Augmented Feature Matrix Approach

Marshall Honaker

12/3/2020

First, let's import the necessary packages:

```
library(fields)
library(RandomFields)
library(mvtnorm)
library(ggplot2)
library(matrixcalc)
library(gstat)
library(GpGp)
library(doParallel)
```

Modified function to return augmented feature matrix as discussed with Dr. Sang.

```
augmented_feature_mat <- function(X, locs, knots, phi, sigma_sq){
  C_knots <- sigma_sq*exp(-rdist(knots, knots)/phi)
  half_inv_C_knots <- chol(solve(C_knots))
  basis_X <- sigma_sq*exp(-rdist(locs, knots)/phi)%*%half_inv_C_knots
  return(cbind(X, basis_X)[,-1])
}
```

Now, let's move on to the more rudimentary functions

```
get_score<- function(y, X, tau, beta){  ## See item (2) below
  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)
}

weight <- function(X, tau){  ## See item (4) below
  n <- nrow(X)
  temp <- X[1,]%*%t(X[1,])
  for(i in 2:n){
    temp <- temp + X[i,]%*%t(X[i,])
  }
  coef <- n/(tau*(1 - tau))
  return(coef*solve(temp))
}

get_likelihood <- function(y, X, tau, beta, locs, C){ ## See item (5) below
  score <- get_score(y, X, tau, beta)
  coef <- -1/(2*length(y))
```

```
  weight_X <- weight(X, tau)
  kernel <- exp(coef*(t(score)%*%weight_X%*%score)[1,1])
  return(C*kernel)
}
```

A few comments on the methods above:

1. The `augmented_feature_mat()` method was returning a feature matrix with the column of 1s for the intercept. `R` will include this automatically, so I modified the `return()` statement so the output would not include this column.
2. The `get_score()` function outputs a p length vector whose entries correspond to the features in the feature matrix. The last element in this vector (corresponding to the additional columns we get by appending `X_basis` to out original feature matrix) is much larger than the others, so we should keep an eye on this.
3. I deleted the `get_spatial_covar_mat()` function because we no longer need it.
4. The paper specifies that the weight matrix W must be positive definite. Even though we followed the definition for W given in the paper to a tee, our function doesn't return a positive definite matrix if you consider the eighth decimal place of each entry. Since the eighth decimal isn't especially significant in the grand scheme of things, I added the `round(., 7)` method to make sure that the method returns a weight matrix that satisfies the positive definiteness requirement.
5. I'm not sure how to come up with good values of C in the likelihood function. I've been using $C = 1$ so far just so the code will run and play nice. I need to read through the paper some more to figure out how they did it.

Next, let's look at the methods for the Importance Sampling algorithm for a single quantile.

```
get_updated_params <- function(y, X, tau, beta, Sigma, locs, draws){
  n <- length(y)
  p <- ncol(X)

  is_weights <- numeric(nrow(draws))
  for(i in 1:nrow(draws)){
    is_weights[i] <- get_likelihood(y, X, tau, draws[i,], locs, 1)*(1/(2*n)^p)/dmvnorm(draws[i,], beta,
  }

  mu_hat <- apply(w*draws, 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*draws[,i]*draws[,j])/sum(w) - mu_hat[i]*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) ## 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

  a_1 <- coef(lm(y ~ X))[-1]
```

