Supplementary Material for "Geometry-driven Bayesian Inference for Ultrametric Covariance Matrices"

Tsung-Hung Yao[a,*], Zhenke Wu[b], Karthik Bharath[c], and Veerabhadran Baladandayuthapani[b]

[a]Department of Biostatistics, The University of Texas MD Anderson Cancer Center
[b]Department of Biostatistics, University of Michigan
[c]School of Mathematical Sciences, University of Nottingham
[*]corresponding author. E-mail: yaots@umich.edu

December 29, 2023

S1 Additional Results for Simulation Studies

S1.1 Convergence diagnostics

We examine the convergence of our algorithm in Section 6 of Main Paper through the likelihood trace plot and ensure convergence likelihood from different initializations. Specifically, given the same dataset, we run the same algorithm with two chains initiated by two different trees and plot the likelihood over iterations. We randomly chose five likelihood trace plots, each from different sample sizes of $n \in \{30, 50, 100, 250, 500\}$, as shown in Figure S1. Obviously, the likelihood increases rapidly and remains at a relatively high plateau. More importantly, both chains converge to a similar level of likelihood, indicating the convergence of the algorithm.

S1.2 Comparing different algorithms

We compare our algorithm to the algorithm from Nye (2020) and empirically investigate the rate of convergence through the likelihood trace plot shown in Figure S2. The algorithm from Nye

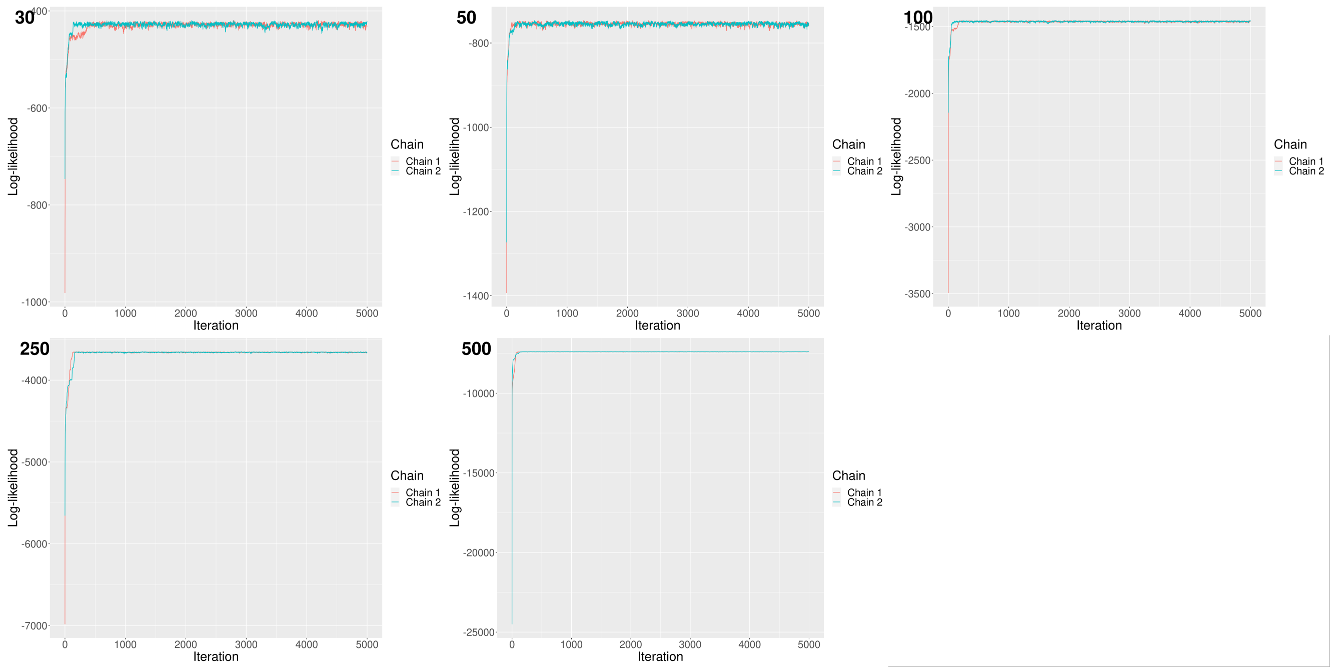

Figure S1: Convergence diagnostics for the algorithm using the likelihood trace plot. Two chains of the same algorithm are initiated by different trees, shown by two colors, across five different sample sizes of $n \in \{30, 50, 100, 250, 500\}$.

