Spatial Partitioning

Mariella Carbajal Carrasco, Laura Goodman, &
Andrew Hutchens

ST533 - Midterm 2

Motivation and intuition

- With spatial partitioning (SP), the spatial domain Δ is partitioned into D subregions $\Delta_1, \ldots, \Delta_D$.
- To get a sparse $\mathbf{\Sigma}$, SP assumes independence between observations across regions, so that a block-diagonal structure for $\mathbf{\Sigma}$ is obtained as follows:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{\Sigma}_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \mathbf{\Sigma}_D \end{bmatrix}$$

where $\underset{n_d \times K}{\mathbf{\Sigma}_d}$ and $\sum_{d=1}^{D} n_d = N$.
- En route to obtaining a sparse covariance matrix $\mathbf{\Sigma}$, SP uses fixed rank Kriging (FRK) to reduce the dimensions of matrix inverses from $N \times N$ to $K \times K$ for $K < N$.

Motivation and intuition cont'd.

- Partitioning into *equal areas* creates partitions of equal size.
- Partitioning by *hierarchical clustering on spatial gradients*: cluster/partition boundaries follow lines of high change in the observations across space Δ, i.e. the data inform the partitions (Heaton et al., 2017).
- Partitioning by *centroid clustering*: data is partitioned by defining a set of centers $\mathbf{c} = [c_1, \ldots, c_M]$ that divide Δ into M distinct regions $R_1, \ldots, R_M$; all of the points $\mathbf{x}_i \in R_i$ are closer to c_i than to any other $c_j \in \Delta$, $i \neq j$ (Kim et al., 2005).
- *Treed-regression partitioning* works by recursively making binary splits of the data according to a specified rule (Konomi et al., 2014). *Mixture model partitioning* works by specifying the response variable and its associated model as region- and individual-specific (Neelon et al., 2014).

Visual examples

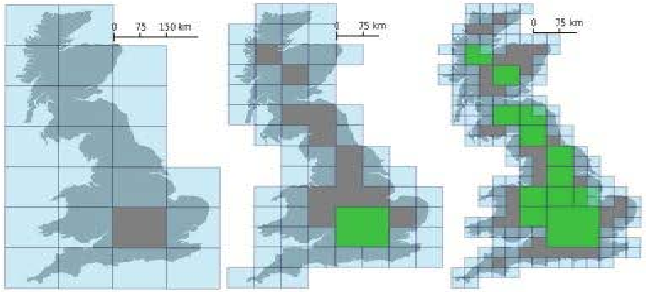


Figure: Treed partitioning (Source: Krauss, Wikipedia)

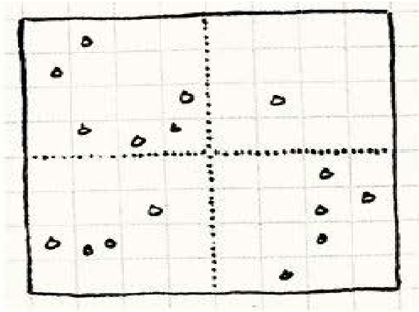


Figure: Equal area partitioning (Source: Robert Nystrom, gameprogrammingpatterns.com)

