University of California
Santa Barbara

Gaussian Models: Regularization, Imputation, and Emulation

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Statistics and Applied Probability

by

Yuanbo Wang

Committee in charge:

Professor Sang-Yun Oh, Chair
Professor Wendy Meiring
Professor Sreenivasa Rao Jammalamadaka
Professor Christina Tague

March 2022

The Dissertation of Yuanbo Wang is approved.

Professor Wendy Meiring

Professor Sreenivasa Rao Jammalamadaka

Professor Christina Tague

Professor Sang-Yun Oh, Committee Chair

December 2021

Gaussian Models: Regularization, Imputation, and Emulation

by

Yuanbo Wang

To Dad, Mom, and Hong

Acknowledgements

First and foremost, I would like to express my deepest and sincere gratitude to my advisor Dr. Sang-Yun Oh, for his continuous support during my Ph.D. study. Completing the research work for earning a doctorate is long and arduous. There are moments full of uncertainty that can be discouraging. Thanks to Sang, professor and mentor, those moments passed more easily than they appeared to be. His meticulous scrutiny, keen interest in truth, unselfish dedication have inspired me to a great extent, not only for completing the doctoral study but also for my future endeavors in my career and life.

I would also like to extend my deep gratefulness to Dr. Wendy Meiring, Dr. Sreenivasa Rao Jammalamadaka, and Dr. Christina (Naomi) Tague for their help and invaluable guidance in the past years. I was deeply inspired and encouraged by Dr. J for pursuing my academic goals. Had it not been for his motivation and engagement, I do not think I would have ever initiated the process of pursuing a Ph.D. degree. Both Wendy and Naomi are brilliant and inspiring role models for female scientists who seek success in their areas. Without Wendy's insightful comments and constant encouragement and Naomi's enthusiasm and continuing support, the fourth chapter would never be there.

I can not leave PSTAT without mentioning Dr. Andrew Carter, Myranda Flores, Jamie Pillsbury, and Patrick Windmiller. Dr. Carter has served as the graduate advisor and provides timely suggestions and help whenever I have questions during teaching and studying. Myranda and Jamie have offered tremendous help serving graduate students throughout my study period. Patrick has helped with the computational side of completing my research work.

I am incredibly grateful to my parents for their enduring love and sacrifices for educating and preparing me for a bright future. Pop, thank you for being the perfect father, mentor and friend and for holding the umbrella for me as long as you could. Your love is

forever engraved in my heart. It's been a long journey without you, and I'll tell you all about it when I see you again. Mom, thank you for being strong and brave throughout the years. I truly appreciate it. All the hardship and wait will pass, and we will see the sunshine together. Life is too short; it is now time to seize all moments by holding our hands together. I can not imagine life without you, and honestly, I am looking forward to the day when you become someone's lovely granny.

I am deeply indebted to Hong (Herbert), who has accompanied me through these years. Life is never easy, but Hong is always there.

My deepest gratitude must also go to my uncle, Weidong Wang, for his care, support, guidance, and love in these years, even during his hardest time.

I would like to thank all my besties. Huimin, I could not have my life today without you and your family taking care of my family in the past eight years. Yulin, I could not be this strong today facing all challenges without you insisting on a positive attitude towards everything. Lanni, I could not firmly believe that life always has a gentle and comforting side without the warmth from you and your family. Yaqian, I could not see myself clearly and put myself together without you being a mirror and a glue. Jiaye, I could not have completed these years without you being a steady supporter and witness. Guo, I could not get over the main difficulty without your insight into it.

Last but not least, special thanks must go to Zach, Bret, Jenifer, Nhan, Mike, Ke, and Zhipu. I enjoyed all the moments, joyful or painful, with all of you on the fifth floor. Friendship lasts forever.

Curriculum Vitæ

Yuanbo Wang

Education

2021	Ph.D. in Statistics and Applied Probability (Expected), University of California, Santa Barbara.
2017	M.A. in Mathematical Statistics, University of California, Santa Barbara.
2014	B.S. in Economics and Mathematics, Tianjin University of Commerce.

Work Experience

Department of Statistics and Applied Probability, UCSB

- Research Assistant
  - Developed a regularization selection algorithm for tuning the Gaussian graphical models.
  - Imputed missing values using graphical model-based methods in several applications.
  - Worked on many high-dimensional datasets including gene expression, paleoclimate temperature field, and microbial community data.
- Teaching Associate
  - Served as instructor of record for *Principles of Data Science with R* in Summer 2019.
  - Taught weekly lectures and created course materials.
- Teaching Assistant
  - Served as teaching assistant for multiple courses such as data mining, statistical machine learning, time series, regression analysis and introductory statistics.

