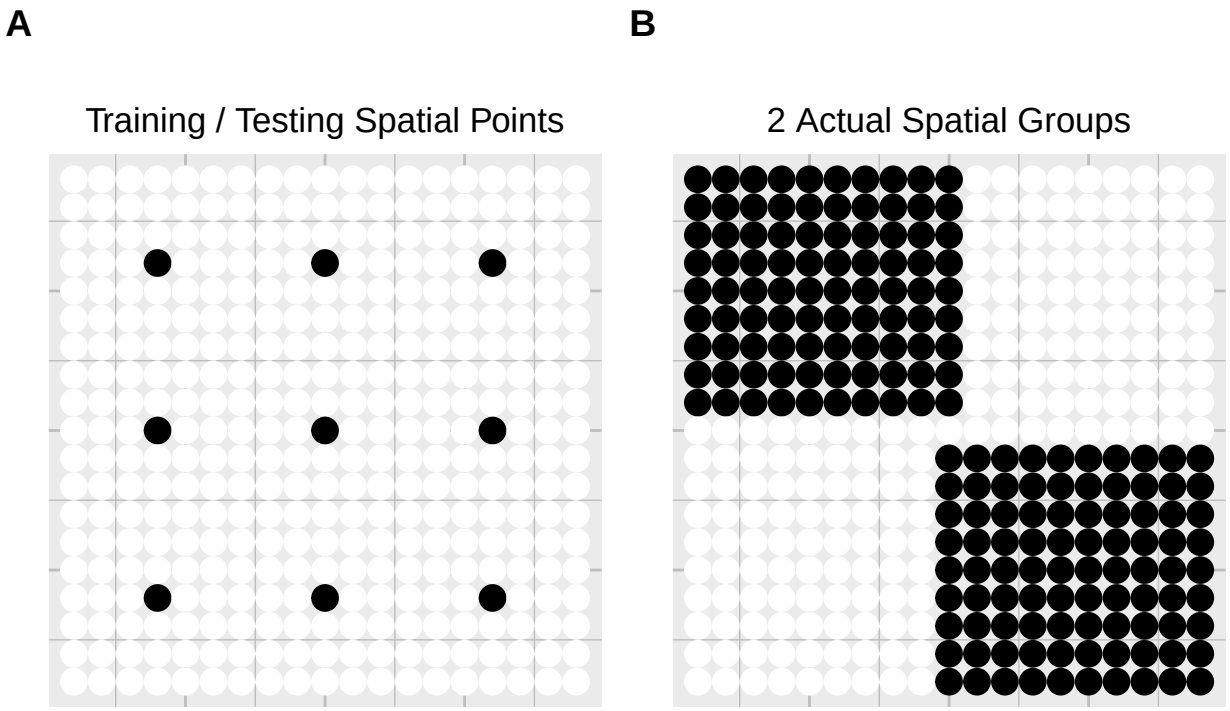
`spatempBFA` Toy Example Illustration

Yifan CHENG y.cheng@u.nus.edu

Data Generation and Model Fitting

```
rm(list=ls())
library(mvtnorm)
library(fields)
library(tidyverse)
library(ggpubr)
numSpatOverallGroups <- 2
M <- 361
m = 352 # 9 testing spatial points
K <- 2
O <- 1
L <- min(10, M)
LjVec <- rep(min(10, M), K)
sqrootM <- 19
Nu <- 310
trainingT <- 300 # training set T = 300
testingT <- 10
Time <- 1:trainingT
TimeDist <- as.matrix(dist(1:Nu))
APsi = 0.1; BPsi = 4.5
set.seed(29)
calcGroup <- function(row, col, sqrootM){
  return((row - 1)*sqrootM + col)
}
sigma2 <- 0.01 # actual sigma^2(i,o) (for i=1,2,...,M and o=1) values
psi <- 2.3
kappa <- 0.7
tempMat <- matrix(runif(K*K,0,1), K, K)
Upsilon <- t(tempMat)%*%tempMat
rho <- 0.8
D <- rdist(expand.grid(1:sqrootM, 1:sqrootM))
Frho <- exp(-rho*D)
Hpsi <- exp(-psi*TimeDist)
Eta <- rmvnorm(1, mean=rep(0, Nu*K), sigma=kronecker(Hpsi, Upsilon))
# actual Eta (c(Eta_1,...,Eta_T)) (vec of length Nu*K)
Lambda <- matrix(5, M * O, K)
theta2j <- c(10, -10)
rowColInd <- rbind(expand.grid(1:10, 1:9), expand.grid(10:19, 11:19))
whichGroup1 <- mapply(calcGroup, rowColInd[,2], rowColInd[,1], sqrootM = sqrootM)
spatGroupOverall <- rep(2,M); spatGroupOverall[whichGroup1] <- 1
# matrix(spatGroupOverall, 19, 19)
Lambda[,2] = theta2j[spatGroupOverall]
```

```
Sigma.NuMO <- matrix(rnorm(Nu * M * O, sd = sqrt(sigma2)), M*O, Nu)
EtaMat <- matrix(Eta, K, Nu)
meanMat <- Lambda%*%EtaMat #M*O\times Nu
YtrainingTemp <- as.vector(meanMat[,1:trainingT] + Sigma.NuMO[,1:trainingT])
YtestingTemp <- as.vector(meanMat[,(trainingT+1):Nu] + Sigma.NuMO[,(trainingT+1):Nu])
testingRowCol <- expand.grid(c(4, 10, 16), c(4, 10, 16))
whichTesting <- mapply(calcGroup, testingRowCol[,2], testingRowCol[,1], sqrootM = sqrootM)
Dtraining <- as.matrix(D[-whichTesting, -whichTesting])
discardInd <- vector()
for(whichTestingLoc in whichTesting){
  discardInd <- c(discardInd, seq(from = whichTestingLoc, by = M, length.out = trainingT))
}
Ytraining <- YtrainingTemp[-discardInd]
YtestingSpat <- YtrainingTemp[discardInd]
rm(YtrainingTemp)
spatGroupOverallTraining <- spatGroupOverall[-whichTesting]
distOrigNew = as.matrix(D[-whichTesting, whichTesting])
distNewNew = as.matrix(D[whichTesting, whichTesting])
dat = data.frame(Y = Ytraining); dist = Dtraining
tempTestingDiscardInd <- vector()
for(whichTestingLoc in whichTesting){
  tempTestingDiscardInd <- c(tempTestingDiscardInd,
                             seq(from = whichTestingLoc, by = M, length.out = testingT))
}
YtestingTemp <- YtestingTemp[-tempTestingDiscardInd]
xcoord <- rep(1:sqrootM, sqrootM)
ycoord <- rep(seq(sqrootM, 1, by = -1), each = sqrootM)
trainingTestingSpatGp <- rep(1, M); trainingTestingSpatGp[whichTesting] = 2
spatGpDF <- data.frame(x = xcoord, y = ycoord, actualGp <- as.factor(spatGroupOverall),
                       trainingTestingSpatGp <- as.factor(trainingTestingSpatGp))
trainingTestingPlotBW <- ggplot(spatGpDF) + ggtitle("Training / Testing Spatial Points") +
  coord_fixed(ratio = 1) +
  geom_point(aes(x = x, y = y, col = trainingTestingSpatGp),
             size = 4.2, show.legend = FALSE) +
  scale_color_manual(values = c("white", "black")) + labs(x = "", y = "") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        panel.grid.major = element_line(color="grey"),
        panel.grid.minor = element_line(color="grey"))
actualSpatPlotBW <- ggplot(spatGpDF) + ggtitle("2 Actual Spatial Groups") +
  coord_fixed(ratio = 1) +
  geom_point(aes(x = x, y = y, col = actualGp), size = 4.2, show.legend = FALSE) +
  scale_color_manual(values = c("black", "white")) + labs(x = "", y = "") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        panel.grid.major = element_line(color="grey"),
        panel.grid.minor = element_line(color="grey"))
ggarrange(trainingTestingPlotBW, actualSpatPlotBW, align = "h", ncol = 2, nrow = 1,
          labels = c("A", "B"))
```