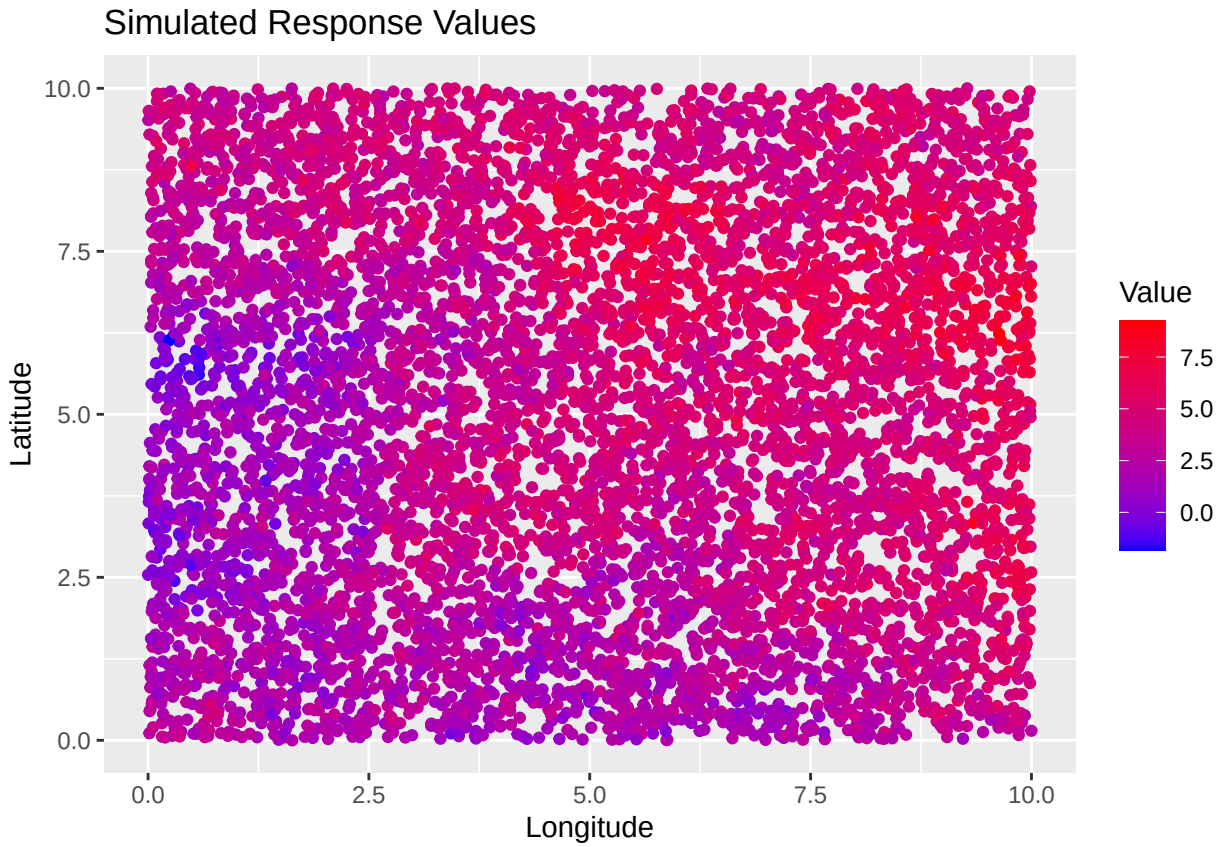
```
  a_0 <- quantile(y - X%*%a_1, tau)
  a <- c(a_0, a_1)
  X <- cbind(rep(1, n), X)
  temp <- X[1,]%*%t(X[1,])
  for(i in 2:n){
    temp <- temp + X[i,]%*%t(X[i,])
  }
  S_0 <- 1*tau*(1 - tau)*solve((1/n)*temp)

  params <- list(a, S_0)
  names(params) <- c("mu", "S")
  draw <- rmvnorm(M, params$mu, params$S)

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

  for(i in 2:num_reps){ ## Step 4: Repeat steps 2 and 3 until we achieve the desired effective sample s
    draw <- rmvnorm(M, params$mu, params$S)  ## Step 2: Simulate M values from the proposal distributio
    params <- get_updated_params(y, X, tau, params$mu, params$S, locs, draw) ## Step 3: Update the para
  }
  return(params)
}
```

Next, let's simulate some data so we can test what we have so far.

First, let's generate and plot the simulated data.

```
n <- 10000
locs <- as.matrix(cbind(runif(n, 0, 10), runif(n, 0, 10)))
sigma_sq <- 2
phi <- 5
nu <- .5
tau_sq <- .5

w <- fast_Gp_sim(c(sigma_sq, phi, nu, 0), "matern_isotropic", locs, 30)

X <- as.matrix(cbind(rep(1, n), locs[,1]/10, locs[,2]/10))
beta <- as.matrix(c(1, 2, 3))
p <- length(beta)
y <- rnorm(n, X%*%beta + w, sqrt(tau_sq))

training_indices <- sample(1:n, round(.75*n), replace = FALSE)

y_train <- y[training_indices]
X_train <- X[training_indices,]
locs_train <- locs[training_indices,]

y_test <- y[-training_indices]
X_test <- X[-training_indices,]
locs_test <- locs[-training_indices,]

training_data <- as.data.frame(cbind(y_train, X_train[,-1], locs_train))
names(training_data) <- c("y_train", "X1_train", "X2_train", "long_train", "lat_train")

ggplot() + geom_point(data = training_data, aes(x = long_train, y = lat_train, col = y_train)) +
```

```
  scale_colour_gradient(low = "blue", high = "red") +
  labs(title = "Simulated Response Values",
       x = "Longitude",
       y = "Latitude",
       col = "Value")
```



