Spatial Partitioning

Group3
Nate Wiecha, Hongjian Yang, Shiyue Yao

Outline:

- **Description**
  - Motivation
  - Types of partition strategies
  - Model specification
  - Spatial Partitioning v.s. Divide-and-conquer
- **Implementation**
  - Implementation processes
  - Implementation details
- **Results**

Motivation:

- Big data strategies

(i) low rank

(ii) **sparse covariance matrices**: by introducing 0's into $\boldsymbol{\Sigma}$

(iii) sparse precision matrices and

(iv) algorithmic

Spatial partitioning

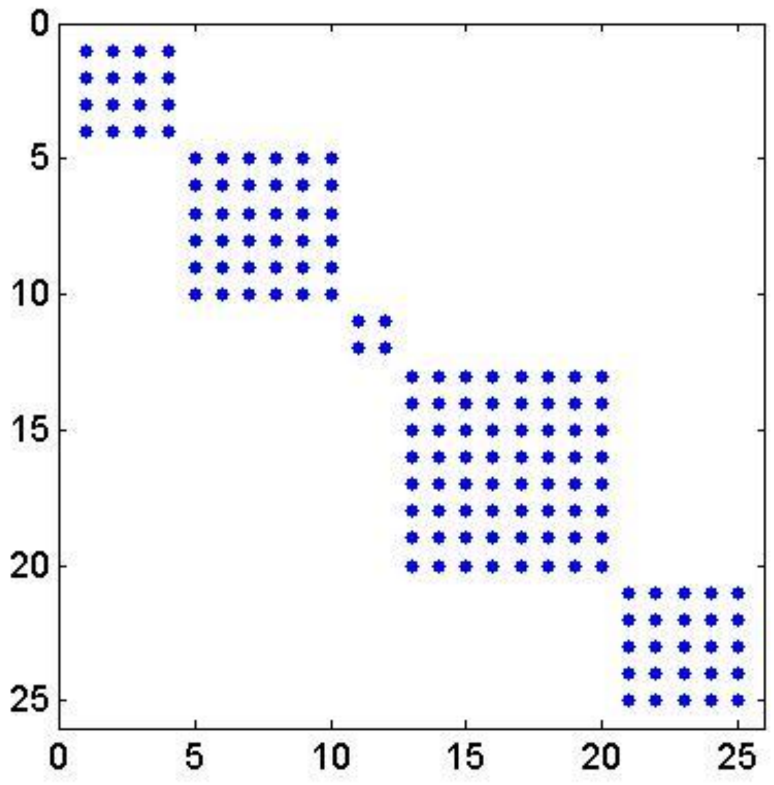
- Spatial partitioning:

  1) split the spatial domain into subregions

  2) assume independence across subregions

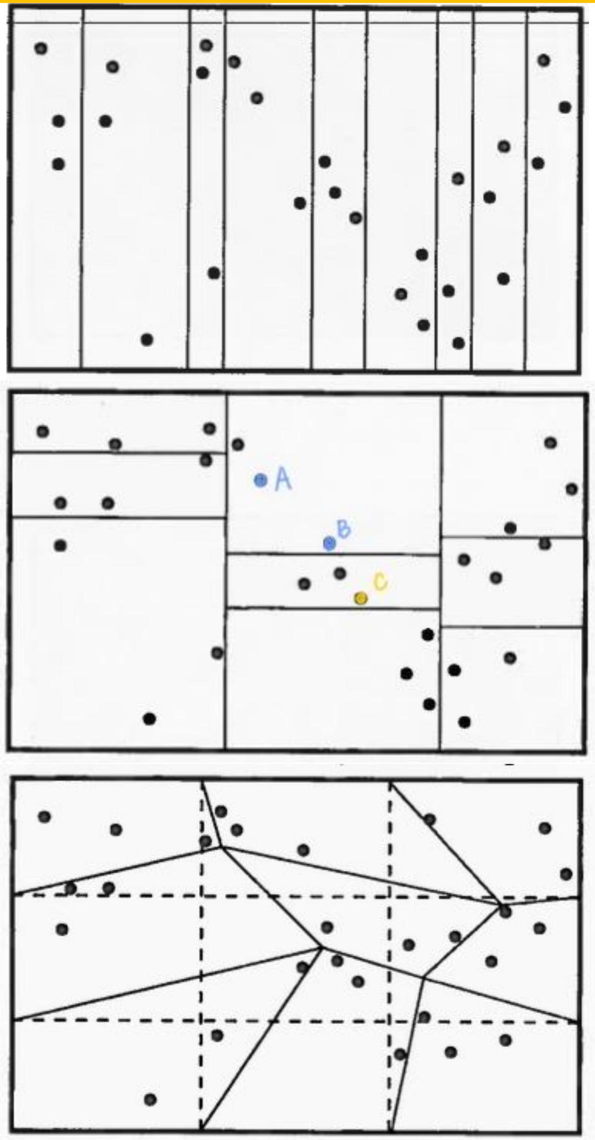
  3) compute likelihood simultaneously

- Advantage: sparse matrix computation and parallel programming



Types of partition strategies

- **Priori methods:**
  - Equal area partition
  - Partitioning based on centroid clustering
  - hierarchical clustering based on spatial gradients
- **Model based methods:**
  - Treed regression
  - Mixture Modeling



Source: Ding, Yuemin & Densham, Paul. (1996). Spatial strategies for parallel spatia modeling. International Journal of Geographical Information Science. 10. 669-698.

Model specification

Basic settings:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{H\omega}^* + \boldsymbol{\xi} + \boldsymbol{\varepsilon}$$

- where $\boldsymbol{X}$ is the design matrix; $\boldsymbol{\beta}$ are the regression coefficients;
- $\boldsymbol{H}$ is the N × K matrix of spatial basis functions with associated random coefficients $\boldsymbol{\omega}^* \sim \boldsymbol{N}\left(\mathbf{0},\ \boldsymbol{\Sigma}_{\boldsymbol{\omega}^*}(\boldsymbol{\theta})\right)$;
- $\boldsymbol{\xi} \sim \boldsymbol{N}(\mathbf{0},\ \boldsymbol{\sigma_\xi}^2)$; and $\boldsymbol{\varepsilon} \sim \boldsymbol{N}(\mathbf{0}, \boldsymbol{\sigma_\varepsilon}^2\boldsymbol{i})$

Model specification

Spatial partitioning settings:

Let the spatial domain $\mathcal{D} = \cup_{d=1}^{D} \mathcal{D}_d$ where $\mathcal{D}_1, \ldots, \mathcal{D}_D$ are subregions that form a partition.

for each = subregion $Y_d\{Y(s_i): \quad s_i \in \mathcal{D}_d\}, d = 1,2, \ldots, D$:

$$\boldsymbol{Y_d} = \boldsymbol{X_d \beta} + \boldsymbol{H_d \omega^*} + \boldsymbol{\xi_d} + \boldsymbol{\varepsilon_d}$$

- where $\boldsymbol{X_d}$ is a design matrix containing covariates associated with $\boldsymbol{Y_d}$,
- $\boldsymbol{H_d}$ is a matrix of spatial basis functions
- $\boldsymbol{\xi_d}$ and $\boldsymbol{\varepsilon_d}$ are the sub-vectors of $\boldsymbol{\xi}$ and $\boldsymbol{\varepsilon}$ corresponding to region $\mathcal{D}_d$.
- each subregion shares common $\boldsymbol{\beta}$ and $\boldsymbol{\omega}^*$ parameters

Spatial Partitioning v.s. Divide-and-Conquer

- they are both strategies for **parallel programming**
- **Divide and conquer:** the full dataset is subsampled, the model is fit to each subset and the results across subsamples are pooled.
- **Spatial partition:** uses all the data simultaneously in obtaining estimates, but the independence across regions facilitates computation.

Implementation

- Implementation processes
- Implementation details

Implementation process

—— Spatial partitioning

- Inherit functions and some codes from the author
  - For example, basis function creation and MLE functions
- Use nested for loops to control subsets and subregions
  - Most complicated part during implementation
- Use equal area method to partition regions
- Make predictions by clusters

Implementation process

—— Spatial partitioning(cont.)