The entire simulated data set consists of $y_t(\boldsymbol{s}_i)$'s from 310 time points and 361 spatial locations, which belong to 2 actual spatial groups (**B**). 352 out of the 361 locations (**A**) are training ones, and the 9 remaining black locations (**A**) are testing ones. The training data set used to fit all 6 methods (our 4 methods `fullGPfixedL`, `NNGPblockFixedL`, `NNGPsequenFixedL`, and `NNGPsequenVaryLj` & `spBFAL10`, `spBFALInf` from the R package `spBFA`) are data corresponding to the 352 white locations (**A**) and the first 300 time points. $y_t(\boldsymbol{s}_i)$'s corresponding to the 9 testing locations for the first 300 time points are used to assess our 4 methods' spatial prediction performances. Similarly, $y_t(\boldsymbol{s}_i)$'s corresponding to the 352 training locations for the last 10 time points are used to assess all 6 methods' temporal prediction performances.

```
library(spatempBFA)
library(coda)
MCMC <- list(NBurn = 80000, NSims = 20000, NThin = 4, NPilot = 5)
regFixedL.simu <- bfaFixedL(Y ~ 0, data = dat, dist = Dtraining, time = Time,  K = K,
                            starting = NULL, hypers = NULL, tuning = NULL,
                            mcmc = MCMC,
                            L = L,
                            family = "normal",
                            temporal.structure = "exponential",
                            spatial.structure = "continuous",
                            seed = 29,
                            gamma.shrinkage = TRUE,
                            include.time = TRUE,
                            include.space = TRUE,
                            clustering = TRUE,
                            seasonPeriod = 1,
                            equalTimeDist = TRUE,
                            spatApprox = FALSE,
```

```
                            alphaMethod = "block",
                            h = 15,
                            storeSpatPredPara = TRUE,
                            storeWeights = TRUE,
                            alphasWeightsToFiles = FALSE)
save(regFixedL.simu, file="toyExfullGPfixedL.RData")
GibbsStepTimeFixedLfullGP <- regFixedL.simu$GibbsStepTime
save(GibbsStepTimeFixedLfullGP, file = "GibbsStepTimeFixedLfullGP.RData")
Diags <- diagnostics(regFixedL.simu, diags = c("dic", "dinf", "meanIC", "waic"),
                     keepDeviance = TRUE)
save(Diags, file = "toyExfullGPfixedLDiags.RData")
Deviance <- as.mcmc(Diags$deviance)
save(Deviance, file = "toyExfullGPfixedLDeviance.RData")
spatpredFixedL <- predictNewLocFixedL(regFixedL.simu, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 29)
save(spatpredFixedL, file = "toyExFullGPspatpredFixedL.RData")
tempppredFixedL <- predictNewTime(regFixedL.simu, (trainingT+1):Nu, seed = 29)
save(tempppredFixedL, file = "toyExFullGPtempppredFixedL.RData")
fittedClusGpMat <- matrix(0, m, 3)
clusFixedL10 <- clusteringFixedL(regFixedL.simu, o = 1, nkeep = 10,
                                 nCent = numSpatOverallGroups)
fittedClusGpMat[,1] <- clusFixedL10$cluster
clusFixedL100 <- clusteringFixedL(regFixedL.simu, o = 1, nkeep = 100,
                                  nCent = numSpatOverallGroups)
fittedClusGpMat[,2] <- clusFixedL100$cluster
clusFixedL1000 <- clusteringFixedL(regFixedL.simu, o = 1, nkeep = 1000,
                                   nCent = numSpatOverallGroups)
fittedClusGpMat[,3] <- clusFixedL1000$cluster
weightsNumIter <- c(10, 100, 1000)
colnames(fittedClusGpMat) <- paste(weightsNumIter, "iterWeights")
save(fittedClusGpMat, file = "fullGPfixedLtoyExfittedClusGpMat.RData")
tempppredFixedL <- predictNewTime(regFixedL.simu, (trainingT+1):Nu, seed = 27)
save(tempppredFixedL, file = "s27toyExFullGPtempppredFixedL.RData")
tempppredFixedL <- predictNewTime(regFixedL.simu, (trainingT+1):Nu, seed = 19)
save(tempppredFixedL, file = "s19toyExFullGPtempppredFixedL.RData")
tempppredFixedL <- predictNewTime(regFixedL.simu, (trainingT+1):Nu, seed = 31)
save(tempppredFixedL, file = "s31toyExFullGPtempppredFixedL.RData")
spatpredFixedL <- predictNewLocFixedL(regFixedL.simu, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 27)
save(spatpredFixedL, file = "s27toyExFullGPspatpredFixedL.RData")
spatpredFixedL <- predictNewLocFixedL(regFixedL.simu, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 31)
save(spatpredFixedL, file = "s31toyExFullGPspatpredFixedL.RData")
spatpredFixedL <- predictNewLocFixedL(regFixedL.simu, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 19)
save(spatpredFixedL, file = "s19toyExFullGPspatpredFixedL.RData")
regFixedL.simu.block <- bfaFixedL(Y ~ 0, data = dat, dist = Dtraining, time = Time,
                                  K = K,
                                  starting = NULL, hypers = NULL, tuning = NULL,
```

```
                                    mcmc = MCMC,
                                    L = L,
                                    family = "normal",
                                    temporal.structure = "exponential",
                                    spatial.structure = "continuous",
                                    seed = 29,
                                    gamma.shrinkage = TRUE,
                                    include.time = TRUE,
                                    include.space = TRUE,
                                    clustering = TRUE,
                                    seasonPeriod = 1,
                                    equalTimeDist = TRUE,
                                    spatApprox = TRUE,
                                    alphaMethod = "block",
                                    h = 15,
                                    storeSpatPredPara = TRUE,
                                    storeWeights = TRUE,
                                    alphasWeightsToFiles = FALSE)
save(regFixedL.simu.block, file="toyExNNGPblockFixedL.RData")
GibbsStepTimeFixedLblock <- regFixedL.simu.block$GibbsStepTime
save(GibbsStepTimeFixedLblock, file = "GibbsStepTimeFixedLblock.RData")
Diags.block <- diagnostics(regFixedL.simu.block,
                           diags = c("dic", "dinf", "meanIC", "waic"),
                           keepDeviance = TRUE)
save(Diags.block, file = "toyExNNGPblockFixedLDiags.RData")
Deviance.block <- as.mcmc(Diags.block$deviance)
save(Deviance.block, file = "toyExNNGPblockFixedLDeviance.RData")
spatpredFixedLblock <- predictNewLocFixedL(regFixedL.simu.block, 9,
                                           distOrigNew, distNewNew,
                                           NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                           seed = 29)
save(spatpredFixedLblock, file = "toyExNNGPspatpredFixedLblock.RData")
tempppredFixedLblock <- predictNewTime(regFixedL.simu.block, (trainingT+1):Nu, seed = 29)
save(tempppredFixedLblock, file = "toyExNNGPtempppredFixedLblock.RData")
fittedClusGpMat <- matrix(0, m, 3)
clusFixedL10 <- clusteringFixedL(regFixedL.simu.block, o = 1, nkeep = 10,
                                 nCent = numSpatOverallGroups)
fittedClusGpMat[,1] <- clusFixedL10$cluster
clusFixedL100 <- clusteringFixedL(regFixedL.simu.block, o = 1, nkeep = 100,
                                  nCent = numSpatOverallGroups)
fittedClusGpMat[,2] <- clusFixedL100$cluster
clusFixedL1000 <- clusteringFixedL(regFixedL.simu.block, o = 1, nkeep = 1000,
                                   nCent = numSpatOverallGroups)
fittedClusGpMat[,3] <- clusFixedL1000$cluster
weightsNumIter <- c(10, 100, 1000)
colnames(fittedClusGpMat) <- paste(weightsNumIter, "iterWeights")
save(fittedClusGpMat, file = "NNGPblockFixedLtoyExfittedClusGpMat.RData")
tempppredFixedLblock <- predictNewTime(regFixedL.simu.block, (trainingT+1):Nu, seed = 27)
save(tempppredFixedLblock, file = "s27toyExNNGPtempppredFixedLblock.RData")
tempppredFixedLblock <- predictNewTime(regFixedL.simu.block, (trainingT+1):Nu, seed = 19)
save(tempppredFixedLblock, file = "s19toyExNNGPtempppredFixedLblock.RData")
tempppredFixedLblock <- predictNewTime(regFixedL.simu.block, (trainingT+1):Nu, seed = 31)
save(tempppredFixedLblock, file = "s31toyExNNGPtempppredFixedLblock.RData")
```

```
spatpredFixedLblock <- predictNewLocFixedL(regFixedL.simu.block, 9,
                                           distOrigNew, distNewNew,
                                           NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                           seed = 27)
save(spatpredFixedLblock, file = "s27toyExNNGPspatpredFixedLblock.RData")
spatpredFixedLblock <- predictNewLocFixedL(regFixedL.simu.block, 9,
                                           distOrigNew, distNewNew,
                                           NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                           seed = 31)
save(spatpredFixedLblock, file = "s31toyExNNGPspatpredFixedLblock.RData")
spatpredFixedLblock <- predictNewLocFixedL(regFixedL.simu.block, 9,
                                           distOrigNew, distNewNew,
                                           NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                           seed = 19)
save(spatpredFixedLblock, file = "s19toyExNNGPspatpredFixedLblock.RData")
regFixedL.simu.sequen <- bfaFixedL(Y ~ 0, data = dat, dist = Dtraining, time = Time,
                                   K = K,
                                   starting = NULL, hypers = NULL, tuning = NULL,
                                   mcmc = MCMC,
                                   L = L,
                                   family = "normal",
                                   temporal.structure = "exponential",
                                   spatial.structure = "continuous",
                                   seed = 29,
                                   gamma.shrinkage = TRUE,
                                   include.time = TRUE,
                                   include.space = TRUE,
                                   clustering = TRUE,
                                   seasonPeriod = 1,
                                   equalTimeDist = TRUE,
                                   spatApprox = TRUE,
                                   alphaMethod = "sequential",
                                   h = 15,
                                   storeSpatPredPara = TRUE,
                                   storeWeights = TRUE,
                                   alphasWeightsToFiles = FALSE)