The nitty gritty

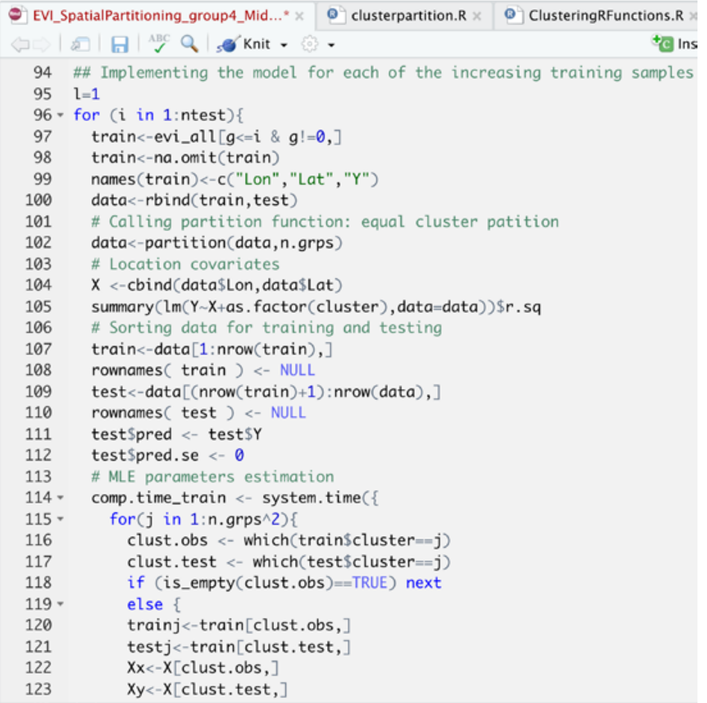
- Base FRK model:
$$\widetilde{Y}(s_i) = \mu(s_i) + w(s_i) + \xi(s_i) + \varepsilon(s_i),$$
where $\mu(s) = X(s_i)'\beta$ and $\varepsilon(s_i) \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is an i.i.d. measurement error.
- Assume K basis functions $\mathbf{h}(s_i) = [h_1(s_i), \ldots, h_K(s_i)]'$ with coefficients $\mathbf{w}^\star = [w_1^\star, \ldots, w_K^\star]$ such that
$$w(s) \approx \widetilde{w}(s_i) = \textstyle\sum_{k=1}^{K} h_k(s_i) w_k^\star, \; \forall s_i \in \Delta.$$
- SP splits the resulting FRK model
$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{H}\mathbf{w}^\star + \boldsymbol{\xi} + \varepsilon$$
across the D subregions so that each region is modeled by
$$\mathbf{Y}_d = \mathbf{X}_d\boldsymbol{\beta} + \mathbf{H}_d\mathbf{w}^\star + \boldsymbol{\xi}_d + \varepsilon_d,$$
where $\underset{n_d \times K}{\mathbf{H}_d}$ contains the spatial basis functions from FRK.

The nitty gritty cont'd.

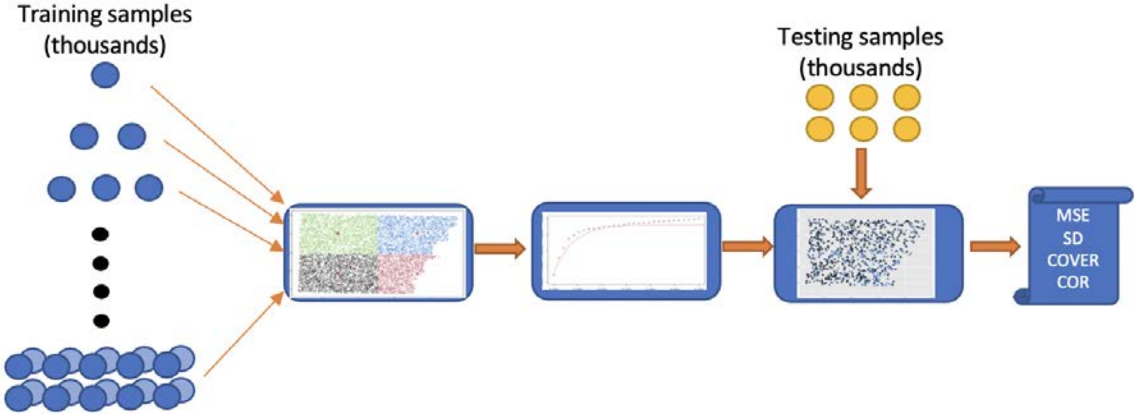
- SP allows β and $\mathbf{w}^\star$ to be the same across all subregions $\Delta_d \in \Delta$, which permits smoothing of estimates across subregions.
- The FRK model used for SP assumes $\mathbf{w}^\star \sim \mathcal{N}(0, \mathbf{\Sigma}_{\mathbf{w}^\star}(\boldsymbol{\theta}))$, $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma_\xi^2 \mathbf{W})$, and $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, where $\mathbf{\Sigma}_{\mathbf{w}^\star}(\boldsymbol{\theta})$ is $\mathbf{w}^\star$'s covariance matrix, $\mathbf{W}$ is a fixed diagonal weight matrix, and $\boldsymbol{\theta}$ consists of parameters $\boldsymbol{\sigma}^2$ and ϕ to be estimated.
- SP captures FRK's advantage of having $K \times K$ matrices to invert $(K < N)$ while further speeding up computation by allowing the inverse of each region's covariance matrix $\mathbf{\Sigma}_d^{-1}$ to be computed individually to find $\mathbf{\Sigma}^{-1}$.