(2020) enables the algorithm to propose a candidate edge set that is the same as the edge set from the previous iteration, resulting in slower convergence. We briefly describe Nye's algorithm and refer the reader to the original paper for more details. Essentially, Nye's algorithm is still an MH update and changes the proposal function illustrated in Step 1 to 4 in Algorithm 1. Instead of directly removing the internal split in Step 1, Nye's algorithm generates an edge length difference from a normal distribution of $\delta \sim N(0, \sigma_{\mathcal{E}})$ and lets $c = |e_A| + \delta$. When $c > 0$, the algorithm will stay in the same edge set with $e_B = e_A$ and $|e_B| = c$. Otherwise, when $c \leq 0$, the algorithm will propose a candidate split e_B from nearby orthants and assign the edge length of $|e_B| = -c$. However, Nye's algorithm does not exclude the original split from the candidates, resulting in a positive probability of staying in the same edge set and, therefore, slower convergence. We implement Nye's algorithm and compare it to our algorithm. Figure S2 empirically compares the rate of convergence. Obviously, our algorithm converges faster than Nye's algorithm under five different sample sizes.

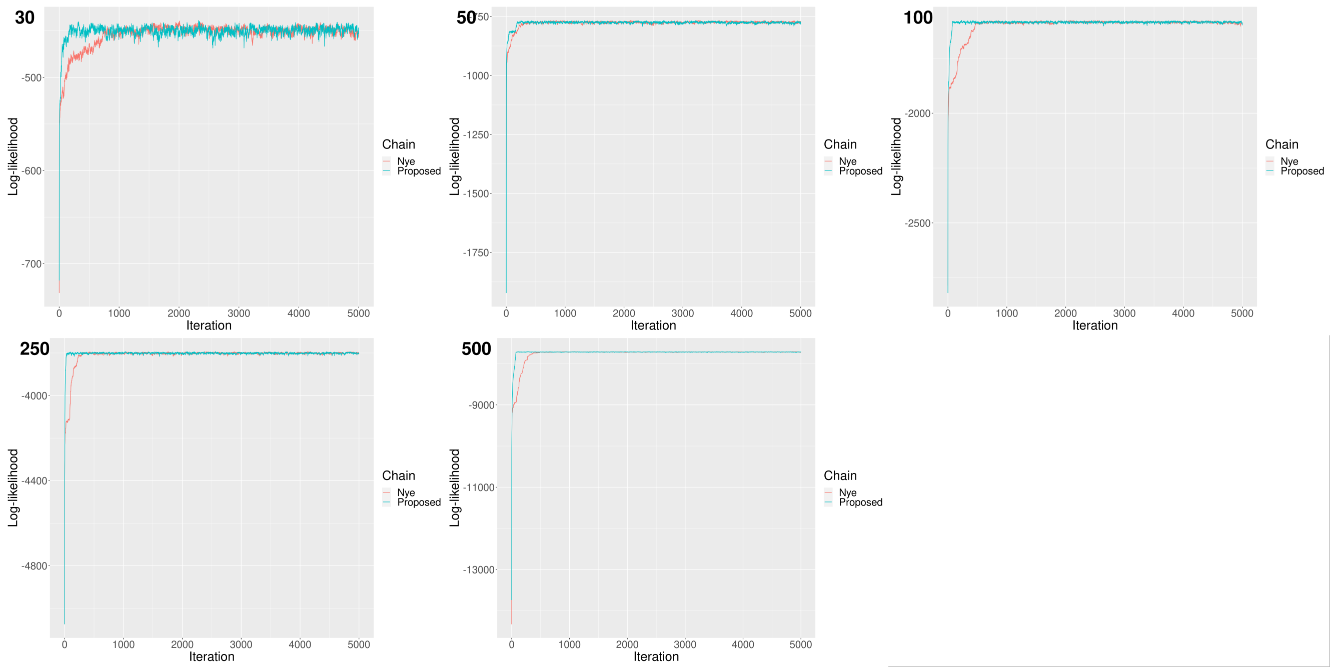

Figure S2: Empirical comparison of our proposed algorithm (blue) and the algorithm from Nye (2020) (red) in terms of the rate of convergence under five different sample sizes of $n \in \{30, 50, 100, 250, 500\}$.