save(regFixedL.simu.sequen, file="toyExNNGPsequenFixedL.RData")
GibbsStepTimeFixedLsequen <- regFixedL.simu.sequen$GibbsStepTime
save(GibbsStepTimeFixedLsequen, file = "GibbsStepTimeFixedLsequen.RData")
Diags.sequen <- diagnostics(regFixedL.simu.sequen,
                            diags = c("dic", "dinf", "meanIC", "waic"),
                            keepDeviance = TRUE)
save(Diags.sequen, file = "toyExNNGPsequenFixedLDiags.RData")
Deviance.sequen <- as.mcmc(Diags.sequen$deviance)
save(Deviance.sequen, file = "toyExNNGPsequenFixedLDeviance.RData")
spatpredFixedLsequen <- predictNewLocFixedL(regFixedL.simu.sequen, 9,
                                            distOrigNew, distNewNew,
                                            NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                            seed = 29)
save(spatpredFixedLsequen, file = "toyExNNGPspatpredFixedLsequen.RData")
tempppredFixedLsequen <- predictNewTime(regFixedL.simu.sequen, (trainingT+1):Nu, seed = 29)
save(tempppredFixedLsequen, file = "toyExNNGPtempppredFixedLsequen.RData")
fittedClusGpMat <- matrix(0, m, 3)
```

```
clusFixedL10 <- clusteringFixedL(regFixedL.simu.sequen, o = 1, nkeep = 10,
                                 nCent = numSpatOverallGroups)
fittedClusGpMat[,1] <- clusFixedL10$cluster
clusFixedL100 <- clusteringFixedL(regFixedL.simu.sequen, o = 1, nkeep = 100,
                                  nCent = numSpatOverallGroups)
fittedClusGpMat[,2] <- clusFixedL100$cluster
clusFixedL1000 <- clusteringFixedL(regFixedL.simu.sequen, o = 1, nkeep = 1000,
                                   nCent = numSpatOverallGroups)
fittedClusGpMat[,3] <- clusFixedL1000$cluster
weightsNumIter <- c(10, 100, 1000)
colnames(fittedClusGpMat) <- paste(weightsNumIter, "iterWeights")
save(fittedClusGpMat, file = "NNGPsequenFixedLtoyExfittedClusGpMat.RData")
tempppredFixedLsequen <- predictNewTime(regFixedL.simu.sequen, (trainingT+1):Nu, seed = 27)
save(tempppredFixedLsequen, file = "s27toyExNNGPtempppredFixedLsequen.RData")
tempppredFixedLsequen <- predictNewTime(regFixedL.simu.sequen, (trainingT+1):Nu, seed = 19)
save(tempppredFixedLsequen, file = "s19toyExNNGPtempppredFixedLsequen.RData")
tempppredFixedLsequen <- predictNewTime(regFixedL.simu.sequen, (trainingT+1):Nu, seed = 31)
save(tempppredFixedLsequen, file = "s31toyExNNGPtempppredFixedLsequen.RData")
spatpredFixedLsequen <- predictNewLocFixedL(regFixedL.simu.sequen, 9,
                                            distOrigNew, distNewNew,
                                            NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                            seed = 27)
save(spatpredFixedLsequen, file = "s27toyExNNGPspatpredFixedLsequen.RData")
spatpredFixedLsequen <- predictNewLocFixedL(regFixedL.simu.sequen, 9,
                                            distOrigNew, distNewNew,
                                            NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                            seed = 31)
save(spatpredFixedLsequen, file = "s31toyExNNGPspatpredFixedLsequen.RData")
spatpredFixedLsequen <- predictNewLocFixedL(regFixedL.simu.sequen, 9,
                                            distOrigNew, distNewNew,
                                            NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                            seed = 19)
save(spatpredFixedLsequen, file = "s19toyExNNGPspatpredFixedLsequen.RData")
regVaryLj.simu.sequen <- bfaVaryingLjs(Y ~ 0, data = dat, dist = Dtraining, time = Time,
                                       K = K,
                                       starting = NULL, hypers = NULL, tuning = NULL,
                                       mcmc = MCMC,
                                       LjVec = LjVec,
                                       family = "normal",
                                       temporal.structure = "exponential",
                                       spatial.structure = "continuous",
                                       seed = 29,
                                       gamma.shrinkage = TRUE,
                                       include.time = TRUE,
                                       include.space = TRUE,
                                       seasonPeriod = 1,
                                       equalTimeDist = TRUE,
                                       spatApprox = TRUE,
                                       alphaSequen = TRUE,
                                       h = 15,
                                       storeSpatPredPara = TRUE,
                                       storeWeights = TRUE)
save(regVaryLj.simu.sequen, file="toyExNNGPsequenVaryLj.RData")
```

```
GibbsStepTimeVaryLjSequen <- regVaryLj.simu.sequen$GibbsStepTime
save(GibbsStepTimeVaryLjSequen, file = "GibbsStepTimeVaryLjSequen.RData")
Diags.sequenVaryLj <- diagnostics(regVaryLj.simu.sequen,
                                  diags = c("dic", "dinf", "meanIC", "waic"),
                                  keepDeviance = TRUE)
save(Diags.sequenVaryLj, file = "toyExNNGPsequenVaryLjDiags.RData")
Deviance.sequenVaryLj <- as.mcmc(Diags.sequenVaryLj$deviance)
save(Deviance.sequenVaryLj, file = "toyExNNGPsequenVaryLjDeviance.RData")
spatpredVaryLj <- predictNewLocVaryLj(regVaryLj.simu.sequen, 9,
                                      distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 29)
save(spatpredVaryLj, file = "toyExNNGPspatpredVaryLj.RData")
tempppredVaryLj <- predictNewTime(regVaryLj.simu.sequen, (trainingT+1):Nu, seed = 29)
save(tempppredVaryLj, file = "toyExNNGPtempppredVaryLj.RData")
fittedClusGpMat <- matrix(0, m, 3)
clusVaryLj10 <- clusteringVaryLj(regVaryLj.simu.sequen, o = 1, nkeep = 10,
                                 nCent = numSpatOverallGroups)
fittedClusGpMat[,1] <- clusVaryLj10$cluster
clusVaryLj100 <- clusteringVaryLj(regVaryLj.simu.sequen, o = 1, nkeep = 100,
                                  nCent = numSpatOverallGroups)
fittedClusGpMat[,2] <- clusVaryLj100$cluster
clusVaryLj1000 <- clusteringVaryLj(regVaryLj.simu.sequen, o = 1, nkeep = 1000,
                                   nCent = numSpatOverallGroups)
fittedClusGpMat[,3] <- clusVaryLj1000$cluster
weightsNumIter <- c(10, 100, 1000)
colnames(fittedClusGpMat) <- paste(weightsNumIter, "iterWeights")
save(fittedClusGpMat, file = "NNGPsequenVaryLjtoyExfittedClusGpMat.RData")
tempppredVaryLj <- predictNewTime(regVaryLj.simu.sequen, (trainingT+1):Nu, seed = 27)
save(tempppredVaryLj, file = "s27toyExNNGPtempppredVaryLj.RData")
tempppredVaryLj <- predictNewTime(regVaryLj.simu.sequen, (trainingT+1):Nu, seed = 19)
save(tempppredVaryLj, file = "s19toyExNNGPtempppredVaryLj.RData")
tempppredVaryLj <- predictNewTime(regVaryLj.simu.sequen, (trainingT+1):Nu, seed = 31)
save(tempppredVaryLj, file = "s31toyExNNGPtempppredVaryLj.RData")
spatpredVaryLj <- predictNewLocVaryLj(regVaryLj.simu.sequen, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 27)
save(spatpredVaryLj, file = "s27toyExNNGPspatpredVaryLj.RData")
spatpredVaryLj <- predictNewLocVaryLj(regVaryLj.simu.sequen, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 31)
save(spatpredVaryLj, file = "s31toyExNNGPspatpredVaryLj.RData")
spatpredVaryLj <- predictNewLocVaryLj(regVaryLj.simu.sequen, 9, distOrigNew, distNewNew,
                                      NewX = NULL, NewTrials = NULL, Verbose = TRUE,
                                      seed = 19)
save(spatpredVaryLj, file = "s19toyExNNGPspatpredVaryLj.RData")
library(spBFA)
# to compare overall runtime, Diags, Deviances, and future time prediction results
reg.simu <- bfa_sp(Y ~ 0, data = dat, dist = Dtraining, time = Time,  K = K,
                   starting = NULL, hypers = NULL, tuning = NULL, mcmc = MCMC,
                   L = L,
                   family = "normal",
                   trials = NULL,
```

```
                    temporal.structure = "exponential",
                    spatial.structure = "continuous",
                    seed = 29,
                    gamma.shrinkage = TRUE,
                    include.space = TRUE,
                    clustering = TRUE)
save(reg.simu, file="toyExspBFAL10.RData")
Diags <- spBFA::diagnostics(reg.simu, diags = c("dic", "dinf", "waic"),
                            keepDeviance = TRUE)
save(Diags, file="toyExspBFAL10Diags.RData")
Deviance <- as.mcmc(Diags$deviance)
save(Deviance, file = "toyExspBFAL10Deviance.RData")
tempppredspBFAL10 <- predict(reg.simu, (trainingT+1):Nu, seed = 29) # from spBFA
save(tempppredspBFAL10, file = "toyExspBFAL10temppred.RData")
tempppredspBFAL10 <- predict(reg.simu, (trainingT+1):Nu, seed = 19)
save(tempppredspBFAL10, file = "s19toyExspBFAL10temppred.RData")
tempppredspBFAL10 <- predict(reg.simu, (trainingT+1):Nu, seed = 27)
save(tempppredspBFAL10, file = "s27toyExspBFAL10temppred.RData")
tempppredspBFAL10 <- predict(reg.simu, (trainingT+1):Nu, seed = 31)
save(tempppredspBFAL10, file = "s31toyExspBFAL10temppred.RData")
reg.simu.LInf <- bfa_sp(Y ~ 0, data = dat, dist = Dtraining, time = Time,  K = K,
                    starting = NULL, hypers = NULL, tuning = NULL, mcmc = MCMC,
                    L = Inf,
                    family = "normal",
                    trials = NULL,
                    temporal.structure = "exponential",
                    spatial.structure = "continuous",
                    seed = 29,
                    gamma.shrinkage = TRUE,
                    include.space = TRUE,
                    clustering = TRUE)
save(reg.simu.LInf, file="toyExspBFALInf.RData")
Diags.LInf <- spBFA::diagnostics(reg.simu.LInf, diags = c("dic", "dinf", "waic"),
                                 keepDeviance = TRUE)
save(Diags.LInf, file="toyExspBFALInfDiags.RData")
Deviance.LInf <- as.mcmc(Diags.LInf$deviance)
save(Deviance.LInf, file = "toyExspBFALInfDeviance.RData")
tempppredspBFALInf <- predict(reg.simu.LInf, (trainingT+1):Nu, seed = 29)
save(tempppredspBFALInf, file = "toyExspBFALInftemppred.RData")
tempppredspBFALInf <- predict(reg.simu.LInf, (trainingT+1):Nu, seed = 19)
save(tempppredspBFALInf, file = "s19toyExspBFALInftemppred.RData")
tempppredspBFALInf <- predict(reg.simu.LInf, (trainingT+1):Nu, seed = 27)
save(tempppredspBFALInf, file = "s27toyExspBFALInftemppred.RData")
tempppredspBFALInf <- predict(reg.simu.LInf, (trainingT+1):Nu, seed = 31)
save(tempppredspBFALInf, file = "s31toyExspBFALInftemppred.RData")
```