Bren School of Environmental Science and Management, UCSB

- Research Assistant
  - Built fast and accurate surrogate models for emulating a complex physics-based simulator RHESSys.
  - Provided sensitivity analyses for understanding the input-output relationship of the simulator at different experimental sites.

Senate Office, UCSB

- Data Analyst
  - Cleaned and summarized survey data acquired from the international undergraduate student survey.
  - Performed correlation and trend analysis and reported to the Senate Office.

Programming and Softwares

- Python, R, MATLAB, SQL, SPSS, EViews, LaTeX

Abstract

Gaussian Models: Regularization, Imputation, and Emulation

by

Yuanbo Wang

Recent years have seen great advances in using Gaussian graphical models to characterize the conditional relationship among variables in many domains of study. In particular, many methods have been proposed for estimating the inverse covariance matrix. Along this line of research, glasso (graphical lasso, proposed by Friedman et al. (2008)) provides an l_1-regularized maximum likelihood estimator. One challenge in such regularization-based methods is determining the scalar tuning parameter that balances the model complexity and fit to the data, the latter frequently based on the likelihood. When working in high dimensions, traditional model selection methods such as k-fold cross-validation, Bayesian information criterion, and Akaike's information criterion can be challenging to apply for several reasons. First, the computation can be prohibitively expensive when estimating high-dimensional inverse covariances multiple times. In addition, reasonable search grids for candidate penalty parameter values can vary considerably across applications. Substantial effort is required to find reasonable search ranges for different applications. Furthermore, using homogeneous regularization for all entries in the inverse covariance matrices can be limiting.

To address these challenges, we first propose block-wise robust selection (BRS), a tuning method based on distributionally robust optimization for selecting block-wise regularization parameters in the glasso estimator. This method finds adaptive penalty parameters for different blocks in the inverse covariance matrix, where the blocks are determined based on data dispersion. In this formulation, the previous penalty parameter

search in an arbitrary range now becomes a search of significance level within the fixed interval of $[0, 1]$, regardless of the application of interest. Our method is computationally efficient and does not require data normalization prior to estimating the inverse covariance matrix.

Next, we demonstrate the application of BRS to the problem of climate field reconstruction, which aims to reconstruct the past temperature evolution by making use of the measurements in the post-instrumental period and partial records in the pre-instrumental times. The reconstruction can be viewed as a missing value imputation task. In these applications, we first use a Gaussian graphical model tuned via BRS to characterize the spatial field over the globe and then perform the imputation. In addition, we explore different clustering methods for grouping the variables in BRS. The reconstruction results confirm that our method is computationally attractive and provides similar imputed values when compared to using a graph tuned by environmental scientists. Furthermore, BRS can be used flexibly with different variable grouping methods.

Finally, we consider the emulation of physics-based simulators for environmental processes, leveraging Gaussian Processes. Our goal is to develop a computationally efficient surrogate model to closely approximate the outputs of the physics-based environmental simulator, which is expensive to run. Specifically, we develop an emulator for the Regional Hydro-Ecologic Simulation System (RHESSys) simulator. Our emulator leverages Gaussian Processes with embedded seasonality within the mean and a separable covariance. The emulator provides an efficient way to approximate the output that would be obtained by running the physics-based simulator at a substantially lower computational cost than running the simulator. Our emulator approximates environmental time series that would be generated by the physics-based model, e.g., streamflow, under different hydrological and ecological scenarios, e.g., different soil properties. In addition, the degree of approximation and computation efficiency of our built emulator enables us to conduct

a global sensitivity analysis on the input-output relationship of the environmental process of interest and identify the key influential environmental factors. Without our emulator, such an analysis would be intractable (or very expensive), as one would need to run the physics-based simulator multiple times for various input settings, which is very costly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Preliminaries

Understanding the underlying structure and information in data is of great importance for a wide variety of domains, such as finance, genetics, and environmental science. Gaussianity plays a fundamental role in these problems for characterizing the distribution of data. Specifically, one key approach to describe the variation in data is through the covariance or the inverse covariance matrix. In this dissertation, we explore different aspects of Gaussian models, including how to estimate the inverse covariance matrix with proper regularization as well as applying Gaussian models to different environmental science applications.

We first consider Gaussian graphical models and the regularization of them via our proposed method, block-wise robust selection. While the estimation of the inverse covariance matrix is straightforward in data-sufficient cases, it becomes challenging when data is deficient. One intuitive way to obtain the inverse covariance matrix is to invert the sample covariance matrix. However, this is not feasible when the number of variables greatly exceeds the sample size of the data. This causes the sample covariance to be rank deficient, thus resulting in an ill-posed problem. To work around this difficulty, one approach is to estimate the undirected graphical model that encodes the conditional correlations among the variables under the Gaussian assumption. Central to this approach is the estimation of the inverse covariance matrix through regularization-based

optimization, where the regularization maintains a balance between model complexity and data likelihood. When doing this, it is critical to select a regularization parameter that results in a proper level of sparsity in the Gaussian graphical model. Although classical approaches such as cross-validation can be used for this purpose, they can be computationally expensive, and it can be difficult to find a set of good candidate values. Moreover, a scalar regularization parameter may not be sufficient to cope with the different levels of variability in the variables. This calls for a new and efficient way of determining such regularization parameters.