S1.3 Element-wise coverage for t-distribution

We show the nominal coverage for element-wise 95% credible intervals under the mis-specified t-distribution. Specifically, the results of nominal coverage for t_4 and t_3 for five different sizes are shown in Figure S3 and S4, respectively. We observe that the nominal coverages for t-distribution are moderate when the sample size is small ($n = 30$). However, when the sample size increases, the nominal coverage worsens. Unfortunately, when our algorithm is mis-specified, it does not generate posterior matrices that converge to the true matrix. We suspect that our algorithm converges to incorrect edge lengths when the model is mis-specified. This conjecture is supported by Table 1 in the Main Paper, which indicates that our algorithm still generates posterior samples with the correct topology for t-distribution when the sample size is larger with $n = 100$ for t_4 and $n = 250$ for t_3.

S1.4 Topology trajectory for the proposed method

In this section, we examine the trajectory of our proposed algorithm in $\mathcal{U}_p$. Specifically, we initiate our algorithm with matrices that are far away (in terms of distance d of Equation (3) in the Main

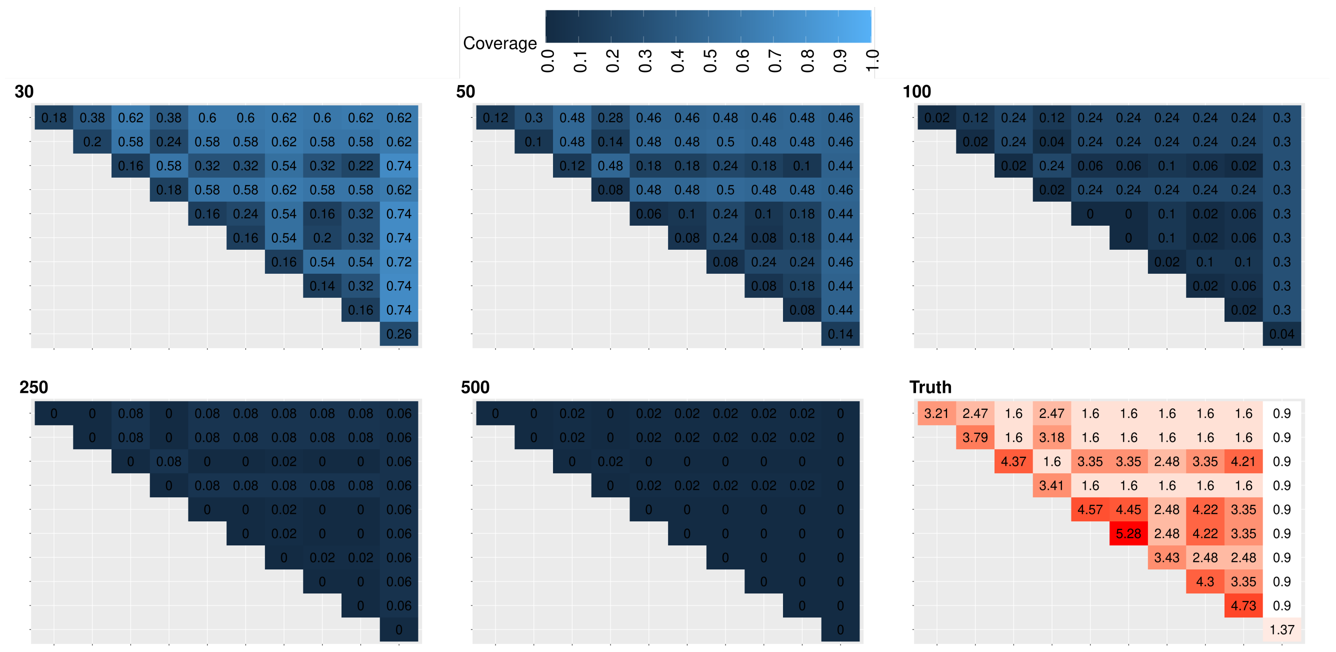
Figure S3: Element-wise coverage from the 95% credit interval for the mis-specified t-distribution of degree of freedom four under five different sample sizes. The true underlying covariance is shown in the lower right panel.