Comparing `spatempBFA` and `spBFA`

Overall Model Fitting Time

```
setwd(paste(projDirec, "simu/mainScalabilityVerificationSimu/toyExample", sep = "/"))
setwd("spBFA")
```

```
load("toyExspBFAL10.RData"); reg.simu$runtime # 7.24 days
```

```
## [1] "Model runtime: 7.24 days"
```

```
load("toyExspBFALInf.RData"); reg.simu.LInf$runtime # 3.08 days
```

```
## [1] "Model runtime: 3.08 days"
```

```
load("toyExspBFAL10Diags.RData")
load("toyExspBFAL10Deviance.RData")
load("toyExspBFAL10temppred.RData")
load("toyExspBFALInfDiags.RData")
load("toyExspBFALInfDeviance.RData")
load("toyExspBFALInftemppred.RData")
spBFADiags <- Diags; spBFADeviance <- Deviance
spBFADiagsLInf <- Diags.LInf; spBFADevianceLInf <- Deviance.LInf
rm(Diags, Diags.LInf, Deviance, Deviance.LInf)
setwd("../fullGPfixedL")
load("toyExfullGPfixedL.RData"); regFixedL.simu$runtime # 15.77 hours
```

```
## [1] "Model runtime: 15.77 hours"
```

```
load("toyExfullGPfixedLDiags.RData")
load("toyExfullGPfixedLDeviance.RData")
load("GibbsStepTimeFixedLfullGP.RData")
load("toyExFullGPtemppredFixedL.RData")
load("toyExFullGPspatpredFixedL.RData")
load("fullGPfixedLtoyExfittedClusGpMat.RData")
fittedClusGpMat.fullGPfixedL <- fittedClusGpMat
setwd("../NNGPblockFixedL")
load("toyExNNGPblockFixedL.RData"); regFixedL.simu.block$runtime # 15.46 hours
```

```
## [1] "Model runtime: 15.46 hours"
```

```
load("toyExNNGPblockFixedLDiags.RData")
load("toyExNNGPblockFixedLDeviance.RData")
load("GibbsStepTimeFixedLblock.RData")
load("toyExNNGPtemppredFixedLblock.RData")
load("toyExNNGPspatpredFixedLblock.RData")
load("NNGPblockFixedLtoyExfittedClusGpMat.RData")
fittedClusGpMat.NNGPblockFixedL <- fittedClusGpMat
setwd("../NNGPsequenFixedL")
load("toyExNNGPsequenFixedL.RData"); regFixedL.simu.sequen$runtime # 15.19 hours
```

```
## [1] "Model runtime: 15.19 hours"
```

```
load("toyExNNGPsequenFixedLDiags.RData")
load("toyExNNGPsequenFixedLDeviance.RData")
load("GibbsStepTimeFixedLsequen.RData")
load("toyExNNGPtemppredFixedLsequen.RData")
load("toyExNNGPspatpredFixedLsequen.RData")
load("NNGPsequenFixedLtoyExfittedClusGpMat.RData")
fittedClusGpMat.NNGPsequenFixedL <- fittedClusGpMat
setwd("../NNGPsequenVaryLj")
load("toyExNNGPsequenVaryLj.RData"); regVaryLj.simu.sequen$runtime # 14.58 hours
```

```
## [1] "Model runtime: 14.58 hours"
```

```
load("toyExNNGPsequenVaryLjDiags.RData")
load("toyExNNGPsequenVaryLjDeviance.RData")
load("GibbsStepTimeVaryLjsequen.RData")
load("toyExNNGPtemppredVaryLj.RData")
load("toyExNNGPspatpredVaryLj.RData")
load("NNGPsequenVaryLjtoyExfittedClusGpMat.RData")
fittedClusGpMat.NNGPsequenVaryLj <- fittedClusGpMat
rm(fittedClusGpMat)
rm(reg.simu, reg.simu.LInf)
rm(regFixedL.simu, regFixedL.simu.block, regFixedL.simu.sequen, regVaryLj.simu.sequen)
setwd("..")
```

As expected, our 4 methods are several times faster than `spBFALInf` and more than 10 times faster than `spBFAL10` in terms of main model fitting.

```
round(apply(GibbsStepTimeFixedLfullGP, 2, summary), digits = 3)
```

```
##               z     xi   theta delta  alpha kappa    rho     eta upsilon   psi
## Min.     21.000 12.000 234.000 0.000 19.000 1.000 10.000 234.000       1 3.000
## 1st Qu.  23.000 13.000 241.000 1.000 19.000 2.000 11.000 242.000       1 3.000
## Median   24.000 13.000 243.000 1.000 20.000 2.000 11.000 243.000       1 3.000
## Mean     24.408 13.164 243.285 0.986 19.829 2.006 11.389 243.966       1 3.242
## 3rd Qu.  25.000 13.000 245.000 1.000 20.000 2.000 11.000 245.000       1 3.000
## Max.    217.000 24.000 276.000 2.000 25.000 5.000 17.000 268.000       2 6.000
```

```
round(apply(GibbsStepTimeFixedLblock, 2, summary), digits = 3)
```

```
##               z      xi   theta delta  alpha kappa   rho     eta upsilon  psi
## Min.     21.000  12.000 235.000 0.000 17.000 1.000 4.000 236.000       1 3.00
## 1st Qu.  22.000  13.000 241.000 1.000 17.000 1.000 5.000 241.000       1 3.00
## Median   24.000  13.000 242.000 1.000 18.000 1.000 5.000 243.000       1 3.00
## Mean     24.158  13.235 242.981 0.977 17.728 1.004 4.961 243.321       1 3.35
## 3rd Qu.  25.000  13.000 244.000 1.000 18.000 1.000 5.000 245.000       1 4.00
## Max.    240.000 200.000 283.000 2.000 22.000 4.000 6.000 273.000       2 6.00
```

```
round(apply(GibbsStepTimeFixedLsequen, 2, summary), digits = 3)
```

```
##               z     xi   theta delta alpha kappa   rho     eta upsilon   psi
## Min.     21.000 12.000 235.000 0.000  7.00 1.000 4.000 236.000       1 3.000
## 1st Qu.  23.000 13.000 241.000 1.000  7.00 1.000 5.000 241.000       1 3.000
## Median   24.000 13.000 243.000 1.000  7.00 1.000 5.000 243.000       1 3.000
## Mean     24.079 13.129 243.138 0.986  7.14 1.007 4.965 243.645       1 3.299
## 3rd Qu.  25.000 13.000 244.000 1.000  7.00 1.000 5.000 245.000       1 4.000
## Max.    214.000 18.000 269.000 1.000 11.00 2.000 8.000 281.000       2 6.000
```

```
round(apply(GibbsStepTimeVaryLjSequen, 2, summary), digits = 3)
```

```
##         u      xi   theta delta   alpha kappa    rho     eta upsilon   psi
## Min.    0   3.000 234.000 0.000  18.000 1.000  4.000 235.000       1 3.000
## 1st Qu. 0   3.000 240.000 1.000  19.000 1.000  5.000 241.000       1 3.000
## Median  0   3.000 242.000 1.000  20.000 1.000  5.000 242.000       1 3.000
## Mean    0   3.279 242.254 0.934  21.003 1.001  4.986 242.892       1 3.575
## 3rd Qu. 0   3.000 244.000 1.000  21.000 1.000  5.000 244.000       1 4.000
## Max.    1 189.000 283.000 1.000 211.000 3.000 21.000 275.000       2 6.000
```

```
rm(GibbsStepTimeFixedLfullGP, GibbsStepTimeFixedLblock)
rm(GibbsStepTimeFixedLsequen, GibbsStepTimeVaryLjSequen)
```

The version of our `spatempBFA` package run for this toy example records Gibbs sampler time (in milliseconds) at each kept post-burn-in MCMC iteration for 10 key parameters $z^o_{jl_j}(\boldsymbol{s}_i)$'s or $u^o_j(\boldsymbol{s}_i)$'s, $\xi^o_j(\boldsymbol{s}_i)$'s, θ_{jl_j}'s, $\delta_{1:k}$, ρ, κ, $\alpha^o_{jl_j}(\boldsymbol{s}_i)$'s, ψ, Υ, and $\boldsymbol{\eta}_t$'s. Some summary statistics given above correspond well to their counterparts in Section 6.1's simulation experiments in terms of relative magnitudes.

