spatial precision matrix instead of the precision matrix of the features, we are essentially scaling the feature centered observations using the)
- **Evaluation of Simulations**:
  - In the paper, the authors mostly compare the performance of the IS algorithm to other well known MCMCs across various quantiles levels. We might be able to do this, but it could be

difficult since there aren't many papers or packages that deal specifically with Bayesian spatial quantile regression. Instead, I might suggest comparing the results from our model/approach to the model/approach taken by the select few other papers that deal with this subject. (Namely, Reich et al. (2011), Ramsey (2019), and/or King & Song (2016)).
  - Also, since this is a Bayesian model, it's a little trickier to makes predictions. (Since the model parameters are themselves random variables we can't just plug in the observations to get out predictions.) We will need to derive a posterior predictive distribution to do this. (I will do my best to derive this and code it as soon as possible.)
- **"Score" Function**: The likelihood function relies on a "score" function given by: $s_\tau(\beta) = \sum_{i=1}^n x_i \psi_\tau\left(y_i - x_i^T \beta\right)$. It should be noted that this is not the traditional definition of the "score" function commonly used in inference and likelihood based statistics, but instead the "rankscore" function, which is simply a more functional way of formulating regression quantiles. (Please see slide 15 of Koenker (2003) and Guttenbrunner and Jurečková (1992)).