Next, let's test out our more basic methods.

```
tau <- .5

X_augmented <- augmented_feature_mat(X_train, locs_train, expand.grid(seq(0.1,9.9,length=5),seq(0.1,9.9
X_augmented[1:10,]
```

```
##             [,1]       [,2]      [,3]      [,4]      [,5]      [,6]       [,7]
##  [1,] 0.90144636 0.38017481 0.2907133 0.3414128 0.5046459 0.6436942 0.49466345
##  [2,] 0.34046266 0.09971688 1.0102839 1.1807000 0.7855979 0.3292099 0.13831739
##  [3,] 0.86672374 0.31800177 0.3244146 0.3863848 0.5801539 0.7436754 0.52769288
##  [4,] 0.21349409 0.91212062 0.3152171 0.2059022 0.1755485 0.1293384 0.06923943
##  [5,] 0.03825113 0.39167863 0.9320539 0.4650838 0.2820765 0.1642182 0.07081242
##  [6,] 0.34693320 0.09757940 0.9988226 1.1756822 0.8086337 0.3337296 0.14030687
##  [7,] 0.99866654 0.63957727 0.1922174 0.2122398 0.2955678 0.3636075 0.31244149
##  [8,] 0.42445405 0.56524650 0.5012441 0.4338743 0.4140102 0.3022069 0.15051588
##  [9,] 0.34310562 0.24936996 0.8822297 0.8646122 0.6680751 0.3435577 0.14219668
## [10,] 0.08506097 0.89480599 0.3392943 0.1973135 0.1566782 0.1095377 0.05608016
##             [,8]       [,9]       [,10]        [,11]        [,12]        [,13]
##  [1,] 0.05126766 0.07047130 0.221054320  0.539005951  0.363663622  0.026117028
##  [2,] 0.11752958 0.18500384 0.057725118 -0.064139970 -0.095253754 -0.002561044
##  [3,] 0.04452468 0.06251096 0.227880898  0.623624910  0.285259825  0.015634216
```

```
##  [4,] 0.28272789 0.15036538 0.111688303  0.064660673  0.004849260  0.457451941
##  [5,] 0.90985730 0.16030779 0.001838208 -0.007755388 -0.040815455  0.706769191
##  [6,] 0.10992524 0.17529704 0.062934258 -0.064638159 -0.096449729 -0.002985362
##  [7,] 0.06228550 0.07559060 0.166390853  0.301453379  0.258557029  0.057608245
##  [8,] 0.32014683 0.33934728 0.330147442  0.140291172 -0.016248229  0.317291242
##  [9,] 0.32667869 0.61351566 0.244775336 -0.043909565 -0.086123790  0.048067642
## [10,] 0.32048372 0.13108321 0.083441688  0.043918267 -0.002311292  0.540302911
##             [,14]        [,15]       [,16]        [,17]        [,18]
##  [1,]  0.04315400  0.127051225 0.381286654  0.156828209 -0.002995504
##  [2,] -0.02232433 -0.009797530 0.008223273 -0.051801849 -0.001710911
##  [3,]  0.02445579  0.080467665 0.255534948 -0.017382638 -0.004087791
##  [4,]  0.29609064  0.178406828 0.083446631 -0.003361837  0.640640797
##  [5,]  0.06875451 -0.009448854 0.006422083 -0.033892343  0.062592405
##  [6,] -0.02196064 -0.010070285 0.008235665 -0.052313660 -0.001931041
##  [7,]  0.08888984  0.205148873 0.464129095  0.518605938  0.021861733
##  [8,]  0.54453330  0.624997936 0.058779166 -0.057689149  0.095161030
##  [9,]  0.05746052  0.039343251 0.005665803 -0.058603749 -0.001417858
## [10,]  0.23802662  0.115582252 0.053325121 -0.008790862  0.862208418
##              [,19]        [,20]        [,21]       [,22]        [,23]
##  [1,] -0.004694600 -0.006329020 0.0358221999 -0.15009105 0.0154114528
##  [2,] -0.002565537 -0.003779586 0.0077263281 -0.04168996 0.0008642146
##  [3,] -0.004442577 -0.006735395 0.0292175523 -0.12772399 0.0136082040
##  [4,]  0.533900564  0.143326624 0.0348537398 -0.03833879 0.4097004612
##  [5,] -0.023330185 -0.009637151 0.0009372025 -0.03465187 0.0050569469
##  [6,] -0.002522743 -0.003744866 0.0078902211 -0.04194529 0.0009648892
##  [7,]  0.034901210  0.095665585 0.2972285179  0.32726684 0.0211992440
##  [8,]  0.170119230  0.176669211 0.0011442769 -0.08207209 0.0068209530
##  [9,] -0.009889336 -0.009067305 0.0056172413 -0.05090298 0.0021703751
## [10,]  0.308069488  0.050256116 0.0169907153 -0.03371836 0.5887477002
##             [,24]       [,25]       [,26]       [,27]
##  [1,] -0.01225382 -0.02848270 -0.02423855 -0.22048491
##  [2,] -0.03291880 -0.03886125 -0.03008973 -0.08016138
##  [3,] -0.01305396 -0.02807303 -0.02339336 -0.19102709
##  [4,]  0.24205697 -0.28072477 -0.17065615 -0.18083448
##  [5,] -0.09126103 -0.07501490 -0.04898021 -0.07899751
##  [6,] -0.03246958 -0.03854836 -0.02991050 -0.08047853
##  [7,] -0.01328062 -0.03664738 -0.04749605 -0.39203985
##  [8,] -0.09612295 -0.12692491 -0.10766943 -0.19137831
##  [9,] -0.04617170 -0.05506727 -0.04375936 -0.10325600
## [10,] -0.12297757 -0.23777947 -0.12290926 -0.13513375
```

```
beta <- coef(lm(y_train ~ X_augmented))[-1]
score <- get_score(y_train, X_augmented, tau, beta)
score
```

```
##             [,1]
##  [1,] 1864.00762
##  [2,] 1866.81891
##  [3,] 1991.73937
##  [4,] 1716.82390
##  [5,] 1712.36437
##  [6,] 1449.54222
##  [7,]  799.60466
##  [8,]  916.29598
##  [9,]  684.70563
```

```
## [10,]  710.53469
## [11,]  625.14678
## [12,]   60.32692
## [13,]  812.77319
## [14,]  711.33975
## [15,]  729.26241
## [16,]  664.59325
## [17,]   15.08519
## [18,]  551.54475
## [19,]  478.10161
## [20,]  460.27955
## [21,]  446.08022
## [22,] -116.73378
## [23,]  219.94426
## [24,] -116.38366
## [25,] -208.47069
## [26,] -193.72344
## [27,] -725.31154
```

```
weight_mat <- weight(X_augmented, tau)
weight_mat
```

```
##             [,1]       [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,]  89491.987 -18960.555  4219.41706 -10926.1884 -18070.3015 -28048.8122
##  [2,] -18960.555  94432.500 -3119.73446  -2893.1454  -1403.1551   1133.0285
##  [3,]   4219.417  -3119.734   538.21075   -723.3845   -628.6716  -1228.3546
##  [4,] -10926.188  -2893.145  -723.38453   2255.0395   2156.6646   3825.2282
##  [5,] -18070.302  -1403.155  -628.67162   2156.6646   4457.9046   5606.8067
##  [6,] -28048.812   1133.028 -1228.35463   3825.2282   5606.8067   9616.2240
##  [7,] -47017.687   5775.315 -2101.82384   5928.7461   9856.5952  14491.7145
##  [8,]   6191.332 -21135.926   517.68202    611.2778   -123.8489   -872.0660
##  [9,]  -5091.207 -16697.253   338.28405   1398.3462   2152.7434   2554.6701
## [10,]  -8179.046 -16100.860    58.82732   2119.1380   2475.2399   3630.7237
## [11,] -13330.290 -15116.200  -182.83143   2671.8633   3826.2880   4942.3561
## [12,] -29793.457 -11928.941  -946.32664   4706.8178   7089.6832  10554.5612
## [13,]   5782.688 -26669.358   987.79853    712.7264    296.4243   -454.4369
## [14,]  -6931.080 -18707.972   147.85911   2072.2584   2559.2761   3233.0029
## [15,] -11197.410 -18462.570   -21.82514   2557.2552   3540.5230   4604.5580
## [16,] -16739.259 -17360.531  -269.71812   3241.2723   4573.1775   6410.8450
## [17,] -30841.299 -17568.784  -854.00707   5169.4546   7620.7501  10946.8640
## [18,]   6482.060 -35715.077  1125.03522   1208.4183    690.0899   -198.0729
## [19,]  -7657.090 -23749.193   