Implementation details

- R Packages: geoR, rdist, tidyverse, parallel
- GitHub repository: https://github.ncsu.edu/mcarbaj/SpSt_Midterm2
- Tuning parameters:
  Type of clustering partition: Equal area (+ centroid)
  Size of partitions: 4 (9,25)

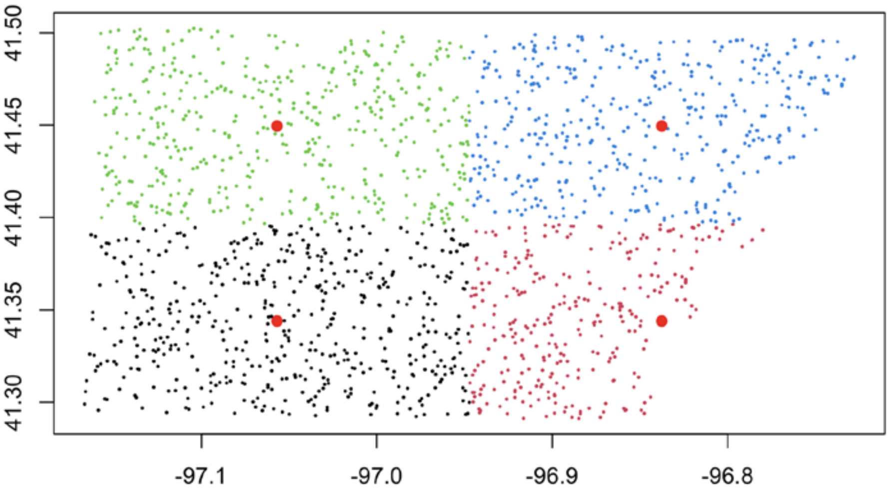



```r
## Implementing the model for each of the increasing training samples
l=1
for (i in 1:ntest){
  train<-evi_all[g<=i & g!=0,]
  train<-na.omit(train)
  names(train)<-c("Lon","Lat","Y")
  data<-rbind(train,test)
  # Calling partition function: equal cluster patition
  data<-partition(data,n.grps)
  # Location covariates
  X <-cbind(data$Lon,data$Lat)
  summary(lm(Y~X+as.factor(cluster),data=data))$r.sq
  # Sorting data for training and testing
  train<-data[1:nrow(train),]
  rownames( train ) <- NULL
  test<-data[(nrow(train)+1):nrow(data),]
  rownames( test ) <- NULL
  test$pred <- test$Y
  test$pred.se <- 0
  # MLE parameters estimation
  comp.time_train <- system.time({
    for(j in 1:n.grps^2){
      clust.obs <- which(train$cluster==j)
      clust.test <- which(test$cluster==j)
      if (is_empty(clust.obs)==TRUE) next
      else {
      trainj<-train[clust.obs,]
      testj<-train[clust.test,]
      Xx<-X[clust.obs,]
      Xy<-X[clust.test,]
```