The estimated Gaussian graphical model and inverse covariance matrix can be very useful for data analysis and statistical inference in various real-world problems. For instance, in environmental science, the estimated graphical model can be used for climate field reconstruction (Guillot et al. (2015); Neukom et al. (2019); Vaccaro et al. (2021)), i.e., imputing the missing fields in pre-instrumental years. In this problem, the Gaussianity is assumed for the spatial grids over the globe, and one can estimate a Gaussian graphical model for them, based on the complete observations in recent years. The estimated model can then be used to infer the missing temperature values in the early years when temperature records were incomplete.

Lastly, we consider the emulation of a physics-based simulator that describes environmental processes through Gaussian Processes. The covariance is used to measure how the features (inputs) affect the outcomes (outputs) of the simulator. The Gaussian Processes-based method is applied to the emulation of complex environmental models, which involves building computational efficient surrogate models to approximate/emulate the input-output behavior of highly complex environmental simulators. In our study, we consider the emulation of the hydro-ecological model, Regional Hydro-Ecologic Simulation System (RHESSys). Our goal is to discover how the soil parameter inputs influence the time series outputs such as streamflow at experimental sites under different climatic

conditions. Furthermore, we study the sensitivity of the outputs to the input soil conditions and how this varies across experimental sites.

In this dissertation, we explore the aforementioned problems through the following topics.

- *Block-wise Robust Selection*: We propose a novel way of tuning the Gaussian graphical model, which finds regularization parameters for different blocks of entries in the inverse covariance matrix. By doing this, we allow the regularization to adapt to different levels of data dispersion in the variables and avoid data normalization in the estimation process, thus preserving the original scales of data. Our method requires lower computational costs as compared to k-fold cross-validation and is shown to perform well in simulation studies.

- *Paleo-Climate Reconstruction Using Block-wise Robust Selection*: We incorporate our tuning method, block-wise robust selection, into the GraphEM (Guillot et al. (2015)) algorithm for reconstructing the paleo-climate temperature field. Our tuning method is computationally more attractive as compared to prior work. The reconstruction results are reasonable when compared to other published results.

- *Emulation of RHESSys*: We emulate RHESSys using a Gaussian Process-based model and provide a sensitivity analysis of the RHESSys simulator. The emulator embeds seasonality in the mean and uses separable covariance that combines the temporal covariance and soil covariance. We conduct variance-based sensitivity measures of first-order, second-order, and total-order to understand the soil-streamflow relationship and provide insights for calibrating RHESSys.

1.1 Block-wise Robust Selection

In Chapter 2, we work on the Gaussian graphical model and consider tuning the hyper-parameter, which balances the model complexity and the data likelihood. Undirected graphical models have been extensively used to characterize the conditional relationship among variables, which have been seen in a diverse range of scientific areas such as genetics, social sciences, and finance. Under the Gaussian assumption, the conditional independence relationship can be captured by the inverse covariance matrix of the variables. Inferring this matrix and learning the corresponding model structure in high dimensions have been extensively studied in the context of various real-world problems. Along this line of research, researchers have proposed estimating the graphical model through regularization. In other words, they formulate this problem as a constrained optimization, where the objective function balances the goodness-of-fit and the complexity of the model. Common choices of measuring the goodness-of-fit include the likelihood or pseudo-likelihood of the data under proper distributional assumptions. In terms of the regularization term, lasso is one of the most researched methods for constraining the complexity, which performs variable selection and regularization at the same time.

One important challenge for these regularization-based methods is to select the proper amount of regularization when solving the aforementioned optimization. Specifically, the learned model should capture as much variability in the data as possible while still being generalizable. Towards this end, the commonly adopted solution is to employ standard statistical methods for model selection such as cross-validation and forward and backward selection. However, these methods can require prohibitively expensive computation to produce a model. Moreover, these methods also require a proper search strategy to cover a range of hyper-parameters, which may not be straightforward to design.

More recently, the robust selection (RS) algorithm has been proposed in Cisneros-

Velarde et al. (2020), which can do away with the expensive hyper-parameter search and directly derive the regularization parameter through a Distributionally Robust Optimization. As such, its practical implementation is less complex and the required computation time is much smaller. In this chapter, we further propose a method for fine-tuning the regularization for different blocks of entries in the inverse covariance matrix based on the RS algorithm. More specifically, rather than imposing the same amount of regularization on the entire model as in the original RS algorithm, we design penalty parameters for different sub-matrices in the inverse covariance matrix, which we refer to as the block-wise robust selection (BRS) algorithm. By doing this, we allow more flexible penalties for variables with different levels of variance.

