Methods

Marshall Honaker

First, let's import the necessary packages:

```r
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
```

```r
library(RandomFields)
```

```
## Loading required package: sp

## Loading required package: RandomFieldsUtils

##
## Attaching package: 'RandomFields'

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

```r
library(mvtnorm)
```

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.

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 $\psi_\tau(u) = \tau - I_{\{u<0\}}(u)$. Note that in this case, $\psi_\tau(.)$ is essentially the check-loss function mentioned earlier in the paper applied top the residuals.

```r
get_score <- function(y, X, tau, beta){
  u <- y - X%*%beta
  psi <- ifelse(u < 0, tau - 1, tau)
  return(t(X)%*%psi)
}
```

I'm also going to define a function to obtain the spatial covaraince matrix. I'm going to set it to use the Matérn covariance function with $\nu = 1.5$ and $\phi = 1.5$ but we can change this later or make it more adaptive as need be.

```r
get_spatial_covar_mat <- function(locs){
  dist <- rdist(locs)
  return(Matern(dist, range = .2, nu = 1.5))
}
```

Next, we will write a method to obtain a spatially adapted rendition of the proposed working likelihood function. The working likelihood 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 weight matrix. Now, to account for the spatial variability/dependence within the data, we will instead use the sample Mahalanobis distance given by $(X - \hat{\mu})^T \Sigma^{-1} (X - \hat{\mu})$ where Σ is the spatial covariance matrix. *Note that the paper requires W to be positive definite, the sample Mahalanobis matrix is only positive semi-definite. We will need to address this later. We will need to find a way to make sure that this this change will still lead to valid inference based on the posterior.*

```r
get_likelihood <- function(y, X, tau, beta, locs, C){
  score <- get_score(y, X, tau, beta)
  X_centered <- X - colMeans(X)
  coef <- -1/(2*length(y))
  kernel <- exp(coef*t(score)%*%t(X_centered)%*%get_spatial_covar_mat(locs)%*%X_centered%*%score)
  return(C*kernel[1,1])
}
```

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

```r
n <- 100
locs <- cbind(runif(n, 0, 10), runif(n, 0, 10))
m1 <- RMexp(var = 2, scale = 1.5) +
  RMnugget(var = .1) +
  RMtrend(mean = 1)
sim_vals <- RFsimulate(m1, x = locs[,1], y = locs[,2])
```

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

```r
test_X <- cbind(rnorm(n), rnorm(n))
test_beta <- c(1, 3)
test_y <- test_beta[1]*locs[,1] + test_beta[2]*locs[,2] + sim_vals$variable1
test_tau <- .5

get_score(test_y, test_X, test_tau, test_beta)
```

```
##          [,1]
## [1,] 1.551219
## [2,] 6.881443
```

```r
get_likelihood(test_y, test_X, test_tau, test_beta, locs, 1)
```

```
## [1] 9.793242e-10
```

So it seems like everything is working alright. Now let's move on to the importance sampling.

Wu & Narisetty describe an Importance Sampling (IS) procedure on pgs. 13-14. I've done my best to code it here, but there are still some issues. Most notably, the IS sampling stops working once you try to run more than 5 iterations. This will require some more testing though, I've only tried each iteration with 500 draws from the proposal. (We might be able to run more iterations/perform more updates if we use a smaller number of draws in each iteration.) Furthermore, the expression they give for the posterior covariance from the IS algorithm does not seem to produce positive definite covariance matrices. I was able to generate valid posterior covariance matrices using the simulated values from the IS algorithm, but **they do not take into account the importance weights.** As mentioned previously, we will still need to ensure that inference procedures based on the resulting posterior will be valid.

A quick overview of these methods: the `get_updated_params(.)` function simply returns a list whose elements are the parameter values for a particular set of simulated observations. This just makes the `adaIS_singleQuantile(.)` function a little more readable when we have to update the parameter values in each iteration.

```
get_updated_params <- function(y, X, tau, beta, Sigma, locs, draw){
  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] <- mean((draw[,i] - mu_hat[i])*(draw[,j] - mu_hat[j]))
    }
  }
  temp <- list(mu_hat, S)
  names(temp) <- c("mu", "S")
  return(temp)
}

adaIS_singleQuantile <- function(y, X, tau, C, locs, M, num_reps){
  p <- ncol(X)
  n <- length(y)
  beta <- coef(lm(y ~ X))[-1]
  S <- cov(X)

  initial_draw <- rmvnorm(M, beta, S)

  params <- get_updated_params(y, X, tau, beta, S, locs, initial_draw)

  for(i in 1:num_reps){
    draws <- rmvnorm(M, params$mu, params$S)
    params <- get_updated_params(y, X, tau, params$mu, params$S, locs, draws)
  }
  return(params)
}
```

Now, to test our IS method.

```r
m <- 10
S_0 <- cov(test_X)
initial_draw <- rmvnorm(m, test_beta, S_0)

get_updated_params(test_y, test_X, test_tau, test_beta, cov(test_X), locs, initial_draw)
```

```
## $mu
## [1] 0.9199095 5.9580444
##
## $S
##            [,1]       [,2]
## [1,]  0.7924300 -0.5833095
## [2,] -0.5833095  8.8720927
```

```r
posterior_parameters <- adaIS_singleQuantile(test_y, test_X, test_tau, 1, locs, m, 5)
```

So everything seems to be working, thought it does need a little tuning/refinement. According to the above example, the mean of the posterior distribution of $\vec{\beta}$ will have mean -12.9653031, -1.5322844 and covariance matrix 41.4735346, 8.4375204, 8.4375204, 1.7382331.