271.72125   2380.1701   3020.4776   3760.8159
## [20,] -11486.123 -24332.169   126.85107   2952.9926   3856.7474   5043.5590
## [21,] -16476.712 -22482.284  -129.49503   3522.3032   4846.5347   6545.1560
## [22,] -29762.988 -24276.663  -651.78205   5395.0989   7785.9602  10962.9354
## [23,]   9114.613 -53489.723  1711.60188   1857.2375   1143.0431   -111.6175
## [24,]  -8868.276 -33146.836   454.47792   3115.7766   3827.5004   4645.6517
## [25,] -13519.118 -34018.038   281.47804   3792.7448   4894.3269   6220.8699
## [26,] -19480.818 -31732.977   -30.88811   4445.6598   6008.7790   8021.1186
## [27,] -29640.010 -31927.834  -411.40393   5881.5738   8270.2705  11418.9761
##             [,7]        [,8]        [,9]        [,10]       [,11]       [,12]
##  [1,] -47017.687   6191.3323  -5091.2070  -8179.04559 -13330.2901 -29793.4566
##  [2,]   5775.315 -21135.9260 -16697.2532 -16100.86025 -15116.1998 -11928.9414
##  [3,]  -2101.824    517.6820    338.2841     58.82732   -182.8314   -946.3266
##  [4,]   5928.746    611.2778   1398.3462   2119.13803   2671.8633   4706.8178
```

```
##  [5,]   9856.595   -123.8489   2152.7434   2475.23990   3826.2880   7089.6832
##  [6,]  14491.714   -872.0660   2554.6701   3630.72375   4942.3561  10554.5612
##  [7,]  25627.487  -2331.6771   3490.7983   5129.58546   7878.6400  16069.6389
##  [8,]  -2331.677   5291.9724   3292.2456   3398.01801   2992.5864   1947.9599
##  [9,]   3490.798   3292.2456   4279.9879   3793.22962   4341.9390   5263.9142
## [10,]   5129.585   3398.0180   3793.2296   4725.57979   4571.7970   6408.5840
## [11,]   7878.640   2992.5864   4341.9390   4571.79703   6001.4460   7870.2080
## [12,]  16069.639   1947.9599   5263.9142   6408.58399   7870.2080  14501.1268
## [13,]  -1857.305   5670.9631   4792.0074   4476.40157   4196.4625   3209.7129
## [14,]   4584.172   4036.6714   4161.5324   4648.35394   5048.1534   6365.8865
## [15,]   6857.503   3795.3515   4775.4714   4952.00831   5855.4833   7933.4133
## [16,]   9764.380   3434.9915   5075.0319   5697.14319   6476.4248   9758.4327
## [17,]  17460.727   3166.1926   6514.0846   7596.56948   9477.0865  14791.8802
## [18,]  -1808.587   8094.3070   6369.1678   6183.63562   5858.1532   4775.0311
## [19,]   5203.878   5029.7109   5484.8520   5713.38349   6173.5168   7674.7709
## [20,]   7294.311   5071.3455   5927.9868   6408.10232   7036.2955   9240.4427
## [21,]   9873.032   4567.9525   6053.9564   6647.95751   7685.4341  10762.5201
## [22,]  17019.355   4640.1312   7705.7294   8744.24935  10511.4661  16182.6233
## [23,]  -2396.466  11892.2906   9618.8613   9337.35040   8876.3093   7369.3813
## [24,]   6287.678   7085.4295   7377.8296   7719.80265   8278.5345  10017.2192
## [25,]   8834.872   7178.6968   8044.9066   8509.69735   9358.7738  11966.0983
## [26,]  11899.039   6524.0051   8171.4915   8862.40212   9987.7593  13683.7038
## [27,]  17459.325   6392.0395   9290.6634  10375.59462  12140.9639  17682.7876
##             [,13]       [,14]        [,15]       [,16]       [,17]        [,18]
##  [1,]   5782.6881  -6931.0796 -11197.41036 -16739.2588 -30841.2989   6482.0595
##  [2,] -26669.3581 -18707.9720 -18462.57013 -17360.5306 -17568.7836 -35715.0773
##  [3,]    987.7985    147.8591    -21.82514   -269.7181   -854.0071   1125.0352
##  [4,]    712.7264   2072.2584   2557.25523   3241.2723   5169.4546   1208.4183
##  [5,]    296.4243   2559.2761   3540.52302   4573.1775   7620.7501    690.0899
##  [6,]   -454.4369   3233.0029   4604.55797   6410.8450  10946.8640   -198.0729
##  [7,]  -1857.3050   4584.1718   6857.50278   9764.3796  17460.7268  -1808.5866
##  [8,]   5670.9631   4036.6714   3795.35149   3434.9915   3166.1926   8094.3070
##  [9,]   4792.0074   4161.5324   4775.47144   5075.0319   6514.0846   6369.1678
## [10,]   4476.4016   4648.3539   4952.00831   5697.1432   7596.5695   6183.6356
## [11,]   4196.4625   5048.1534   5855.48327   6476.4248   9477.0865   5858.1532
## [12,]   3209.7129   6365.8865   7933.41333   9758.4327  14791.8802   4775.0311
## [13,]   8033.9767   4987.3289   5216.42284   4790.1813   4817.9219   9795.9611
## [14,]   4987.3289   5536.9497   5351.43962   6038.8674   7771.6332   7301.6829
## [15,]   5216.4228   5351.4396   6631.57351   6769.6274   9532.7411   7103.2398
## [16,]   4790.1813   6038.8674   6769.62740   8391.5180  11201.0456   6751.5240
## [17,]   4817.9219   7771.6332   9532.74108  11201.0456  17914.0810   6943.8086
## [18,]   9795.9611   7301.6829   7103.23976   6751.5240   6943.8086  13995.5660
## [19,]   6717.6900   6073.9099   6858.06631   7323.0744   9422.1216   8850.1952
## [20,]   6788.5623   6901.1552   7447.92873   8413.8372  11144.4325   9399.9476
## [21,]   6255.9780   7098.2051   8106.80905   8977.3079  12647.5994   8673.1861
## [22,]   6674.7457   9119.6392  10749.12385  12709.2525  18074.6643   9456.8071
## [23,]  15224.0128  10796.0891  10755.29337  10234.1900  10630.8636  19905.2266
## [24,]   9313.3285   8610.5502   9217.74539   9773.4811  12434.1714  12854.2060
## [25,]   9530.7245   9326.9461  10334.69561  11121.7241  14590.1007  13068.7241
## [26,]   8851.4484   9525.4206  10714.11332  12021.9907  16338.5256  12231.5502
## [27,]   8921.1748  10960.8559  12694.58781  14629.8412  20863.8005  12477.2837
##             [,19]       [,20]      [,21]       [,22]       [,23]        [,24]
##  [1,]  -7657.0903 -11486.1226 -16476.712 -29762.988   9114.6130  -8868.2760
##  [2,] -23749.1932 -24332.1693 -22482.284 -24276.663 -53489.7231 -33146.8356
```

```
##  [3,]    271.7213    126.8511   -129.495   -651.782   1711.6019    454.4779
##  [4,]   2380.1701   2952.9926   3522.303   5395.099   1857.2375   3115.7766
##  [5,]   3020.4776   3856.7474   4846.535   7785.960   1143.0431   3827.5004
##  [6,]   3760.8159   5043.5590   6545.156  10962.935   -111.6175   4645.6517
##  [7,]   5203.8779   7294.3110   9873.032  17019.355  -2396.4655   6287.6784
##  [8,]   5029.7109   5071.3455   4567.952   4640.131  11892.2906   7085.4295
##  [9,]   5484.8520   5927.9868   6053.956   7705.729   9618.8613   7377.8296
## [10,]   5713.3835   6408.1023   6647.958   8744.249   9337.3504   7719.8026
## [11,]   6173.5168   7036.2955   7685.434  10511.466   8876.3093   8278.5345
## [12,]   7674.7709   9240.4427  10762.520  16182.623   7369.3813  10017.2192
## [13,]   6717.6900   6788.5623   6255.978   6674.746  15224.0128   9313.3285
## [14,]   6073.9099   6901.1552   7098.205   9119.639  10796.0891   8610.5502
## [15,]   6858.0663   7447.9287   8106.809  10749.124  