In this part, we provide the necessary background and discuss the related work on undirected graphical model selection and inverse covariance estimation. The conditional independence relationship of a set of variables can be represented by an undirected graph $G = (V, E)$, where V is the set of vertices corresponding to the variables and E is the set of edges connecting any pair of the vertices. In particular, two variables are connected by an edge if they are conditionally dependent, given the remaining variables. Suppose that independent and identically distributed ($i.i.d.$) samples $(X_1, \ldots, X_n)$ are drawn from a p-dimensional Gaussian distribution with mean zero and covariance Σ. Let $K = \Sigma^{-1}$ denote the precision matrix and let A denote the sample covariance matrix. Under the normality assumption, variable i and j are conditionally independent of each other given the rest if and only if $K_{ij} = 0$ $(1 \leq i \neq j \leq p)$ (Lauritzen (1996)). Thus, the sparsity pattern presented in the precision matrix dictates the edge structure of the conditional independence graph G. Consequently, the objective of the undirected graphical model selection problem is equivalent to identifying the nonzero entries in K.

A straightforward way to obtain K is to take the inverse of the sample covariance matrix A, which is the maximum likelihood estimator (MLE), when $p < n$. However,

A becomes rank deficient when $p > n$, making inverting A ill-posed and infeasible. To address this issue, a series of optimization-based approaches have been proposed, which introduce a penalty term to regularize the problem. We summarize these methods by a general formulation:

$$\underset{K}{\operatorname{argmax}}\ L(K) - P(K),$$

where $L(\cdot)$ denotes the likelihood function and $P(\cdot)$ denotes the non-negative penalty term.

Several candidates for $L(\cdot)$ have been studied in the literature. For instance, Friedman et al. (2008); Yuan and Lin (2007); Yuan (2008); Rothman et al. (2008) have used Gaussian likelihood function assuming the data distribution is multivariate normal. On the other hand, Peng et al. (2009); Meinshausen and Bühlmann (2006); Khare et al. (2015) have proposed a variety of pseudo-likelihood functions for cases where the normality assumption does not hold, which is common in practice. More specifically, the pseudo-likelihood function can be the regression residual-based loss function from node-wise regression (see Peng et al. (2009); Meinshausen and Bühlmann (2006); Khare et al. (2015)). These methods then aim to solve for the maximum pseudo-likelihood estimator. In these cases, while maximizing $L(\cdot)$ is straightforward in low dimensional problems, it becomes much more challenging in high dimensions. Specifically, the rank deficient problem, i.e., $p > n$, makes the estimation problem ill-conditioned.

A common way to remedy the aforementioned rank-deficient problem is to add a regularizer $P(\cdot)$ to the original likelihood-based objective function. The penalty term serves to constrain the model complexity, which leads to sparse, better-conditioned inverse covariance estimators. Various sparsity-promoting penalties have been proposed in the literature, including l_0 penalty (Edwards (2012)), l_1 penalty (Meinshausen and Bühlmann (2006); Peng et al. (2009); Khare et al. (2015); Friedman et al. (2008); Yuan

and Lin (2007); Yuan (2008)), and SCAD penalty (Fan et al. (2009)). Although the l_0 penalty naturally induces sparsity, it is non-convex and thus computationally infeasible to use in high dimensional settings. The convex l_1 penalty is a natural relaxation of the l_0 penalty and thus becomes the prevalent choice in inverse covariance estimation problems. The non-concave SCAD penalty, as well as adaptive Lasso (Zou (2006)), are proposed to reduce the enlarged bias that can be caused by the l_1 penalty.

In this paragraph, we discuss one prominent method for Gaussian graphical model selection, graphical lasso (glasso), which sets the prerequisite for introducing our proposed methodology. In glasso, it is assumed that the observations come from a multivariate normal distribution with mean zero and positive definite covariance matrix Σ. The goodness-of-fit of the graphical model is measured by the Gaussian log-likelihood function, which can be written as $L(K) = \log|K| - \mathrm{tr}\,(AK)$, where $K = \Sigma^{-1}$. Regularization is through lasso penalty which takes the form $P(K) = \lambda\|K\|_1$. The explicit form of the problem is as follows:

$$\underset{K\succ 0}{\operatorname{argmax}}\ \log|K| - \mathrm{tr}\,(AK) - \lambda\|K\|_1. \tag{1.1}$$

To solve this optimization problem, Banerjee et al. (2008) has proposed a block coordinate descent algorithm using interior point methods. Friedman et al. (2008) has proposed a fast coordinate descent algorithm. There are many other speed-up variants for solving this problem, e.g., Mazumder and Hastie (2012); Hsieh et al. (2013). To solve the optimization, we need to specify λ in advance, which requires either domain knowledge or parameter tuning, as we shall discuss next.