Paper) from the true matrix and track the topologies generated by our algorithm over iterations.

Consider a dataset of sample size $n = 500$ that is generated from an underlying normal distribution with a true ultrametric matrix of dimension $p = 10$. For the ultrametric matrices of $p = 10$, over 34 million possible topology exists, and we index each topology with a natural number. We initiate our algorithm with trees that share no common split with the true matrix and run the algorithm with correct specified normal likelihood for 10,000 iterations.

Figure S5 presents the trajectory of our algorithm applied on the same dataset with 15 different initial trees in terms of the distance d between the estimated matrices and the true matrix. Each Panel is initiated by the matrix with different topology. For example, Panels in the top row shows the results for the algorithm initiated by matrices with five different topology $2, 27, 52, 72$ and 91, from the left to right. For each Panel, different colors represent different topologies. Obviously, all initial matrices are distant from the true matrix with a higher distance d. Over iterations, our algorithm traverses different nearby orthants and quickly moves to the correct topology (topology 1) with a smaller distance d to the true matrix. For example, the top-left Panel shows the distance d for the algorithm initiated by the matrix with topology 2. Since the initial topology 2 share

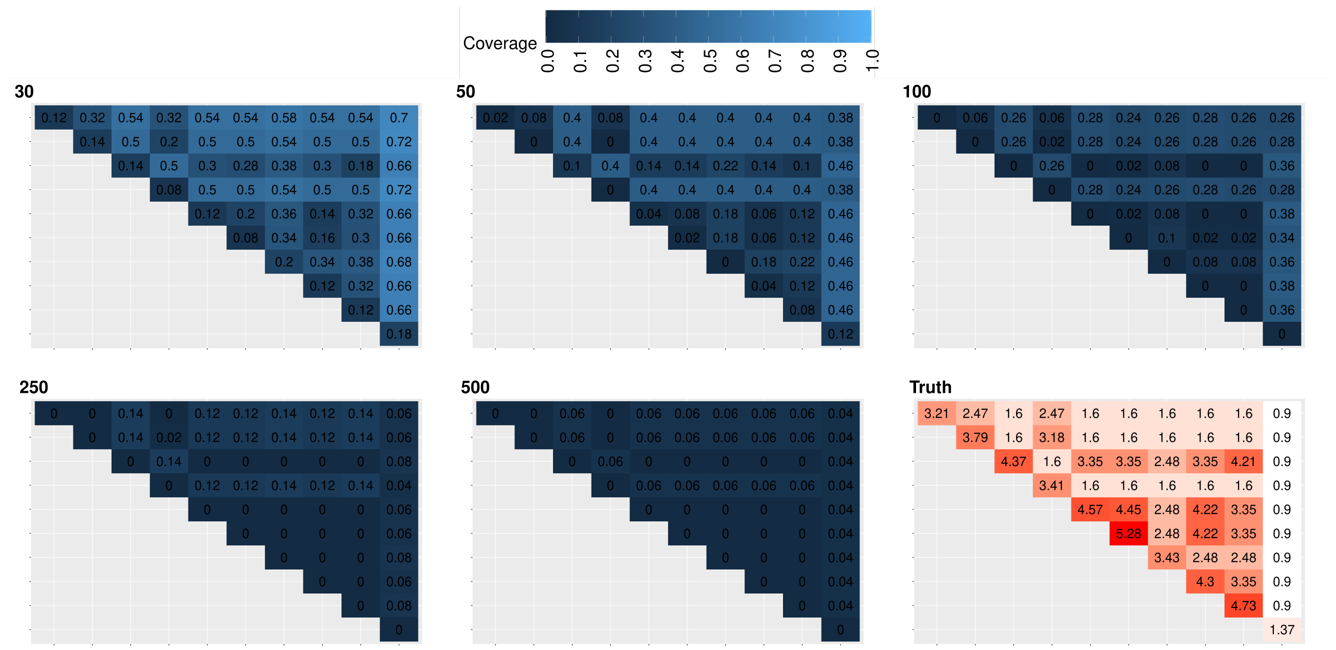

Figure S4: Element-wise coverage from the 95% credit interval for the mis-specified t-distribution of degree of freedom three under five different sample sizes. The true underlying covariance is shown in the lower right panel.

no common split to the true matrix, a larger distance d is obtained. The algorithm then quick traverses the topologies 3 to 26 and lingers in the correct topology 1 within the first 150 iterations with a rapid decrease in distance d. After reaching the true topology 1, we observe that our algorithm sometimes moves to the nearby orthants of topologies 25 and 26, but moves back to the correct topology 1 soon after a short period.