10755.2934   9217.7454
## [16,]   7323.0744   8413.8372   8977.308  12709.253  10234.1900   9773.4811
## [17,]   9422.1216  11144.4325  12647.599  18074.664  10630.8636  12434.1714
## [18,]   8850.1952   9399.9476   8673.186   9456.807  19905.2266  12854.2060
## [19,]   8249.9864   8311.1388   8787.277  11086.516  13793.1605  10349.4053
## [20,]   8311.1388   9846.2752   9662.923  12935.768  14058.0653  11549.3922
## [21,]   8787.2771   9662.9226  11006.774  14062.926  13159.1951  11716.6796
## [22,]  11086.5158  12935.7681  14062.926  20893.753  14388.1334  14739.3346
## [23,]  13793.1605  14058.0653  13159.195  14388.133  30939.6351  18824.9236
## [24,]  10349.4053  11549.3922  11716.680  14739.335  18824.9236  15316.2219
## [25,]  11649.2733  12454.2858  13221.066  16946.351  19824.5196  15423.1490
## [26,]  11763.9581  13231.3224  13701.696  18441.054  18459.8089  15966.0700
## [27,]  13426.8141  15368.7143  16984.016  22617.409  18912.2804  17892.4493
##             [,25]        [,26]       [,27]
##  [1,] -13519.118 -19480.81840 -29640.0105
##  [2,] -34018.038 -31732.97724 -31927.8340
##  [3,]    281.478    -30.88811   -411.4039
##  [4,]   3792.745   4445.65982   5881.5738
##  [5,]   4894.327   6008.77903   8270.2705
##  [6,]   6220.870   8021.11857  11418.9761
##  [7,]   8834.872  11899.03938  17459.3248
##  [8,]   7178.697   6524.00506   6392.0395
##  [9,]   8044.907   8171.49152   9290.6634
## [10,]   8509.697   8862.40212  10375.5946
## [11,]   9358.774   9987.75929  12140.9639
## [12,]  11966.098  13683.70380  17682.7876
## [13,]   9530.725   8851.44841   8921.1748
## [14,]   9326.946   9525.42062  10960.8559
## [15,]  10334.696  10714.11332  12694.5878
## [16,]  11121.724  12021.99073  14629.8412
## [17,]  14590.101  16338.52564  20863.8005
## [18,]  13068.724  12231.55020  12477.2837
## [19,]  11649.273  11763.95808  13426.8141
## [20,]  12454.286  13231.32241  15368.7143
## [21,]  13221.066  13701.69630  16984.0162
## [22,]  16946.351  18441.05375  22617.4089
## [23,]  19824.520  18459.80889  18912.2804
## [24,]  15423.149  15966.06997  17892.4493
## [25,]  18025.800  17351.22775  20595.9999
## [26,]  17351.228  19224.02880  21630.1487
## [27,]  20596.000  21630.14873  28393.8946
```

```
lik <- get_likelihood(y_train, X_augmented, tau, beta, locs_train, 1)
lik
```

```
## [1] 0
```

As can be seen above, the code for the more basic methods runs. However, we still run into the same issues we discussed earlier.

Now, let's move on to the Importance Sampling methods.

```
Sigma <- cov(X_augmented)
draw <- rmvnorm(1000, beta, Sigma)
## pars <- get_updated_params(y_train, X_augmented, tau, beta, Sigma, locs_train, draw)
## pars

## post_dist_params <- adaIS_singleQuantile(y_train, X_augmented, tau, 1, locs_train, 5000, 10)
## get_updated_params() isn't working, so we can't expect adaIS_singleQuantile() to work either
```

Because this is a Bayesian regression model, we will need to derive the posterior predictive distribution to make predictions at the test locations using the test data. Once the Importance Sampling methods work and give us proper values for the parameters of the posterior distribution, we can make the desired predictions to evaluate the predictive performance of our model.