It is critical to determine the proper penalty parameter λ in these optimization problems in order to generate well-conditioned estimators.[1] More specifically, all of the afore-

[1]Note that a penalty parameter is present in the general constrained optimization formulation and not just in the graphical lasso.

mentioned penalty schemes $P(K)$ have a hyper-parameter λ that needs to be fixed or tuned prior to solving the optimization. In general, higher values of λ lead to more regularization and thus sparser models, whereas lower values impose less penalty and thus more complex models. One way to select λ is to leverage the existing domain knowledge for the problem at hand. However, this requires deep domain expertise and extensive empirical studies, which may not be available for any given problem. Furthermore, such hand-tuned values can be vulnerable to subtle disturbances, such as the shift of data distribution, violation of model assumptions, and anomalies in the data.

Standard statistical methods for model selection are often used as data-driven alternatives for choosing the regularization parameter, such as k-fold cross-validation (hereinafter k-fold CV), Bayesian information criterion (BIC), and Akaike's information criterion (AIC). These procedures work by iterating over different values of λ and selecting the best one according to a certain criterion. One practical challenge while using these methods is to come up with a proper range/set of candidate λ values to perform the search, in which the optimal value should be included. However, finding such a search range is not trivial; it requires a deep understanding of the model selection mechanism as well as trials and errors. In addition, applying these methods is computationally expensive, especially in the high-dimensional setting, as the optimization needs to be solved for every possible value of λ. For instance, the computation for k-fold CV can be prohibitively expensive when the set of candidate values is large, as the high-dimensional inverse covariance matrix needs to be estimated multiple times. Furthermore, the quality of the obtained model using the selected λ depends heavily on how well the selection criterion aligns with the problem of interest.

Researchers have proposed computationally faster and model stability-based approaches to address this parameter tuning and model selection problem (Meinshausen and Bühlmann (2010); Liu et al. (2010)). Stability Selection proposed by Meinshausen and Bühlmann

(2010) is a general approach that combines sub-sampling with high dimensional model selection mechanics. In the context of graphical model selection, this approach uses sub-samples and finds the edges with high selection probabilities in the undirected graph. StARS is another sub-sampling-based method for Gaussian graphical model selection, which selects the penalty by having the total edge instability not exceeding a certain threshold.

More recently, Cisneros-Velarde et al. (2020) have formulated the p-norm regularized Gaussian graphical model selection problem via distributionally robust optimization (DRO). When $\rho = 1$, the size of the ambiguity set in DRO corresponds to the penalty parameter λ in glasso. The algorithm for finding such a λ is referred to as the robust selection (RS) algorithm. Central to the RS algorithm is the robust Wasserstein profile (RWP) function, which is computed based on bootstrap samples. The definition of the RWP function is given as follows:[2]

Theorem 1.1 *(RWP function, Theorem 3.2 in Cisneros-Velarde et al. (2020)). Consider the cost function* $c(U, V) = \|\boldsymbol{vec}\,(U - V)\|_\infty$ *for symmetric matrices* U *and* V*. For a positive definite* K*, the RWP function is defined as:* $R_n(K) = \|\boldsymbol{vec}\,(A - K^{-1})\|_\infty$*, where* $\boldsymbol{vec}(\cdot)$ *denotes matrix vectorization and* A *is the sample covariance matrix.*

The RS algorithm works by computing the RWP function on bootstrapped datasets, which is obtained by sampling the original data with replacement. The sample covariance matrices are then computed on each bootstrapped dataset, as well as on the original data. Next, the regularization parameter λ is set to the $\lfloor (B+1)(1-\alpha) \rfloor^{th}$ order statistic of the RWP function. As a result, RS only requires estimating the inverse covariance matrix once after the penalty parameter is determined and therefore reduces computation, as compared to conventional methods like k-fold CV.

[2]See the original paper for more details on the RWP function, including its convergence properties and related concepts such as ambiguity set.

While RS provides a theoretically rigorous and computationally undemanding solution for penalty parameter tuning, it assumes a single scalar parameter, i.e., the same amount of regularization is applied to all variables and edges via

$$P(K) = \lambda \|K\|_1.$$

This is also the case for most existing l_1-penalized methods for inverse covariance estimation, e.g., Friedman et al. (2008); Peng et al. (2009); Rothman et al. (2008); Yuan (2008); Yuan and Lin (2007). When applying a single scalar penalty parameter, all edges are equally penalized in the optimization problem, regardless of the different scales, units, noise levels, and physical meanings of the node variables. As a result, small-variance variables and their edges are likely to be overly suppressed under this uniform penalty. One possible method to mitigate this problem is to perform data normalization. More specifically, the raw data can be transformed by removing certain measures of central tendency and dispersion. However, this inevitably changes the underlying distribution and signal strengths of the variables, thus removing or distorting information contained in the original data.