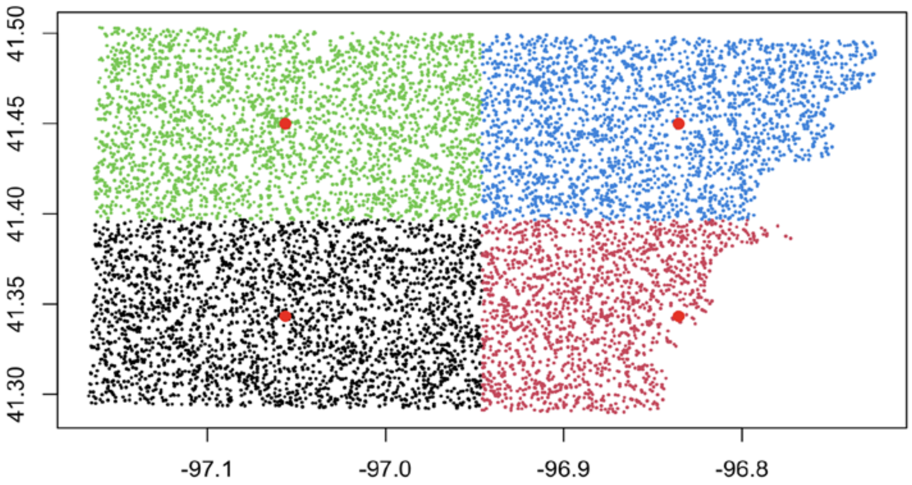


Spatial Partitioning: Equal area

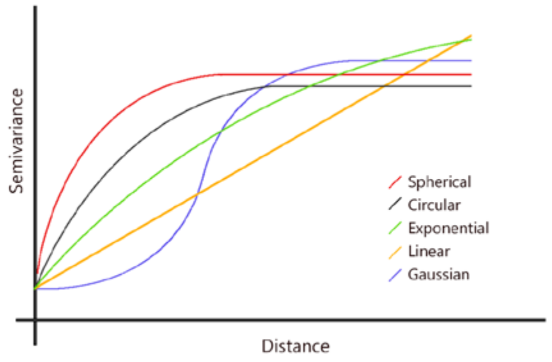
Train set: 1,000

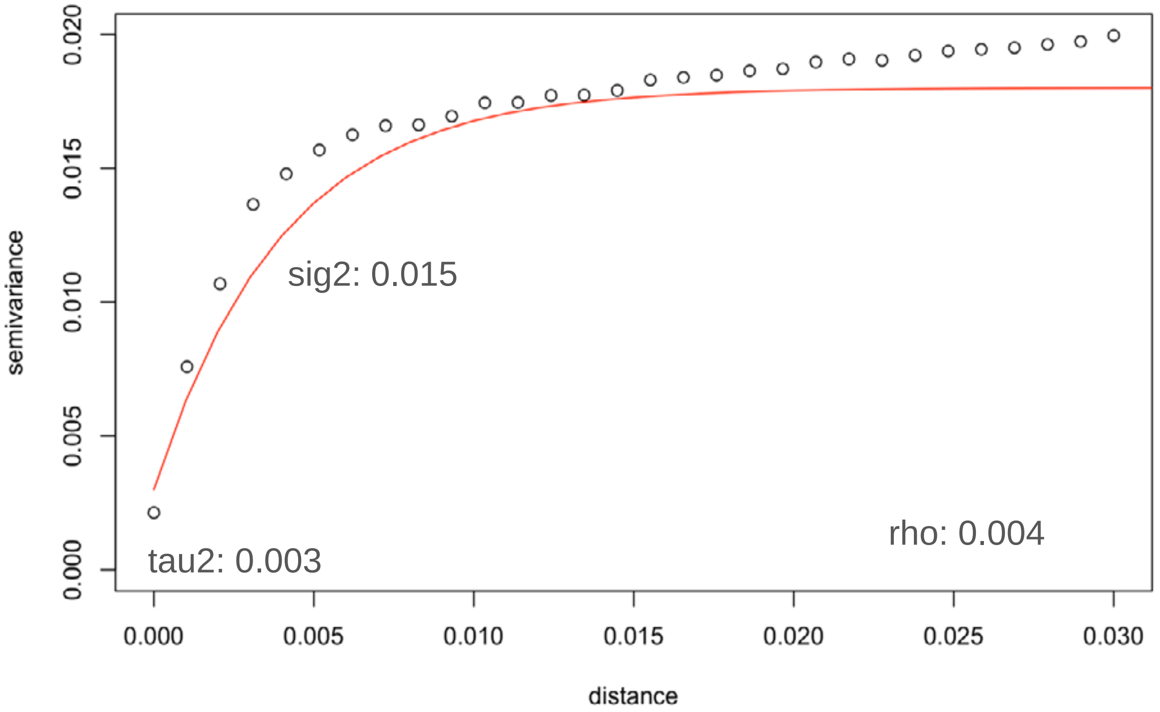


Train set:



Semivariogram fitting parameters

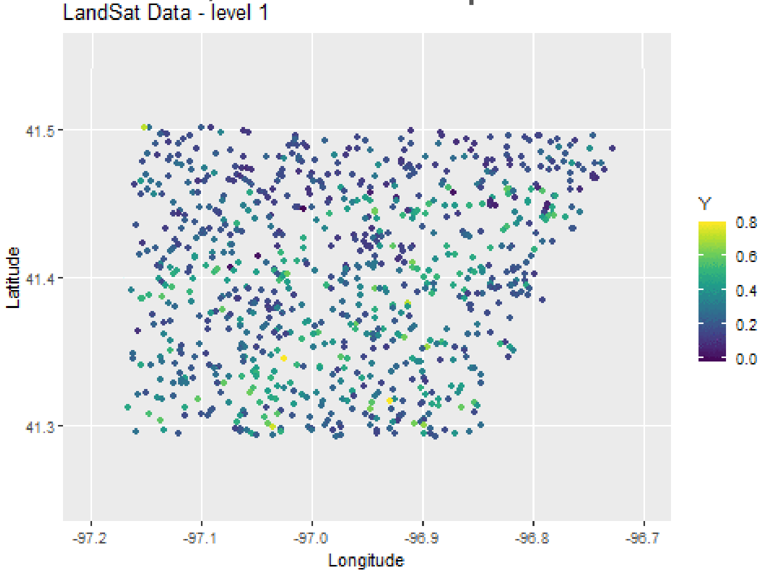


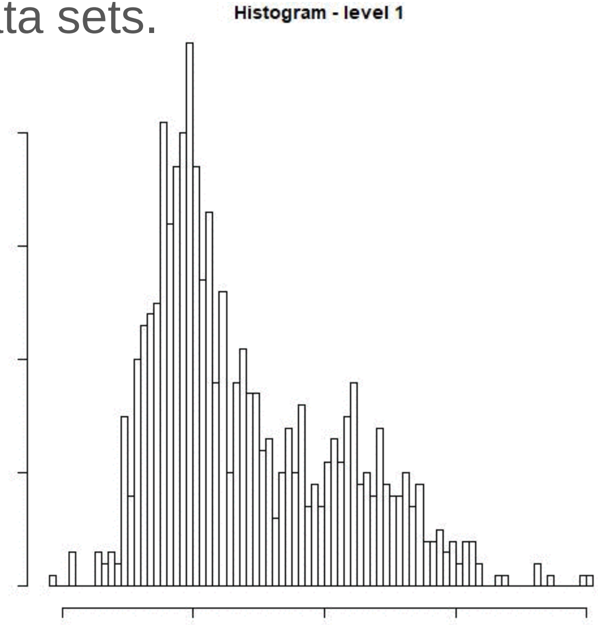


Applying the Method

With each additional training the amount of data processed increases.

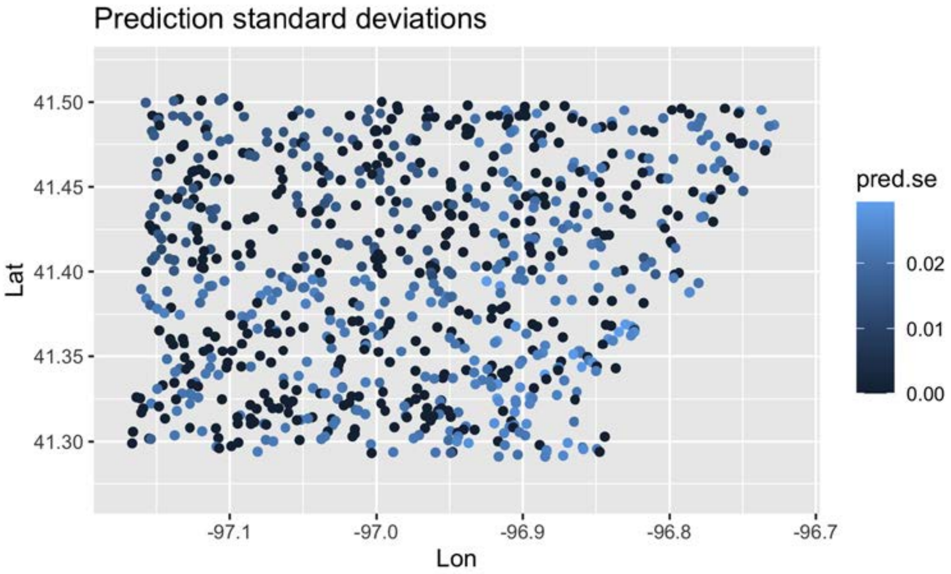
We compare with Maximum Likelihood Estimation since that typically provides good results, but is not optimized for large data sets.