- All codes run on High Performance Computing Cluster in the statistics department
- Packages:
  - LatticeKrig
  - parallel

Implementation process

—— Standard MLE/Kriging

- First plot the variogram to estimate effective range, spatial variances and nugget
- Then apply MLE/Kriging from geoR package
- Run on High Performance Computing Cluster in the statistics department
- Packages:
    - geoR

Implementation details

—— Spatial partitioning

- Two tuning parameters:
  - Number of subregions and number of cores
- Number of subregions
  - More subregions, faster computation, less accuracy
  - We tested 9, 12, and 25 subregions with 30 cores
- Number of cores
  - No effect on accuracy; More cores, faster computation
  - We test 2, 4, 9 cores with 9 subregions for demonstration
  - For 2 cores, we limit subsets to 12

Selecting parameters

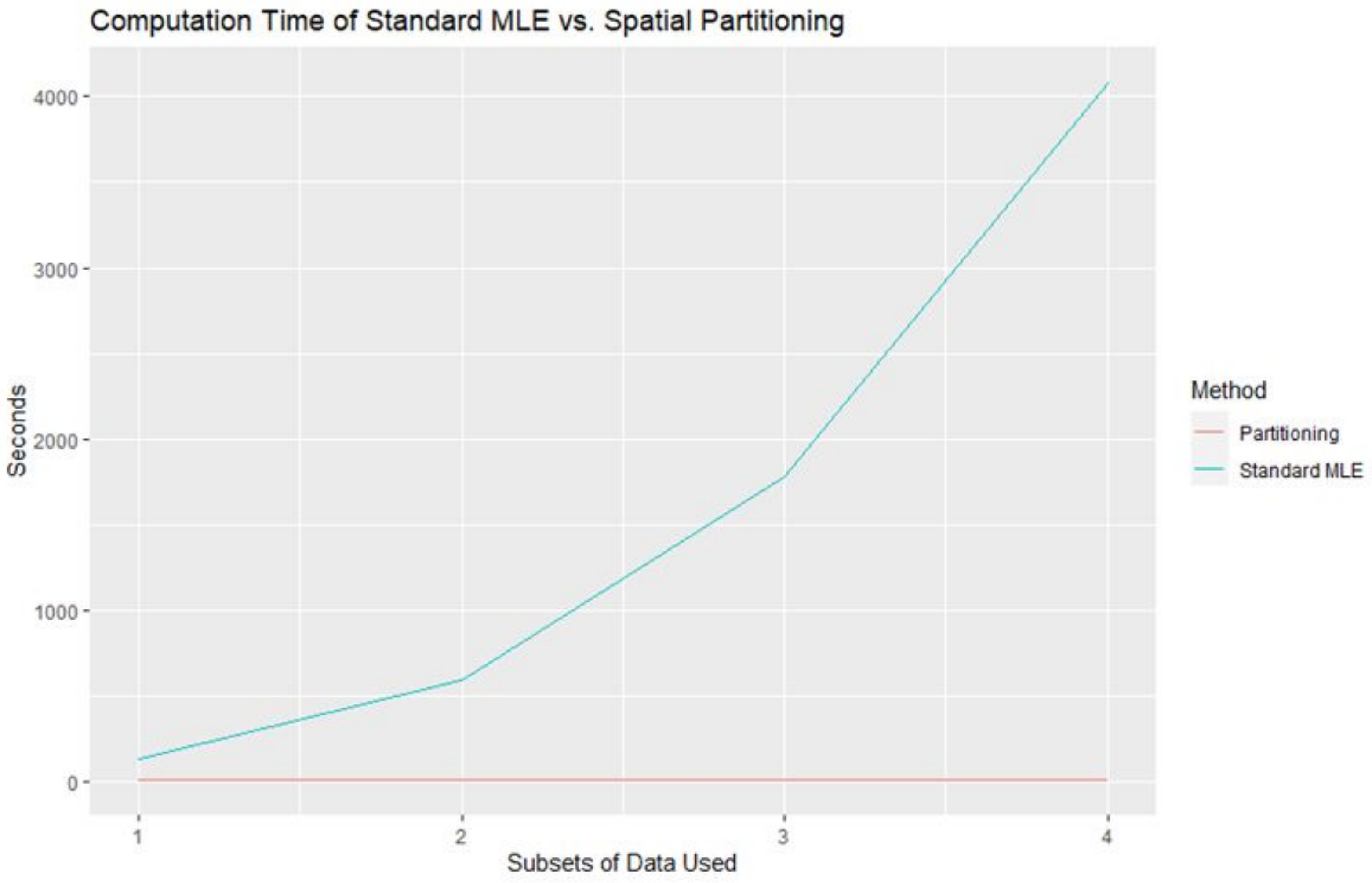
- If possible, use as many cores as possible
  - Limited by hardware
  - Sometimes run into cpu error if occupying too many cores
- More subregions can improve computational speed tremendously, with little compromise on accuracy
  - Some regions have very few or no data: need manual adjustment