- Compared to their `fullGPfixedL` counterparts, `NNGPblockFixedL`'s Gibbs sampler steps corresponding to ρ and κ are evidently accelerated by our spatial NNGP prior;
- The only Gibbs sampler step time that should clearly differ between `NNGPblockFixedL` and `NNGPsequenFixedL` is the step updating all $\alpha^o_{jl_j}(\boldsymbol{s}_i)$'s, which result from whether we adopt our sequential updating method or not. Here, `NNGPsequenFixedL` is mostly more than two times faster than `NNGPblockFixedL` for the posterior sampling step corresponding to $\alpha^o_{jl_j}(\boldsymbol{s}_i)$'s;
- Thanks to our slice sampling approach, `NNGPsequenVaryLj`'s Gibbs sampler steps for $u^o_j(\boldsymbol{s}_i)$'s and $\xi^o_j(\boldsymbol{s}_i)$'s are significantly faster than `NNGPsequenFixedL`'s Gibbs sampler steps for $z^o_{jl_j}(\boldsymbol{s}_i)$'s and $\xi^o_j(\boldsymbol{s}_i)$'s. It turns out that `NNGPsequenVaryLj`'s Gibbs sampler step for $\alpha^o_{jl_j}(\boldsymbol{s}_i)$'s is slower than its `NNGPsequenFixedL` counterpart, indicating that inefficiencies caused by case discussion, calculating all required upper or lower bounds, and rejection sampling outweigh acceleration brought about by slice sampling's ensured non-increasing posterior samples for L_j's through the MCMC iterations;
- There aren't any significant differences between our 4 methods regarding posterior sampling time for the 3 temporal parameters ψ, Υ, and $\boldsymbol{\eta}_t$'s.

Diagnostics and Posterior Deviances

```
print(Diags[setdiff(names(Diags), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -174384.7
##
## $dic$pd
## [1] 621.2673
##
##
## $dinf
## $dinf$p
## [1] 1770.739
##
## $dinf$g
## [1] 1047.284
##
## $dinf$dinf
## [1] 2818.023
##
##
## $meanIC
## $meanIC$postMSE
## [1] 0.02668247
##
## $meanIC$postVar
## [1] 0.01676836
##
##
## $waic
## $waic$waic
##           [,1]
```

```
## [1,] -157883.7
##
## $waic$p_waic
##          [,1]
## [1,] 9000.328
##
## $waic$lppd
## [1] 87942.2
##
## $waic$p_waic_1
## [1] 878.4693
```

```
print(Diags.block[setdiff(names(Diags.block), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -174345.7
##
## $dic$pd
## [1] 663.04
##
##
## $dinf
## $dinf$p
## [1] 1770.63
##
## $dinf$g
## [1] 1047.284
##
## $dinf$dinf
## [1] 2817.914
##
##
## $meanIC
## $meanIC$postMSE
## [1] 0.02668144
##
## $meanIC$postVar
## [1] 0.01676733
##
##
## $waic
## $waic$waic
##           [,1]
## [1,] -158075.7
##
## $waic$p_waic
##          [,1]
## [1,] 8772.018
##
## $waic$lppd
## [1] 87809.87
##
## $waic$p_waic_1
## [1] 611.0213
```

```r
print(Diags.sequen[setdiff(names(Diags.sequen), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -174361.8
##
## $dic$pd
## [1] 646.9392
##
##
## $dinf
## $dinf$p
## [1] 1770.648
##
## $dinf$g
## [1] 1047.291
##
## $dinf$dinf
## [1] 2817.939
##
##
## $meanIC
## $meanIC$postMSE
## [1] 0.02668167
##
## $meanIC$postVar
## [1] 0.0167675
##
##
## $waic
## $waic$waic
##           [,1]
## [1,] -158203.9
##
## $waic$p_waic
##          [,1]
## [1,] 8705.174
##
## $waic$lppd
## [1] 87807.12
##
## $waic$p_waic_1
## [1] 605.4927
```

```r
print(Diags.sequenVaryLj[setdiff(names(Diags.sequenVaryLj), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -174348
##
## $dic$pd
## [1] 662.9201
##
##
```

```
## $dinf
## $dinf$p
## [1] 1770.53
##
## $dinf$g
## [1] 1047.274
##
## $dinf$dinf
## [1] 2817.804
##
##
## $meanIC
## $meanIC$postMSE
## [1] 0.02668039
##
## $meanIC$postVar
## [1] 0.01676638
##
##
## $waic
## $waic$waic
##           [,1]
## [1,] -157898.2
##
## $waic$p_waic
##          [,1]
## [1,] 8917.304
##
## $waic$lppd
## [1] 87866.43
##
## $waic$p_waic_1
## [1] 721.9712
```

```
print(spBFADiags[setdiff(names(spBFADiags), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -174440.4
##
## $dic$pd
## [1] 564.7672
##
##
## $dinf
## $dinf$p
## [1] 1771.181
##
## $dinf$g
## [1] 1047.208
##
## $dinf$dinf
## [1] 2818.39
##
##
```

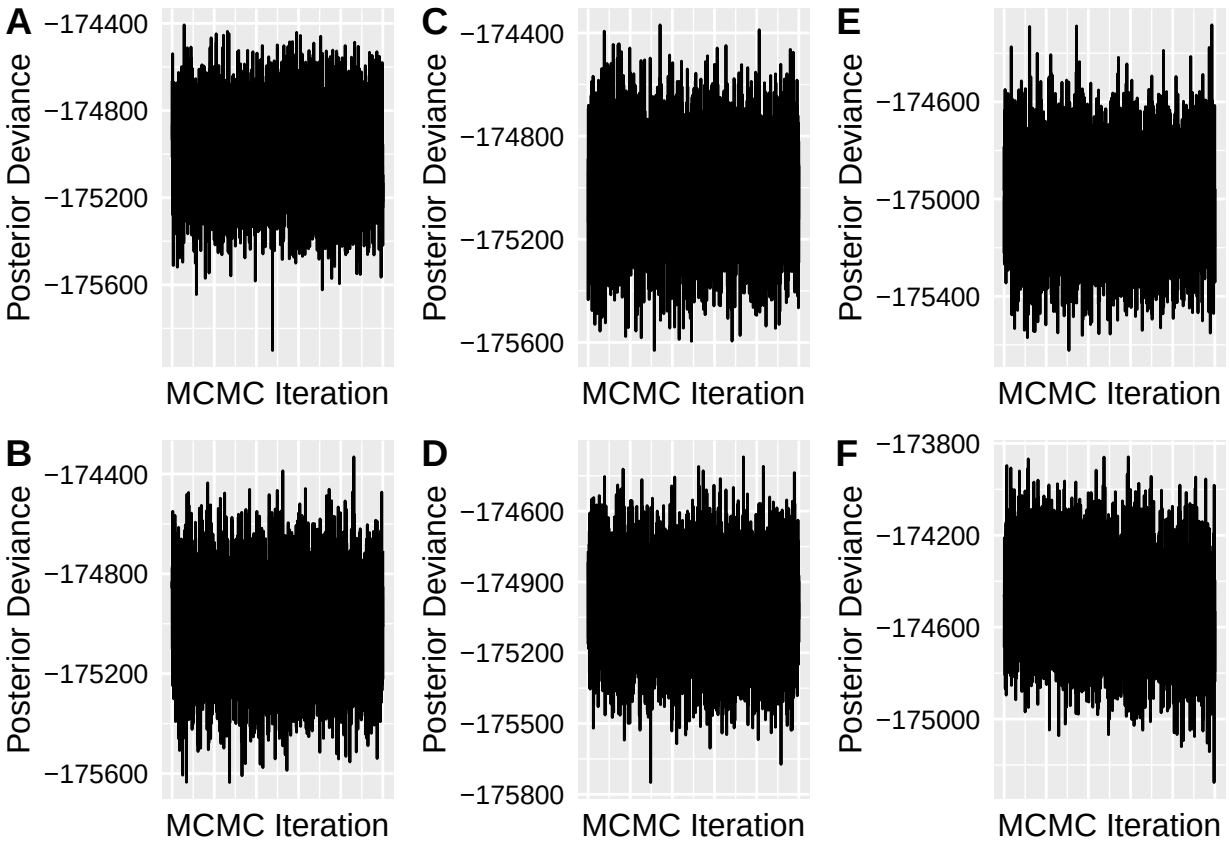
```
## $waic
## $waic$waic
## [1] -157419.3
##
## $waic$p_waic
## [1] 9093.332
##
## $waic$lppd
## [1] 87802.98
##
## $waic$p_waic_1
## [1] 600.7735
```

```
print(spBFADiagsLInf[setdiff(names(spBFADiagsLInf), "deviance")])
```

```
## $dic
## $dic$dic
## [1] -173945.4
##
## $dic$pd
## [1] 557.425
##
##
## $dinf
## $dinf$p
## [1] 1777.412
##
## $dinf$g
## [1] 1053.143
##
## $dinf$dinf
## [1] 2830.555
##
##
## $waic
## $waic$waic
## [1] -156695.4
##
## $waic$p_waic
## [1] 9281.774
##
## $waic$lppd
## [1] 87629.49
##
## $waic$p_waic_1
## [1] 756.1147
```

```
rm(Diags, Diags.block, Diags.sequen, Diags.sequenVaryLj, spBFADiags, spBFADiagsLInf)
NKeep <- 5000
postDeviancesDF <- data.frame(MCMCiter = 1:NKeep, fullGPfixedL = as.vector(Deviance),
                              NNGPblockFixedL = as.vector(Deviance.block),
                              NNGPsequenFixedL = as.vector(Deviance.sequen),
                              NNGPsequenVaryLj = as.vector(Deviance.sequenVaryLj),
                              spBFAL10 = as.vector(spBFADeviance),
                              spBFALInf = as.vector(spBFADevianceLInf))
```

```
rm(Deviance, Deviance.block, Deviance.sequen, Deviance.sequenVaryLj)
rm(spBFADeviance, spBFADevianceLInf)
fullGPfixedLpostDeviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = fullGPfixedL)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
NNGPblockFixedLpostDeviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = NNGPblockFixedL)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
NNGPsequenFixedLpostDeviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = NNGPsequenFixedL)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
NNGPsequenVaryLjpostDeviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = NNGPsequenVaryLj)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
spBFAL10Deviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = spBFAL10)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
spBFALInfDeviances <- ggplot(postDeviancesDF) +
  geom_line(aes(x = MCMCiter, y = spBFALInf)) +
  labs(x = "MCMC Iteration", y = "Posterior Deviance") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        axis.text.x = element_blank(),
        axis.text.y = element_text(size = 10, color = "black"),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank())
ggarrange(fullGPfixedLpostDeviances, NNGPsequenFixedLpostDeviances, spBFAL10Deviances,
          NNGPblockFixedLpostDeviances, NNGPsequenVaryLjpostDeviances, spBFALInfDeviances,
          labels = c("A", "C", "E", "B", "D", "F"), align = "v", ncol = 3, nrow = 2)
```




```
#ggsave("toyExPostDeviancesSmall.png", width = 24, height = 16, units = "cm")
rm(fullGPfixedLpostDeviances, NNGPblockFixedLpostDeviances)
rm(NNGPsequenFixedLpostDeviances, NNGPsequenVaryLjpostDeviances)
rm(postDeviancesDF, spBFAL10Deviances, spBFALInfDeviances)
```

All our 4 methods' diagnostics and converged posterior deviances are comparable to their `spBFAL10` counterparts and clearly better than their `spBFALInf` counterparts.