S1.5 Simulation results for the tree with the same edge lengths from the root

In this Section, we demonstrate additional simulation results when all leaves in the true underlying tree are equidistant to the root. Specifically, we obtain a tree from the coalescence model implemented by the function `rcoal` in the `R` package `ape` and generate the data from the normal and t-distribution described in the Main Paper Section 6. We run Algorithm 1 without restricting the prior on the edge lengths for 10,000 iterations and discard the first 9,000 iterations. We summarize the posterior samples by the point estimation and the quantify the uncertainty through the

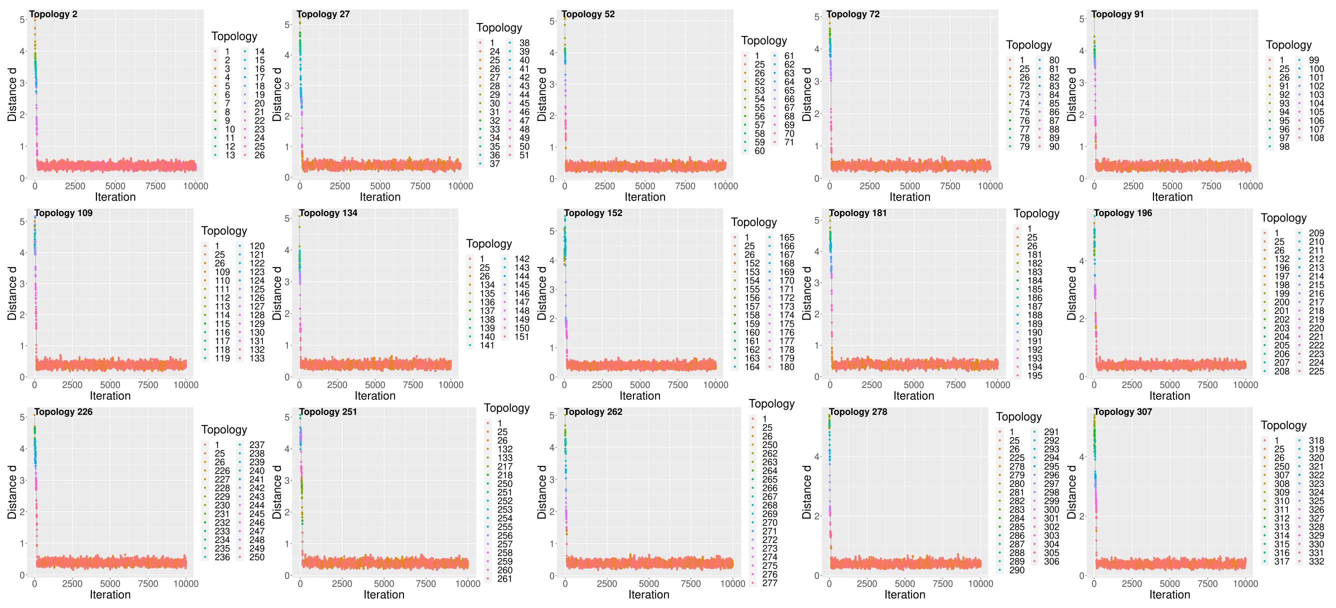

Figure S5: Trajectory of our algorithm in terms of distance d. Over iterations, distances d between each posterior matrix and the true matrix are measured. Each posterior sample is colored according to the corresponding topology. The same algorithm is initiated by 15 different matrices that are far away (in terms of the distance d) from the true matrix. Our algorithm traverses different orthants and arrives at the true topology (topology 1) quickly after a few iterations.

element-wise 95% credible interval. The performance of the point estimator is compared to the projection-based method of Bravo et al. (2009) and the sample covariance under the measurement of the distance d from Equation (3) in the Main Paper and the matrix norm. For the uncertainty quantification, we calculate the nominal coverage of the 95% credible interval. All results were obtained from 50 independent replicates.