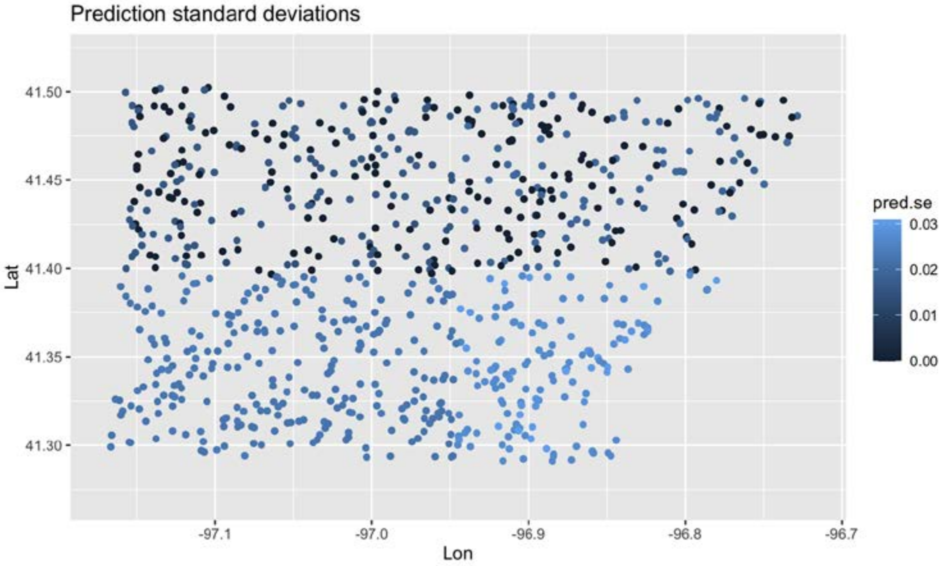




Results of Spatial Partitioning

Train set: 1,000 Train set: 10,000

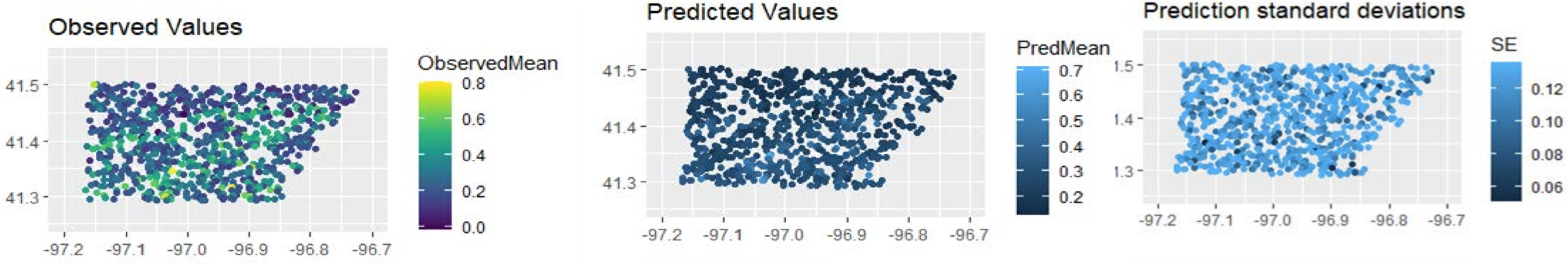




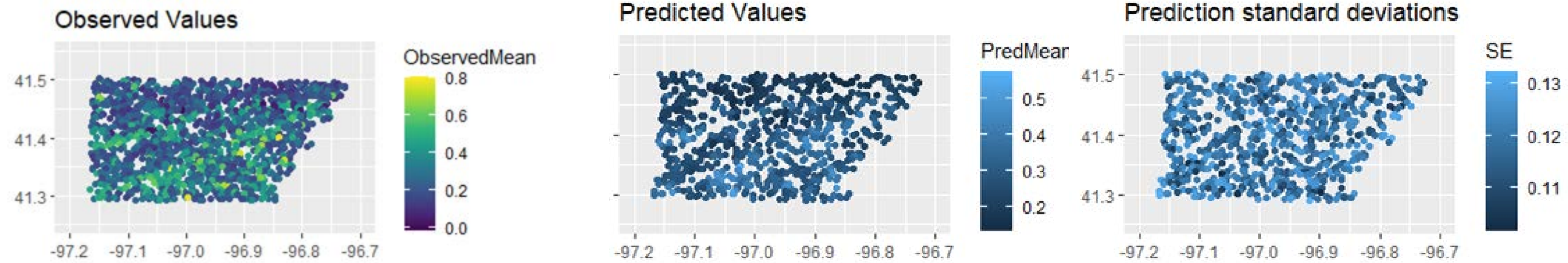
Results of Maximum Likelihood Estimation