Predictions for all 352 Training Locations at the 10 Future Time Points

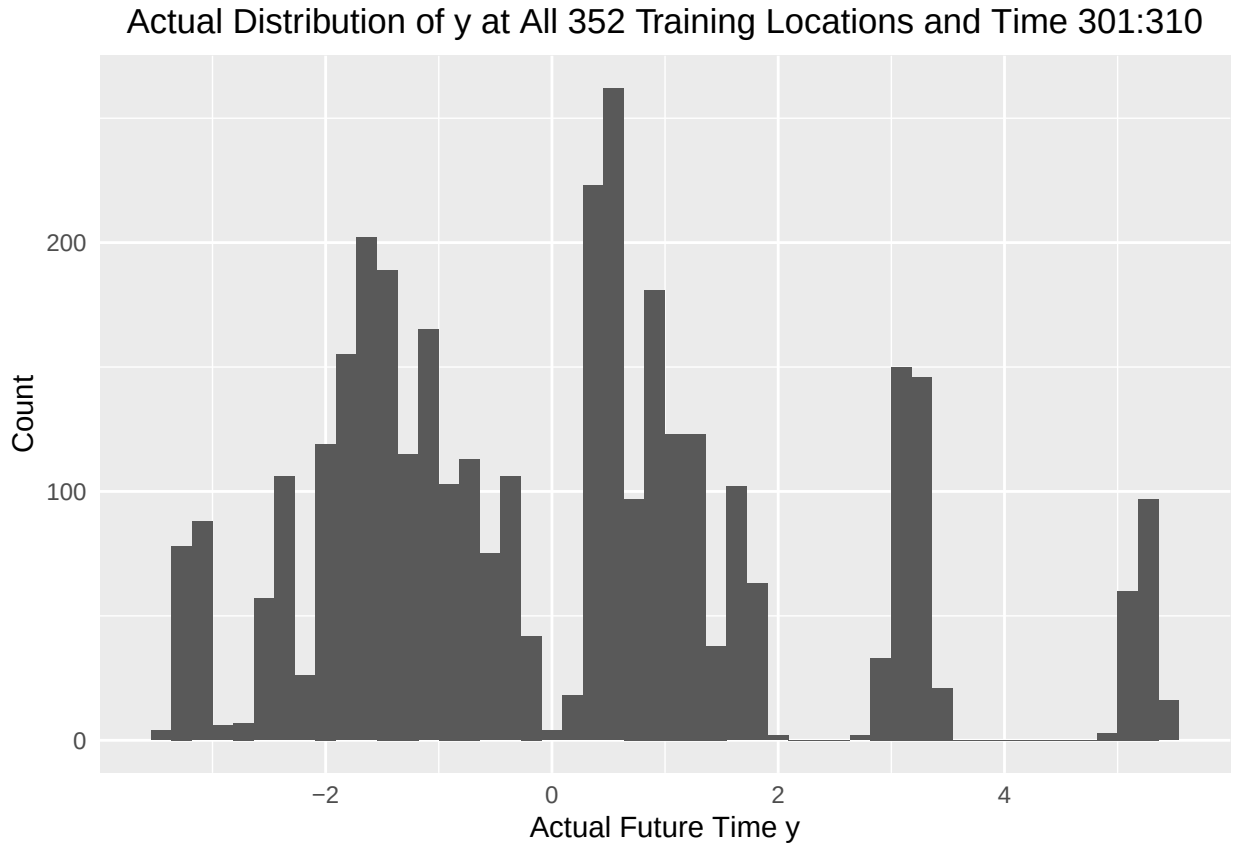
```
summary(YtestingTemp)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -3.40731 -1.52202  0.04399  0.10879  1.14903  5.49685
```

```
summary(abs(YtestingTemp))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.008901 0.766441 1.371636 1.667743 2.200213 5.496845
```

```
YtestingTempDF <- data.frame(YtestingTemp = YtestingTemp)
ggplot(YtestingTempDF) + geom_histogram(aes(x = YtestingTemp), bins = 50) +
  labs(y = "Count", x = "Actual Future Time y") +
  theme(axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  ggtitle("Actual Distribution of y at All 352 Training Locations and Time 301:310")
```




```
tempPredMetricMat <- matrix(0, 3, 6,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj",
                                              "spBFAL10", "spBFALInf")))
# testingT = 10
ytempPredList = tempppredFixedL$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 1] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 1] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLblock$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 2] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 2] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLsequen$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
```

```
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 3] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 3] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredVaryLj$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 4] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 4] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFAL10$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 5] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 5] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 5] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFALInf$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 6] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 6] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 6] = mean(apply(ytempPred, 1, var))
tempPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     3.749685        3.753012         3.729849         3.728981
## postMSE        18.518662       18.754345        18.299211        19.340736
## postVar        14.771932       15.004334        14.572277        15.614878
##              spBFAL10 spBFALInf
## postMeanMSE  4.009299  11.55326
## postMSE     19.409493 164.42639
## postVar     15.403274 152.90371
```

Temporal prediction results from all our 4 methods are close and significantly better than their `spBFALInf` counterparts. Among our 4 methods and `spBFAL10`, `NNGPsequenFixedL` appears to be the best, followed by `fullGPfixedL`, `NNGPblockFixedL`, and finally `NNGPsequenVaryLj` and `spBFAL10`. Similar conclusions can be drawn from future-time prediction metrics calculated from temporal prediction objects generated using 3 other random seeds, as presented below.

```
setwd("fullGPfixedL")
load("s19toyExFullGPtemppredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s19toyExNNGPtemppredFixedLblock.RData")
```

```
setwd("../NNGPsequenFixedL")
load("s19toyExNNGPtemppredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s19toyExNNGPtemppredVaryLj.RData")
setwd("../spBFA")
load("s19toyExspBFAL10temppred.RData")
load("s19toyExspBFALInftemppred.RData")
setwd("..")
tempPredMetricMat <- matrix(0, 3, 6,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj",
                                              "spBFAL10", "spBFALInf")))
# testingT = 10
ytempPredList = tempppredFixedL$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 1] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 1] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLblock$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 2] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 2] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLsequen$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 3] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 3] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredVaryLj$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 4] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 4] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFAL10$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
```

```
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 5] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 5] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 5] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFALInf$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 6] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 6] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 6] = mean(apply(ytempPred, 1, var))
tempPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     3.760867        3.765498         3.722402         3.730441
## postMSE        18.522127       18.762238        18.296852        19.352976
## postVar        14.764213       14.999740        14.577366        15.625660
##              spBFAL10 spBFALInf
## postMeanMSE  4.037946  10.79162
## postMSE     19.355234 166.43690
## postVar     15.320353 155.67641
```

```
setwd("fullGPfixedL")
load("s27toyExFullGPtemppredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s27toyExNNGPtemppredFixedLblock.RData")
setwd("../NNGPsequenFixedL")
load("s27toyExNNGPtemppredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s27toyExNNGPtemppredVaryLj.RData")
setwd("../spBFA")
load("s27toyExspBFAL10temppred.RData")
load("s27toyExspBFALInftemppred.RData")
setwd("..")
tempPredMetricMat <- matrix(0, 3, 6,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj",
                                              "spBFAL10", "spBFALInf")))
# testingT = 10
ytempPredList = tempppredFixedL$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 1] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 1] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLblock$Y
```

```
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 2] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 2] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLsequen$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 3] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 3] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredVaryLj$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 4] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 4] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFAL10$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 5] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 5] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 5] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFALInf$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 6] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 6] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 6] = mean(apply(ytempPred, 1, var))
tempPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     3.910571         3.91431         3.886396         3.890572
## postMSE        18.929454        19.14805        18.693109        19.765437
## postVar        15.021888        15.23679        14.809675        15.878040
##              spBFAL10 spBFALInf
## postMeanMSE  4.178536  12.44008
## postMSE     19.742454 171.27550
## postVar     15.567031 158.86719
```

```
setwd("fullGPfixedL")
load("s31toyExFullGPtemppredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s31toyExNNGPtemppredFixedLblock.RData")
setwd("../NNGPsequenFixedL")
load("s31toyExNNGPtemppredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s31toyExNNGPtemppredVaryLj.RData")
setwd("../spBFA")
load("s31toyExspBFAL10temppred.RData"); load("s31toyExspBFALInftemppred.RData")
setwd("..")
tempPredMetricMat <- matrix(0, 3, 6,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj",
                                              "spBFAL10", "spBFALInf")))
# testingT = 10
ytempPredList = tempppredFixedL$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 1] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 1] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLblock$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 2] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 2] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredFixedLsequen$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 3] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 3] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredVaryLj$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 4] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 4] = mean(apply(ytempPred, 1, var))
```

```
ytempPredList = tempppredspBFAL10$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 5] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 5] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 5] = mean(apply(ytempPred, 1, var))
ytempPredList = tempppredspBFALInf$Y
ytempPred <- t(matrix(unlist(ytempPredList), ncol = testingT * m * O))
# (testingT * m * O) * NKeep
ytempPredMean <- apply(ytempPred, 1, mean)
# of length N = testingT * O * m (rowMeans) note that O = 1 here
tempPredMetricMat[1, 6] <- mean((ytempPredMean - YtestingTemp)^2)
diffMat <- sweep(ytempPred, 1, YtestingTemp, "-")
tempPredMetricMat[2, 6] = mean(rowMeans(diffMat^2))
tempPredMetricMat[3, 6] = mean(apply(ytempPred, 1, var))
rm(tempppredFixedL, tempppredFixedLblock, tempppredFixedLsequen, tempppredVaryLj)
rm(tempppredspBFAL10, tempppredspBFALInf)
tempPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     3.868427        3.869237         3.831659         3.839874
## postMSE        18.723844       18.970039        18.502927        19.570108
## postVar        14.858389       15.103822        14.674203        15.733381
##              spBFAL10 spBFALInf
## postMeanMSE  4.127419  11.33985
## postMSE     19.569936 170.15262
## postVar     15.445606 158.84453
```