In principle, one can apply an individual scalar penalty parameter for each variable and each edge. This makes it possible to exhaustively accommodate variables of different scales. More formally, such a penalty function can be written as follows:

$$P(K) = \|\Lambda \circ K\|_1,$$

where $\Lambda = (\lambda_{ij})_{1 \leq i,j \leq p}$ is a symmetric $p \times p$ penalty matrix. In this matrix, each upper-triangular element can be different and $\circ$ denotes the Hadamard product. While this provides the maximum level of modeling flexibility, tuning this matrix of penalty param-

eters may become computationally expensive or even intractable. For example, consider using k-fold CV to select the penalty parameter for each of the $p(p-1)/2$ upper-triangular entries in Λ. Suppose the search grid is of length L, then the computation complexity is approximately $\mathcal{O}(kLp^3)$.

In this work, we propose a block-wise approach which allow a more flexible penalty scheme to adapt to different variable scales while avoiding the expensive way of treating each variable and edge individually. More specifically, we group variables that share similar characteristics (e.g., variance magnitudes). This grouping then results in a block-wise structure of the inverse covariance matrix and thus the penalty matrix. As a result, we can use different penalty parameters for different blocks in the inverse covariance matrix. Entries within the same block are assigned the same amount of penalty. More precisely, entries within a diagonal block capture interactions of variables in the same group, i.e., those that share similar characteristics, while entries in an off-diagonal block capture interactions of variables from two different groups. Different amounts of regularization are applied to different blocks. Then, in order to efficiently tune these parameters, we leverage the robust selection (RS) algorithm and combine it with our block-wise penalty scheme. We refer to our novel, integrated approach as the block-wise robust selection.

1.1.1 Contributions

The main contributions of this chapter are summarized as follows:

1. We propose a block-wise robust selection algorithm (BRS) for regularization selection in the Gaussian graphical models. The method enables adaptive penalty on different blocks of variables and edges based on the levels of data dispersion.

2. Our numerical studies validate the performance of our proposed BRS method in terms of edge recovery.

1.2 Paleo-Climate Reconstruction Using Block-wise Robust Selection

In Chapter 3, we embed our proposed graphical model tuning method, BRS, into the GraphEM algorithm (Guillot et al. (2015)) for reconstructing the paleo-climate field for the pre-instrumental period. Climate field reconstruction in environmental science has long been treated as a missing value imputation problem (Gomez-Navarro et al. (2015); Mann et al. (2007); Smerdon et al. (2011); Schneider (2001)). The goal is to reconstruct the paleo-climate where the temperatures are largely missing due to the lack of instrumental measurements. Towards this end, researchers have proposed many statistical methods for imputation, utilizing the few available data in the pre-instrumental period, temperatures measured in more recent years, as well as other physical variables which are often referred to as proxies, such as tree rings, corals, ice cores, and ocean and lake sediments (Mann et al. (2007); Smerdon et al. (2011); Schneider (2001); Guillot et al. (2015); Neukom et al. (2019); Vaccaro et al. (2021)).

The GraphEM algorithm proposed by Guillot et al. (2015) is a commonly used method for this paleo-climate imputation problem by embedding a Gaussian graphical model into the expectation-maximization (EM) algorithm. Considering the spatial coordinates as random variables, GraphEM first estimates a graph that captures the conditional dependence relationship among them. When estimating the Gaussian graphical model, the algorithm optimizes the Gaussian likelihood along with a regularization term. The regularization term is a key factor to the estimation quality and thus, it is crucial to use a proper amount of regularization. Based on this estimated graph structure, the algorithm then performs imputation via a modified EM algorithm.

In this chapter, we apply our proposed BRS method as an alternative way to derive the amount of regularization for tuning the Gaussian graphical model for paleo-climate

reconstruction. We also compare with other alternative tuning methods, e.g., based on searching for the target sparsity determined from domain knowledge. In addition, we explore variants of our BRS algorithm by using different ways to group the variables. Specifically, in addition to binary segmentation, we experiment with two other methods, i.e., k-means and Gaussian mixture model. Our study is conducted on two high-dimensional spatio-temporal datasets from environmental science for paleo-climate reconstruction.

1.2.1 Contributions

1. Our proposed tuning method, block-wise robust selection, is applied in learning the Gaussian graphical model for imputing the two environmental datasets. It is shown that our method performs well and is computationally attractive in terms of finding the graph structure for down-streaming imputation.

2. A few variants of the method to determine the variable blocks in block-wise robust selection are explored and their effects to the imputation results are discussed.