Implementation details
—— Standard MLE/Kriging

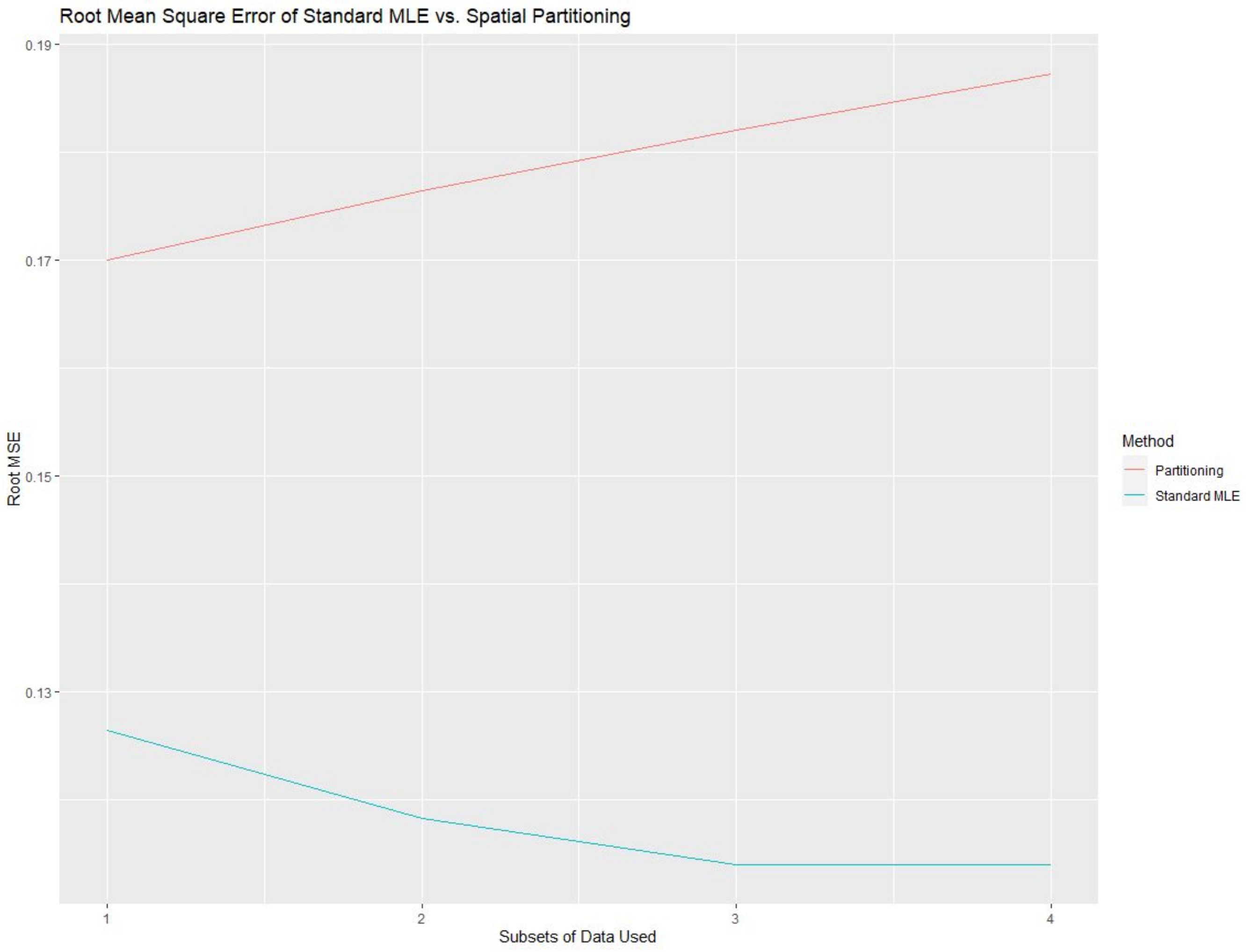
- First-order covariate matrix
  - Second-order covariate matrix always gets a “singular matrix” error message
- Super slow
  - Five subsets take more than 2 hours to compute!