Spatial Prediction and Clustering by spatempBFA

Predictions for all 300 Training Time Points at the 9 Testing Locations

Spatial Prediction Time Breakdown for Our 4 Methods

```
spatPredKrigTime <- matrix(0, 2, 4,
                           dimnames = list(c("alpha", "weightsXiLambda"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
spatPredKrigTime[1, 1] = spatpredFixedL$alphaKrigTime
spatPredKrigTime[2, 1] = spatpredFixedL$weightsXiLambdaKrigTime
spatPredKrigTime[1, 2] = spatpredFixedLblock$alphaKrigTime
spatPredKrigTime[2, 2] = spatpredFixedLblock$weightsXiLambdaKrigTime
spatPredKrigTime[1, 3] = spatpredFixedLsequen$alphaKrigTime
spatPredKrigTime[2, 3] = spatpredFixedLsequen$weightsXiLambdaKrigTime
spatPredKrigTime[1, 4] = spatpredVaryLj$alphaKrigTime
spatPredKrigTime[2, 4] = spatpredVaryLj$weightsXiLambdaKrigTime
spatPredKrigTime
```

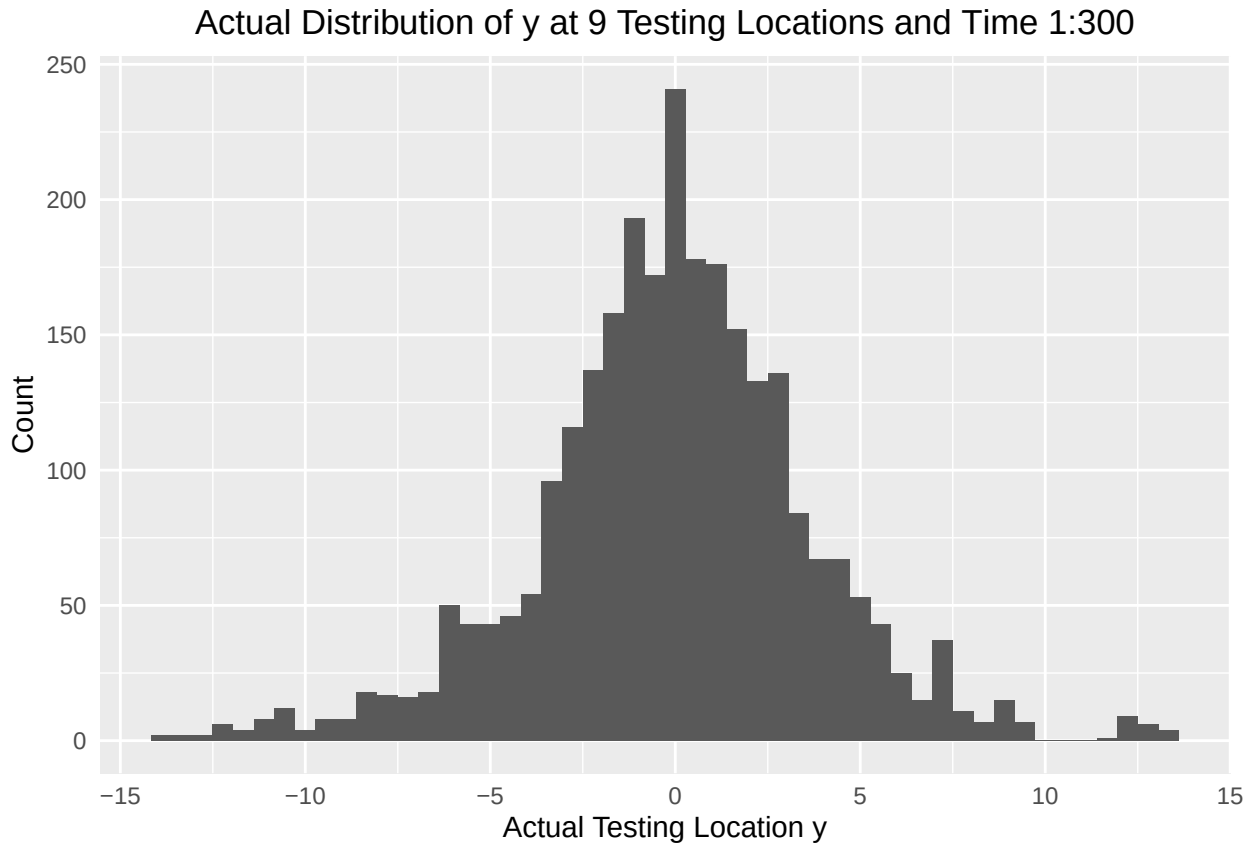
```
##                 fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## alpha                  63988            2838             2888             2492
## weightsXiLambda          118             129              139               76
```

The results correspond well to what we have derived in Appendix H.

- `alphaKrigTime` corresponding to `NNGPblockFixedL` and `NNGPsequenFixedL` should be close and markedly (more than 20 times) smaller than that of `fullGPfixedL`. `NNGPsequenVaryLj`'s `alphaKrigTime` should be even smaller;
- `weightsXiLambdaKrigTime` corresponding to `NNGPsequenVaryLj` should be smaller than their counterparts from the other 3 methods.

Spatial Prediction Metrics

```
YtestingSpatDF <- data.frame(YtestingSpat = YtestingSpat)
ggplot(YtestingSpatDF) + geom_histogram(aes(x = YtestingSpat), bins = 50) +
  labs(y = "Count", x = "Actual Testing Location y") +
  theme(axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  ggtitle("Actual Distribution of y at 9 Testing Locations and Time 1:300")
```




```
summary(YtestingSpat)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -13.798117  -2.066726   0.000602  -0.047764   2.155001  13.432128
```

```
summary(abs(YtestingSpat))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  0.000114  0.958706  2.117600  2.785832  3.785223 13.798117
```

```
spatPredMetricMat <- matrix(0, 3, 4,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj")))
testingLocNum <- M - m
ylocPredList = spatpredFixedL$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 1] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 1] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLblock$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 2] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 2] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLsequen$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 3] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 3] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredVaryLj$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 4] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 4] = mean(apply(ylocPred, 1, var))
spatPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     2.511976        2.478452         2.658194         2.476308
## postMSE        14.123149       14.003086        14.341286        12.764447
## postVar        11.613496       11.526939        11.685429        10.290197
```

In light of the magnitudes of the actual $y_t(\boldsymbol{s}_i)$'s at the 9 testing locations and time $t = 1, 2, \ldots, 300$, all spatial prediction metrics are appropriately small. `NNGPsequenVaryLj` appears to be the best, followed by `NNGPblockFixedL`, `fullGPfixedL`, and finally `NNGPsequenFixedL`.