Figure S6 demonstrates the nominal coverage of the element-wise 95% credible interval with the true generating covariance in the Panel (F). As we expected, the diagonal elements in the true underlying covariance are equivalent, implying that all leaves in the true underlying tree are equidistant to the root. Similar to the results shown in Main Paper Section 6, the 95% credible interval gives a high nominal coverage (around 0.73 to 1), and the estimated coverage is higher when the sample size increases. In summary, our algorithm can efficiently draw posterior samples of matrices under different conditions imposed on the edge lengths of the true underlying tree.

The results of the point estimators are shown in Figure S7. For the point estimator, the mean and MAP trees from our method are comparable to the estimated matrix from Bravo et al. (2009)

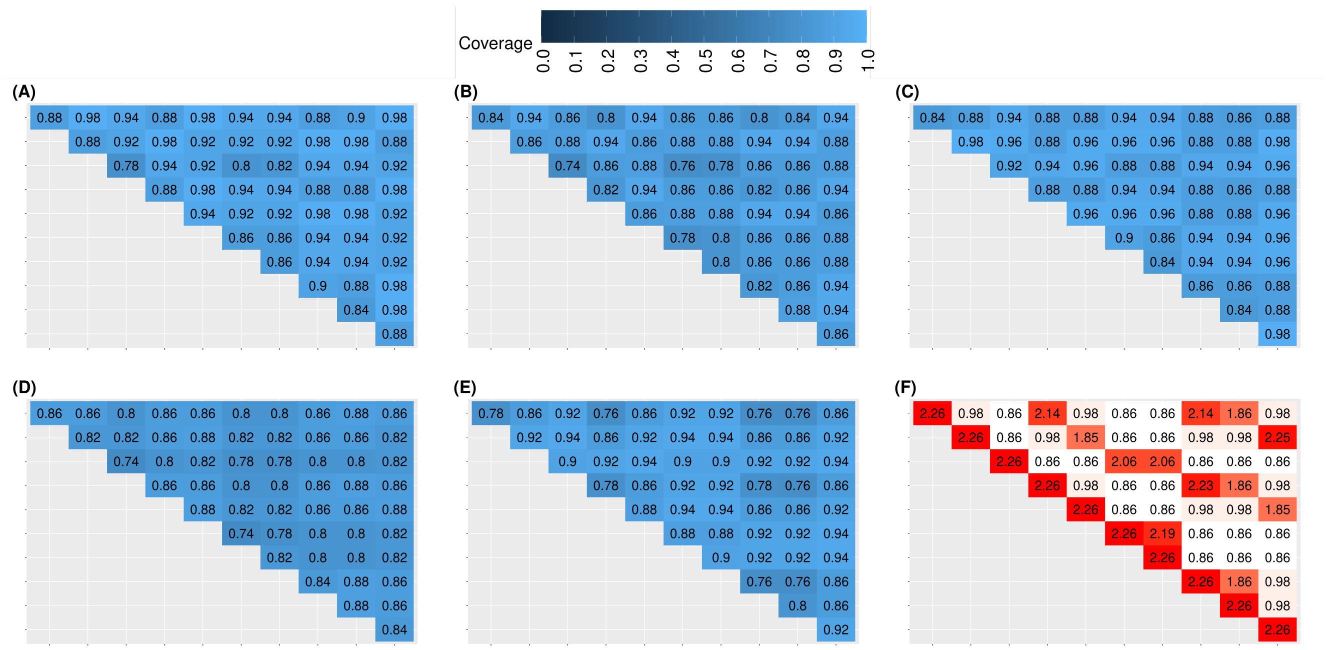

Figure S6: Element-wise coverage from the 95% credit interval for the correct specified normal distribution with five different sample sizes with the true underlying matrix in the Panel (F). Equal diagonal elements in the true matrix indicate that all leaves in the true underlying tree are equidistant to the root.

and sample covariance in terms of distance d and matrix norm across different data generating mechanisms and sample sizes. When the model is correctly specified, all methods benefit from the increase in the sample size, resulting in a smaller distance to the true matrix. For the misspecified scenario, the advantage from the larger sample size is moderate. Essentially, when all leaves in the true underlying tree are equidistant to the root, our algorithm still generates posterior samples that are comparable to existing methods with a similar level of performance in terms of the distance d and matrix norm.