1.3 Emulation of RHESSys

With the rapid developments of computer hardware and software in the past decades, mathematical models that characterize complex physical processes have been extensively implemented in computer codes, as described in Kennedy and O'Hagan (2001); O'Hagan (2006); Mohammadi et al. (2019). These implemented models are commonly referred to as simulators. Simulators enable researchers to gain more in-depth and comprehensive understanding of real-world processes in a more feasible way as compared to observing the actual natural processes. In order to closely capture the reality, simulators are often

based on highly complicated mathematical models and require significantly computation power and time to run. In particular, such high computation costs become a critical issue for conducting Monte Carlo experiments with the simulator.

To address this challenge, researchers have proposed surrogate models for the simulators, which are significantly faster to run and closely approximate important components of the simulator output; see Liu et al. (2009); Gladish et al. (2018); Mohammadi et al. (2019); Oyebamiji et al. (2019). These surrogate models are known as emulators or meta-models in the literature (Kennedy and O'Hagan (2001); O'Hagan (2006); Liu et al. (2009); Saltelli et al. (2008); Conti and O'Hagan (2010)). In addition, emulators can be used to provide statistical pathways for calibrating the simulator and for conducting sensitivity analysis, as discussed in Kennedy and O'Hagan (2001); O'Hagan (2006); Saltelli et al. (2008).

In Chapter 4, we consider the Regional Hydro-Ecologic Simulation System (RHESSys) simulator and develop an emulator to efficiently approximate its output. In particular, we study how variations in soil properties affect RHESSys streamflow output at the watershed level. Soil descriptors for different soil properties are "soil inputs" when running RHESSys, and also used as inputs into our emulator (referred throughout as soil inputs/factors/properties). We further utilize the constructed emulator to conduct sensitivity analysis to study how the variation in RHESSys streamflow output can be attributed to different soil inputs, enabling us to identify the most influential soil factors (soil inputs/properties) amongst those in our study.

To build such an emulator, we need to first obtain reference or training runs of the RHESSys simulator. "Training data" from each RHESSys training run consists of a soil property setting (i.e., a vector containing a value for each of the soil inputs) that was input into RHESSys and the corresponding streamflow output resulting from running RHESSys. A collection of these training runs for a variety of soil input settings (the

input design for the RHESSys runs) enables us to estimate the parameters in the proposed statistics-based emulator. More specifically, we develop a Gaussian Process-based emulator, which provides modeling flexibility and straightforward inference procedures thanks to the favorable statistical properties of Gaussian Processes and the multivariate Gaussian distribution (Owen and Liuzzo (2019); Gu et al. (2016); Yang et al. (2018); Conti et al. (2009); Rajabi and Ketabchi (2017)).

For fast emulation of univariate time series computer model output using Gaussian Processes (GP), we follow the work proposed by Olson et al. (2018). In Chapter 4, we define Gaussian Processes and the *stilt* model in the context of our problem, including the mean and space-time separable covariance functions of the GP, and also extend the original *stilt* code to emulate seasonality in streamflow. The *stilt* method is fast in computation since it constructs the maximum likelihood estimates of emulator parameters, instead of utilizing a full Bayesian approach that simulates full posterior distributions through sampling algorithms such as Markov Chain Monte Carlo. Due to this benefit, emulating long time series is made possible in practice without much concern in computational power and running time. Using our application as an instance, the ten-year daily time series outputs are emulated and predicted within a reasonable amount of time, as illustrated in Section 4.4.

In addition to "Training Data" from training runs of RHESSys used to develop our emulator, we have "Testing Data" from additional runs of RHESSys (under collections of different values of the soil inputs) that we use to assess the approximation of our emulator output to that of RHESSys, described further in Section 4.3. For our development, the Tague Team Lab[3] provided us training and testing RHESSys runs for two watersheds with different meteorological and ecosystem dynamics: Sagehen Creek in the Sierra Nevada (Godsey et al. (2014); Tague and Peng (2013)) and Rattlesnake Canyon

[3]The Tague Team Lab: https://tagueteamlab.org/

near Santa Barbara (Hanan et al. (2017)). Both are dominated by Winter precipitation, but the former receives much of this in the form of snow, whereas Rattlesnake Canyon's precipitation is rain-dominated.

Due to the importance of emulation of complex mechanistic models, multiple types of emulators have been developed for time series output beyond the class we consider here. Notably there is a series of work based on neural networks that also builds on prosperous development of machine learning and artificial intelligence. For example, Scher (2018) builds a surrogate model of a general circulation model (GCM) for weather forecasting based on a deep convolutional neural network (CNN); and Kratzert et al. (2018) emulates the hydrological rainfall-runoff relationship with a long short-term memory (LSTM) network that is skilled at capturing long-term dependencies of the output. However, as argued by Reichstein et al. (2019), such neural network models remain hard to interpret and to conduct uncertainty and sensitivity analyses, although the modeling assumptions are rather flexible. We do not elaborate on this series of work and refer the interested readers to the referenced papers in that the statistical modeling and inference are our primary purposes.