We present below results from some other spatial prediction instances from the same model fitting objects we have obtained.

```
setwd("fullGPfixedL")
load("s27toyExFullGPspatpredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s27toyExNNGPspatpredFixedLblock.RData")
setwd("../NNGPsequenFixedL")
load("s27toyExNNGPspatpredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s27toyExNNGPspatpredVaryLj.RData")
setwd("..")
spatPredKrigTime <- matrix(0, 2, 4,
                           dimnames = list(c("alpha", "weightsXiLambda"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
spatPredKrigTime[1, 1] = spatpredFixedL$alphaKrigTime
spatPredKrigTime[2, 1] = spatpredFixedL$weightsXiLambdaKrigTime
spatPredKrigTime[1, 2] = spatpredFixedLblock$alphaKrigTime
spatPredKrigTime[2, 2] = spatpredFixedLblock$weightsXiLambdaKrigTime
spatPredKrigTime[1, 3] = spatpredFixedLsequen$alphaKrigTime
spatPredKrigTime[2, 3] = spatpredFixedLsequen$weightsXiLambdaKrigTime
spatPredKrigTime[1, 4] = spatpredVaryLj$alphaKrigTime
spatPredKrigTime[2, 4] = spatpredVaryLj$weightsXiLambdaKrigTime
spatPredKrigTime
```

```
##                 fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## alpha                  64165            2893             2889             2461
## weightsXiLambda          176             196              207              117
```

```
spatPredMetricMat <- matrix(0, 3, 4,
                           dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
testingLocNum <- M - m
ylocPredList = spatpredFixedL$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 1] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 1] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLblock$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 2] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 2] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLsequen$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
```

```
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 3] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 3] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredVaryLj$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 4] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 4] = mean(apply(ylocPred, 1, var))
spatPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     2.563105        2.553847         2.731399         2.530977
## postMSE        17.541539       17.490599        17.798567        12.216428
## postVar        14.981430       14.939740        15.070183         9.687389
```

```
setwd("fullGPfixedL")
load("s31toyExFullGPspatpredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s31toyExNNGPspatpredFixedLblock.RData")
setwd("../NNGPsequenFixedL")
load("s31toyExNNGPspatpredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s31toyExNNGPspatpredVaryLj.RData")
setwd("..")
spatPredKrigTime <- matrix(0, 2, 4,
                           dimnames = list(c("alpha", "weightsXiLambda"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
spatPredKrigTime[1, 1] = spatpredFixedL$alphaKrigTime
spatPredKrigTime[2, 1] = spatpredFixedL$weightsXiLambdaKrigTime
spatPredKrigTime[1, 2] = spatpredFixedLblock$alphaKrigTime
spatPredKrigTime[2, 2] = spatpredFixedLblock$weightsXiLambdaKrigTime
spatPredKrigTime[1, 3] = spatpredFixedLsequen$alphaKrigTime
spatPredKrigTime[2, 3] = spatpredFixedLsequen$weightsXiLambdaKrigTime
spatPredKrigTime[1, 4] = spatpredVaryLj$alphaKrigTime
spatPredKrigTime[2, 4] = spatpredVaryLj$weightsXiLambdaKrigTime
spatPredKrigTime
```

```
##                 fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## alpha                  64159            2920             2880             2472
## weightsXiLambda          125             137              145               68
```

```
spatPredMetricMat <- matrix(0, 3, 4,
                           dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
testingLocNum <- M - m
ylocPredList = spatpredFixedL$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
```

```
#(testingLocNum * trainingT * 0) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * 0 * trainingT (rowMeans) note that 0 = 1 here
spatPredMetricMat[1, 1] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 1] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLblock$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * 0))
#(testingLocNum * trainingT * 0) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * 0 * trainingT (rowMeans) note that 0 = 1 here
spatPredMetricMat[1, 2] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 2] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLsequen$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * 0))
#(testingLocNum * trainingT * 0) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * 0 * trainingT (rowMeans) note that 0 = 1 here
spatPredMetricMat[1, 3] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 3] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredVaryLj$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * 0))
#(testingLocNum * trainingT * 0) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * 0 * trainingT (rowMeans) note that 0 = 1 here
spatPredMetricMat[1, 4] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 4] = mean(apply(ylocPred, 1, var))
spatPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     2.455618        2.505853         2.663366         2.516074
## postMSE        15.698755       15.778148        16.027189        19.132043
## postVar        13.245786       13.274950        13.366496        16.619293
```

```
setwd("fullGPfixedL")
load("s19toyExFullGPspatpredFixedL.RData")
setwd("../NNGPblockFixedL")
load("s19toyExNNGPspatpredFixedLblock.RData")
setwd("../NNGPsequenFixedL")
load("s19toyExNNGPspatpredFixedLsequen.RData")
setwd("../NNGPsequenVaryLj")
load("s19toyExNNGPspatpredVaryLj.RData")
setwd("..")
spatPredKrigTime <- matrix(0, 2, 4,
                           dimnames = list(c("alpha", "weightsXiLambda"),
                                           c("fullGPfixedL", "NNGPblockFixedL",
                                             "NNGPsequenFixedL", "NNGPsequenVaryLj")))
```

```
spatPredKrigTime[1, 1] = spatpredFixedL$alphaKrigTime
spatPredKrigTime[2, 1] = spatpredFixedL$weightsXiLambdaKrigTime
spatPredKrigTime[1, 2] = spatpredFixedLblock$alphaKrigTime
spatPredKrigTime[2, 2] = spatpredFixedLblock$weightsXiLambdaKrigTime
spatPredKrigTime[1, 3] = spatpredFixedLsequen$alphaKrigTime
spatPredKrigTime[2, 3] = spatpredFixedLsequen$weightsXiLambdaKrigTime
spatPredKrigTime[1, 4] = spatpredVaryLj$alphaKrigTime
spatPredKrigTime[2, 4] = spatpredVaryLj$weightsXiLambdaKrigTime
spatPredKrigTime
```

```
##                 fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## alpha                  63997            2877             2893             2509
## weightsXiLambda          128             139              147               80
```

```
spatPredMetricMat <- matrix(0, 3, 4,
                            dimnames = list(c("postMeanMSE", "postMSE", "postVar"),
                                            c("fullGPfixedL", "NNGPblockFixedL",
                                              "NNGPsequenFixedL", "NNGPsequenVaryLj")))
testingLocNum <- M - m; ylocPredList = spatpredFixedL$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 1] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 1] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 1] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLblock$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 2] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 2] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 2] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredFixedLsequen$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 3] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 3] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 3] = mean(apply(ylocPred, 1, var))
ylocPredList = spatpredVaryLj$Y
ylocPred <- t(matrix(unlist(ylocPredList), ncol = testingLocNum * trainingT * O))
#(testingLocNum * trainingT * O) * NKeep
ylocPredMean <- apply(ylocPred, 1, mean)
# of length N = testingLocNum * O * trainingT (rowMeans) note that O = 1 here
spatPredMetricMat[1, 4] <- mean((ylocPredMean - YtestingSpat)^2)
diffMat <- sweep(ylocPred, 1, YtestingSpat, "-")
spatPredMetricMat[2, 4] = mean(rowMeans(diffMat^2))
spatPredMetricMat[3, 4] = mean(apply(ylocPred, 1, var))
```

```
rm(spatpredFixedL, spatpredFixedLblock, spatpredFixedLsequen, spatpredVaryLj)
spatPredMetricMat
```

```
##             fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## postMeanMSE     2.489561        2.460305         2.630899         2.528279
## postMSE        33.001335       32.971822        33.190347        20.482859
## postVar        30.517878       30.517621        30.565560        17.958172
```

Spatial Clustering of Temporal Trends for all 352 Training Spatial Locations

```
calcRandIndex <- function(predictedCluster, actualGroup, numObs){
  numerator <- denominator <- numObs * (numObs-1) / 2
  for (i in 1:(numObs-1)){
    for (j in (i + 1):numObs){
      if ((predictedCluster[i]==predictedCluster[j])+(actualGroup[i]==actualGroup[j]) == 1)
        numerator = numerator - 1
    }
  }
  randIndex <- numerator / denominator
  return(randIndex)
}
calcAccuRatio <- function(predictedCluster, actualGroup, numObs){
  actualGroupsPoss1 = factor(actualGroup, labels=c(1,2))
  actualGroupsPoss2 = factor(actualGroup, labels=c(2,1))
  accuRatio <- max(sum(predictedCluster==actualGroupsPoss1),
                   sum(predictedCluster==actualGroupsPoss2))/numObs
  return(accuRatio)
}
fittedClusGpMat <- cbind(fittedClusGpMat.fullGPfixedL, fittedClusGpMat.NNGPblockFixedL,
                         fittedClusGpMat.NNGPsequenFixedL,
                         fittedClusGpMat.NNGPsequenVaryLj)
modelVec <- c("fullGPfixedL", "NNGPblockFixedL", "NNGPsequenFixedL", "NNGPsequenVaryLj")
clusIterWeightsVec <- colnames(fittedClusGpMat.fullGPfixedL)
# paste(c(10, 100, 1000), "iterWeights", sep = " ")
accuRatioMat <- matrix(apply(fittedClusGpMat, 2, calcAccuRatio,
                             actualGroup = spatGroupOverallTraining,
                             numObs = m), 3, 4,
                       dimnames = list(clusIterWeightsVec, modelVec))
accuRatioMat
```

```
##                  fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## 10 iterWeights              1               1                1                1
## 100 iterWeights             1               1                1                1
## 1000 iterWeights            1               1                1                1
```

```
randIndMat <- matrix(apply(fittedClusGpMat, 2, calcRandIndex,
                           actualGroup = spatGroupOverallTraining,
                           numObs = m), 3, 4,
                     dimnames = list(clusIterWeightsVec, modelVec))
randIndMat
```

```
##                  fullGPfixedL NNGPblockFixedL NNGPsequenFixedL NNGPsequenVaryLj
## 10 iterWeights              1               1                1                1
## 100 iterWeights             1               1                1                1
## 1000 iterWeights            1               1                1                1
```

The above results suggest that all our 4 methods `fullGPfixedL`, `NNGPblockFixedL`, `NNGPsequenFixedL`, and `NNGPsequenVaryLj` lead to completely correct spatial clustering outcomes depicted below.

```
xcoordTraining <- xcoord[-whichTesting]; ycoordTraining = ycoord[-whichTesting]
spatTrainingGpDF <- data.frame(x = xcoordTraining, y = ycoordTraining,
                      actualTrainingSpatGp = as.factor(spatGroupOverallTraining))
actualTrainingSpatPlotBW <- ggplot(spatTrainingGpDF) +
  geom_point(aes(x = x, y = y, col = actualTrainingSpatGp),
             size = 5, show.legend = FALSE) +
  scale_color_manual(values = c("black", "white")) + labs(x = "", y = "") +
  ggtitle("2 Actual Spatial Groups for the 352 Training Locations") +
  coord_fixed(ratio = 1) +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
        panel.grid.major = element_line(color="grey"),
        panel.grid.minor = element_line(color="grey"))
actualTrainingSpatPlotBW
```