S1.6 Scalability of the proposed method

We assess the scalability of the proposed method by considering a larger dimension of $p \in \{10, 20, 30\}$ for the true matrix Σ^{T^0}. We repeat the same data generating mechanism of the normal distribution of $X_i \overset{i.i.d.}{\sim} N(0, \Sigma^{T^0})$ with $i = 1, \ldots, n$ and $n = 50p$. We measure the running time of the proposed Algorithm with 10,000 iterations and compare it to the running time of the competing method MIP (Bravo et al., 2009). All results are executed on Linux-based cluster with a CPU 2x 3.0 GHz Intel Xeon Gold 6154 and varying memory sizes required for different methods.

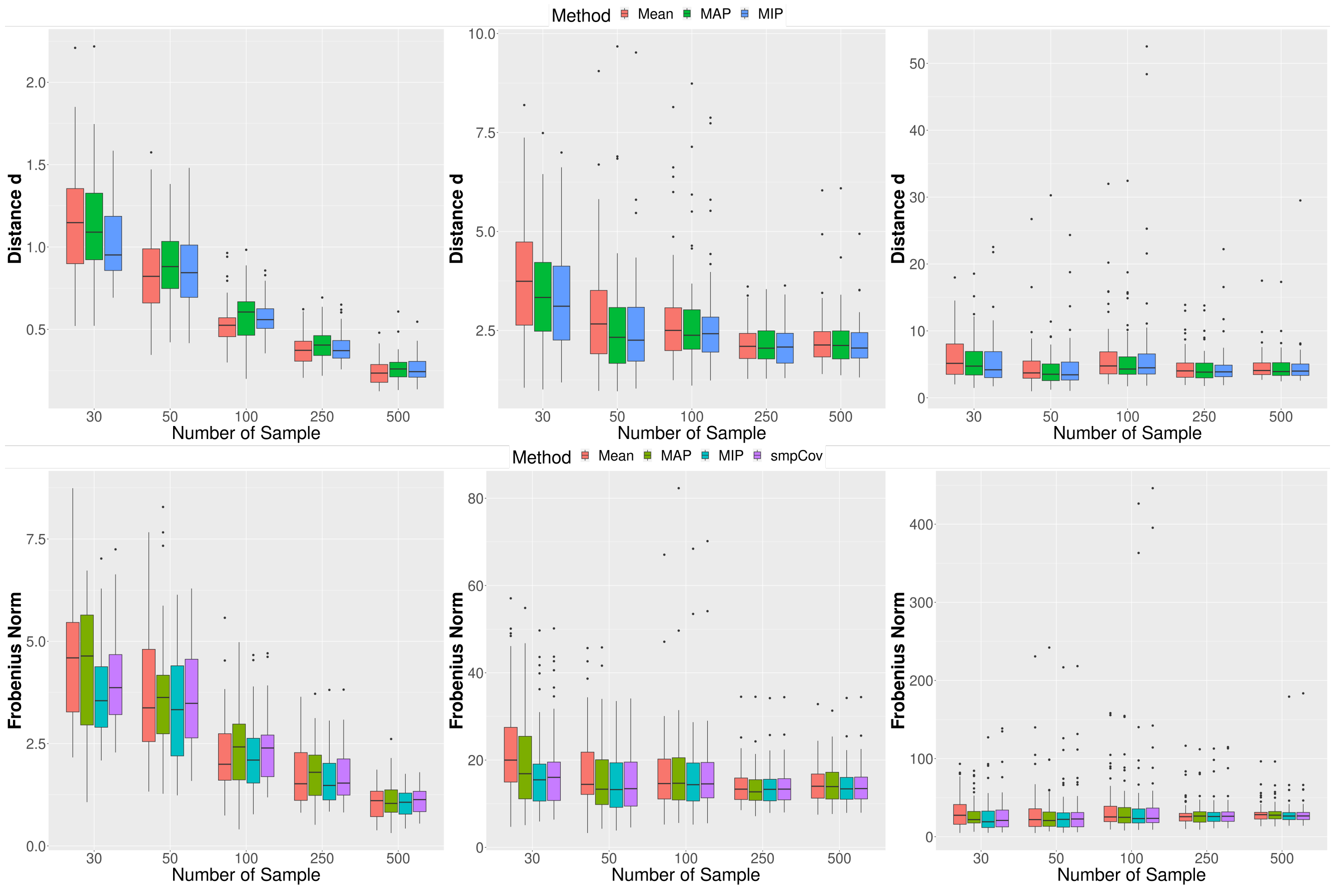

Figure S7: Distances between the estimated matrix and the true matrix under different data generating mechanism and sample sizes. The mean (red) and MAP tree (green) from our method is comparable to competing methods (blue for MIP and purple for sample covariance) in terms of the distance d (top row) and matrix norm (bottom row).