Marginal benefits to adding more data, but significant run time increases

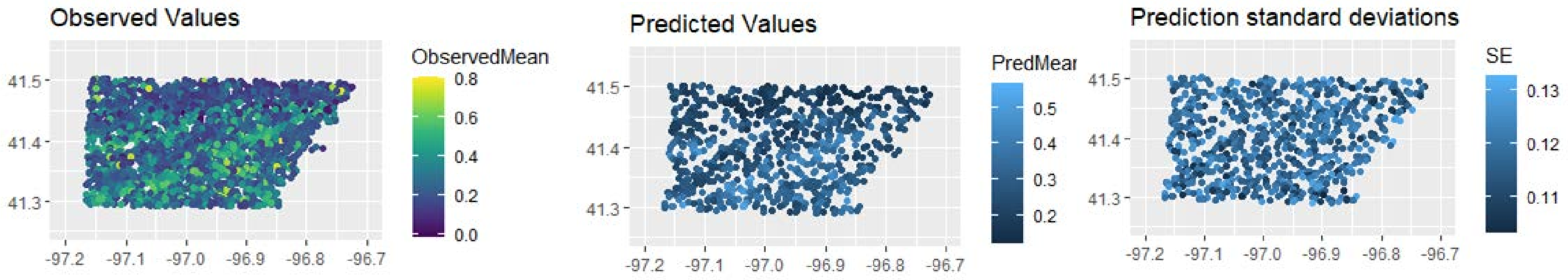
1000 Point Training Set
MLE - 164 s



2000 Point Training Set
MLE - 798 s

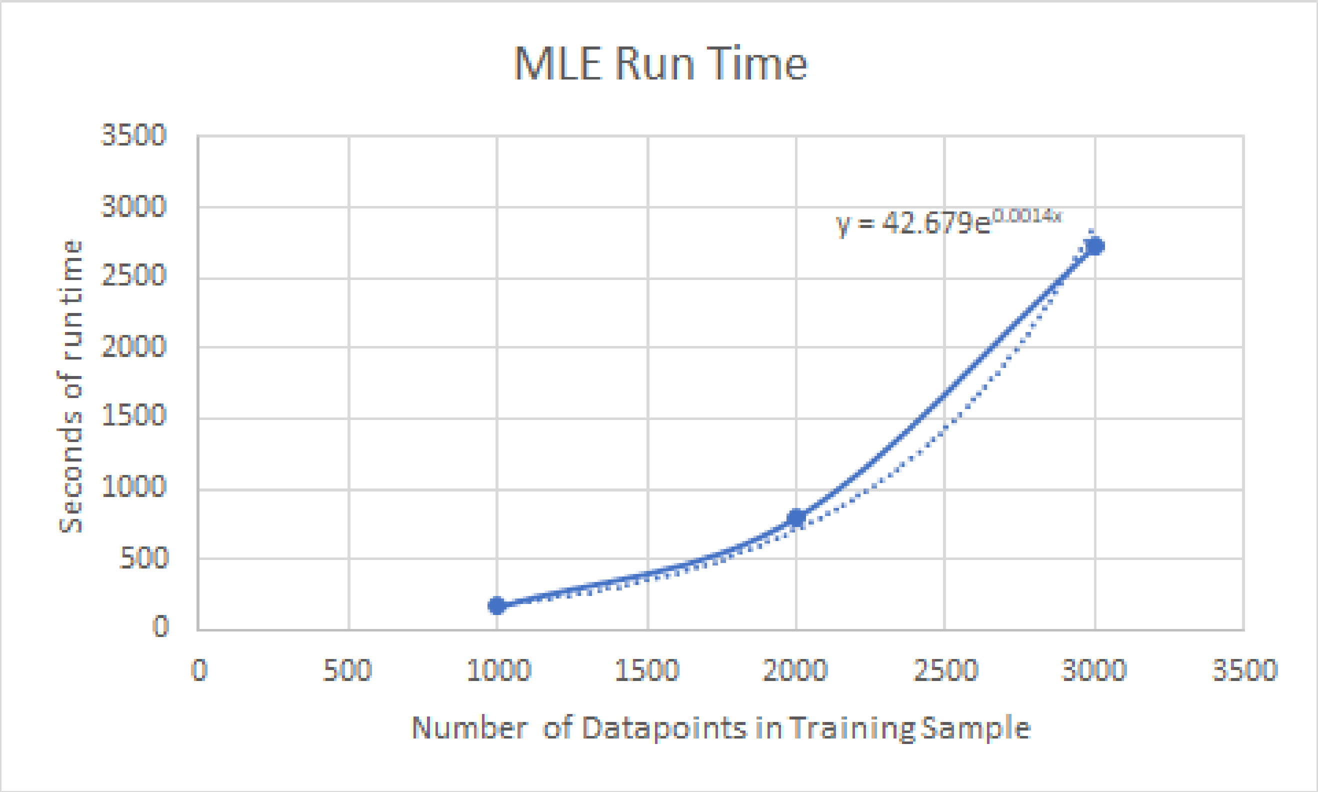


3000 Point Training Set
MLE - 2725 s

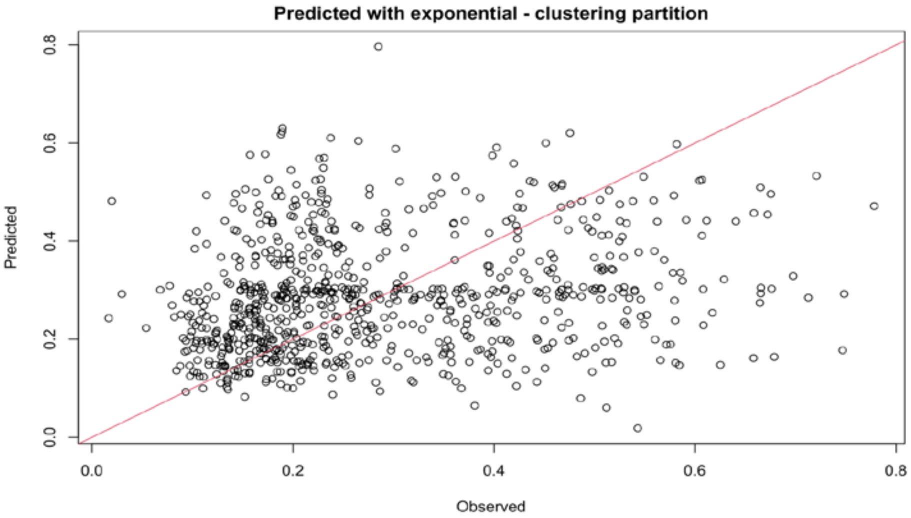


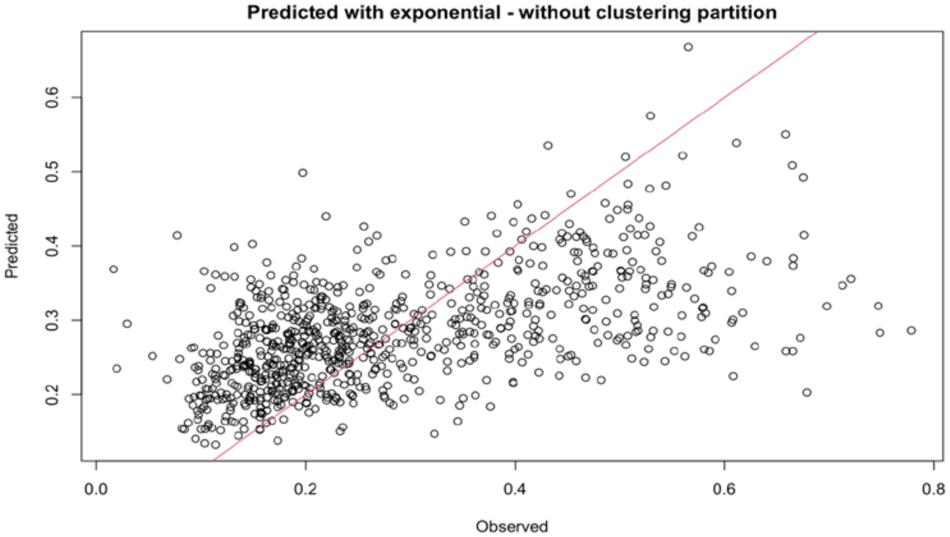
4000+ Points Training Set
MLE ran for well over an hour each time, so cut program short

MLE not feasible for large numbers of datapoints



Comparison: MLE vs spatial partition

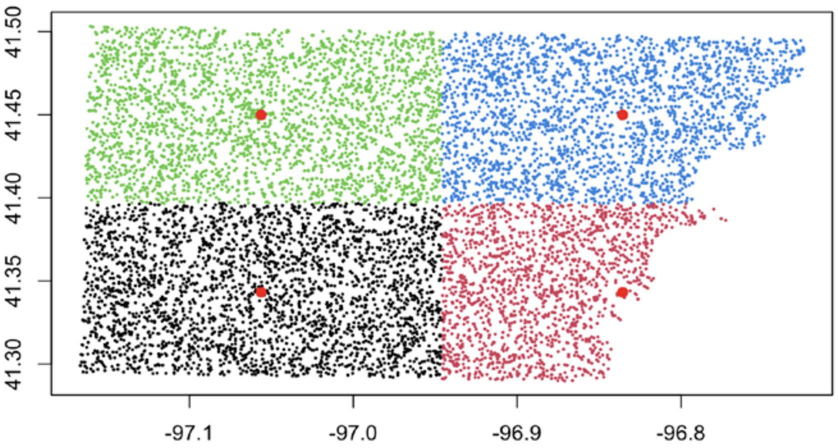




Results for training set = 1,000

	MLE	Spatial partition
MSE	0.0155	0.0284
COR	0.546	0.213
Time	0.898 s	11.1 s

Conclusions



- From our analysis, with fewer data points, Maximum Likelihood Estimation (MLE) *outperforms* Spatial Partitioning with Equal Area Partitioning.
- As the points considered increases, MLE is no longer feasible as computer memory constraints become an issue as well as run time durations. Thus, partitioning the data allows for calculations to be made.
- Equal Area Partitioning did not appear to improve the predictions as there were small areas with little variability and the partitioning removed information from the system.