Results

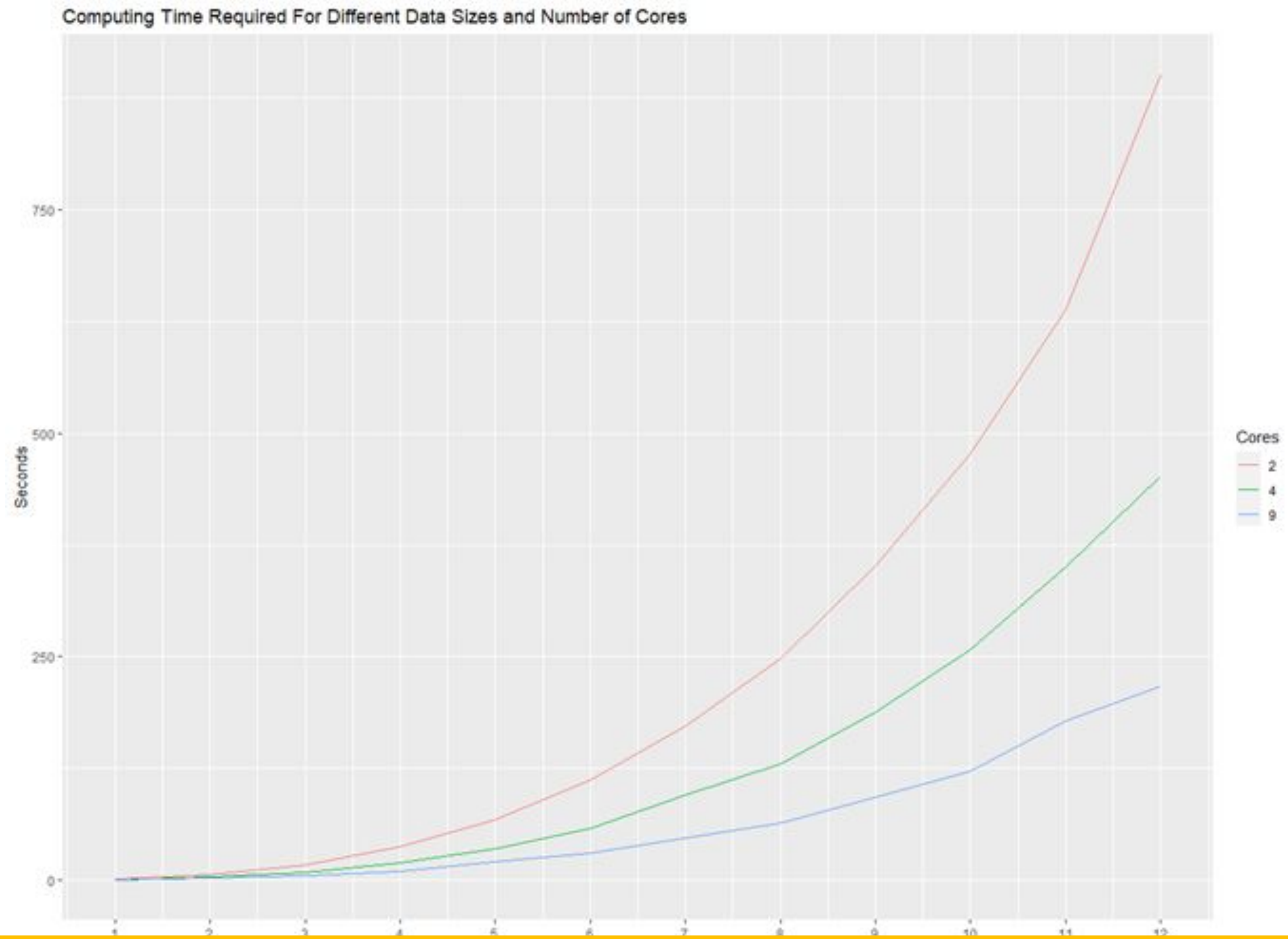
Comparison to Standard MLE: Time



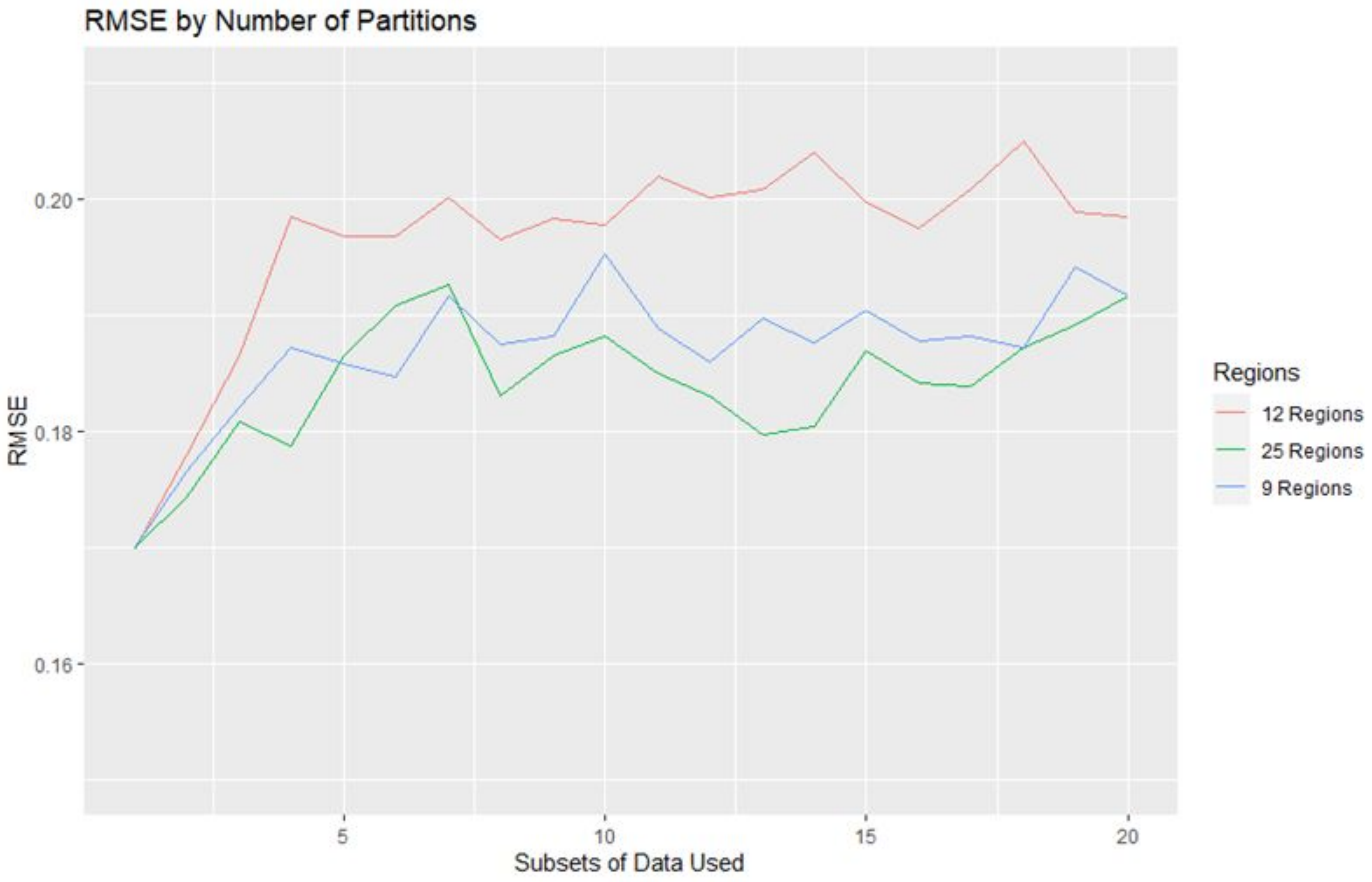
Comparison to MLE: RMSE



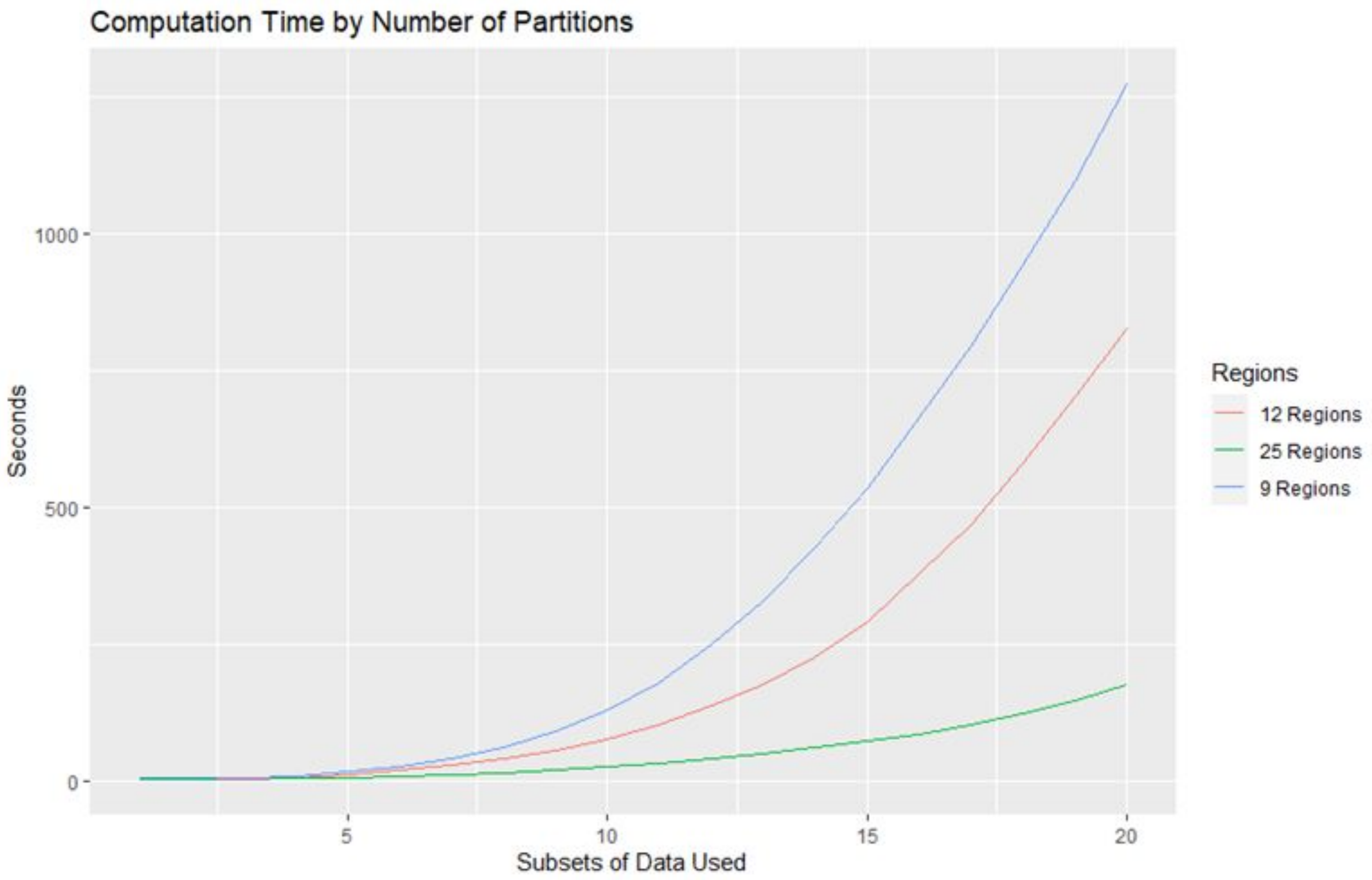
Parallel Computing



Number of Partitions: RMSE



Number of Partitions: Time



Conclusions

- Much, much faster than standard MLE/Kriging due to parallel computing
- Not much less accurate than standard methods
- More regions leads to faster computation, comparable accuracy with this dataset
  - Presumably there is a tradeoff at some point