Specifically, we assign 1GB of memory for the proposed method and 25GB of memory to MIP.

The results of the running time are shown in Figure S8. For $p = 10$, MIP is faster than the proposed MCMC (median running time in seconds for MIP: 154 and MCMC: 511). However, when considering a larger dimension of $p = 20$, the proposed MCMC takes around 30 minutes (median: $1,788$ seconds), which is 1.75 times faster than the time required for MIP (median: $3,080$ seconds). If we increase the dimension to $p = 30$, the proposed method spends 66 minutes (median: $3,976$ seconds), while MIP fails after running for 3 days due to memory and computation limits. In conclusion, our method scales better in p with a much efficient memory requirement.

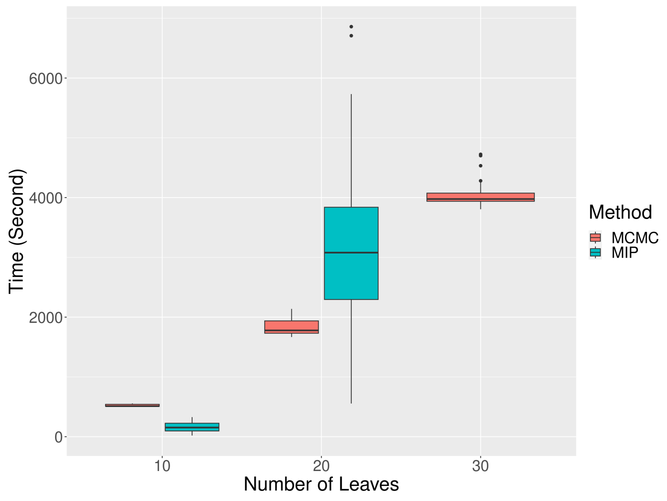

Figure S8: Computation time (in seconds) required for the proposed method (MCMC) and the competing method (MIP) under three different number of leaves $p \in \{10, 20, 30\}$. The computation time of MIP for $p = 30$ is missing for the MIP fails after running for 3 days due to the memory and computation limits.

S2 Additional Results for the Real Data Application

We run the convergence diagnostics of our algorithm through the likelihood trace plot. Specifically, given the PDX dataset, we run the same algorithm with two chains initiated by two different trees and plot the likelihood over iterations. When the algorithm is converged, the likelihood initiated by two different chain should reach a similar level. Obviously, the likelihood increases rapidly and remains at a relatively high plateau. More importantly, both chains converge to a similar level of likelihood, indicating the convergence of the algorithm.

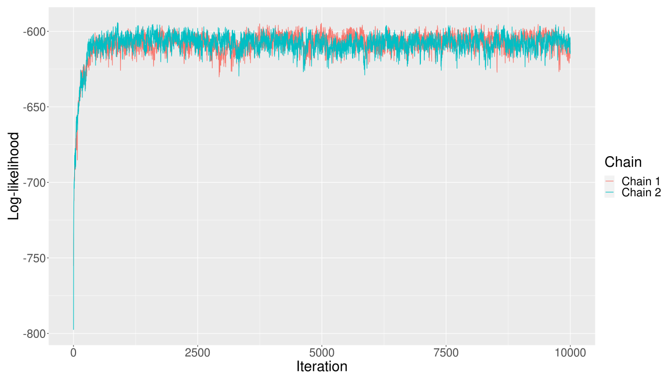

Figure S9: Convergence diagnostics for the algorithm on PDX dataset by using the likelihood trace plot. Two chains of the same algorithm are initiated by different trees, shown by two colors.

References

Bravo, H. C., Wright, S., Eng, K. H., Keles, S., and Wahba, G. (2009). Estimating tree-structured covariance matrices via mixed-integer programming. *J Mach Learn Res*, 5:41–48.

Nye, T. M. (2020). Random walks and Brownian motion on cubical complexes. *Stochastic Processes and their Applications*, 130(4):2185–2199.