Once an emulator is constructed, one can conduct sensitivity analysis, using emulator runs to approximate how components of variation in the simulator model output is associated with different input components/sources, as pointed out in Saltelli et al. (2008). Sensitivity analysis provides valuable insights in a number of settings as described in Saltelli et al. (2004). In particular, we are concerned with a combination of *Factor Prioritisation (FP)* setting and *Factor Fixing (FF)* setting. *FP* describes a setting where the identification of one or more input factors can reduce the variance of output to the largest extent. *FF* depicts a scenario where identification of one or more input factors, when varied over their ranges of distribution, does not influence the variance of the output. Combining both settings, we want to the identify the most-influential and non-influential

input components (in our case, amongst the input soil descriptors) in terms of reducing the variance in the output. We do not pursue two remaining settings here, namely the *Variance Cutting (VC)* and *Factor Mapping (FM)* settings, and refer the interested readers to Saltelli et al. (2004, 2008) for explanation.

Depending on the purpose of conducting sensitivity analysis, various approaches are available and can be performed for statistical inquiries. In our work, we are concerned with studying the sensitivity in the simulator (RHESSys) output, associated with multiple inputs and their interactions over their particular uncertain ranges (in our case over potential soil descriptor ranges), and identifying factors that contribute the most and the least to variation in the simulator results (i.e., in RHESSys output). This intention overlaps with the subject of global sensitivity analysis. In particular, variance-based methods which utilize variance to describe the uncertainty in the simulator output best match our purposes. Variance-based approaches are firstly employed by Cukier et al. (1973) in their method named *Fourier Amplitude Sensitivity Test (FAST)*, where they proposed to compute the first-order indices based on conditional variances. Sobol (1993) proposed to decompose the uncertainty in the model output into first-order and higher-order terms of uncertainty in the model inputs, which enables the computation of sensitivity indices to be completed based on Quasi-Monte-Carlo experiments. Quasi-Monte-Carlo methods greatly reduce the computational difficulty, by using simulation to approximate the variance-based sensitivity measures. In practice, commonly computed indices using variance-based methods include first-order (or main effects) indices, total-order indices, and second-order indices, as discussed in Saltelli et al. (2008, 2004); Puy et al. (2021); Homma and Saltelli (1996); Pianosi et al. (2016); Razavi et al. (2021); Saltelli et al. (2010, 2004). The first-order and second-order indices measure the amount of uncertainty in the model output that can be attributed to each model input and its two-way interactions. The total-order indices quantify the proportion of variation in the model output that can

be explained by terms involving each input at any order. Sobol (1993, 2001) provide a fundamental way of computing each sensitivity measure and the indices are often referred as Sobol's indices. Throughout the years, researchers have also developed variants for performing the calculation of the above-mentioned indices, for example, Saltelli et al. (2008, 2010); Jansen (1999); Azzini et al. (2020); Homma and Saltelli (1996).

Many readers may be familiar with another statistical term "Uncertainty Quantification" (UQ). However, we emphasize that the distinction between sensitivity analysis and uncertainty quantification is crucial. According to Saltelli et al. (2008), the aim of uncertainty quantification is to provide a probabilistic view of model output if certain inputs are unknown, whereas sensitivity analysis quantifies how variation in model output can be apportioned to different sources of the model inputs. The intention of our study is the latter as we are interested in how the different input values influence the time series output.

1.3.1 Contributions

The main contributions of this chapter are summarized in the following.

1. We present a Gaussian Process-based emulator for fast approximation of the time series output from a simulator which exhibits seasonality based on the method proposed in Olson et al. (2018).

2. In our global sensitivity analyses, we propose to compute the first-order, second-order, and total-order sensitivity indices for each time step, extending the conventional definition to have an additional time domain. This enables us to study how the variation in the model output can be explained by each input factor and their interactions at different times.

3. The comparison of emulating the same hydrological outputs at two experimental watersheds are provided and the results are discussed to assist the understanding of RHESSys behavior, thus aiding sampling design and calibration of the simulator.

4. The global sensitivity analysis based on the emulators are provided. The sensitivity of individual soil attributes, second-order interactions, and total-order effects are studied. Based on the analysis, the influential and non-influential soil properties for the variation in streamflow are identified for both experimental sites.

Chapter 2

Block-wise Robust Selection

This chapter presents our proposed method on using block-wise penalty parameters to tune Gaussian graphical models. We start with graphical lasso (glasso) for estimating inverse covariance matrix, whose objective function combines the Gaussian log-likelihood and an l_1 regularization term. Rather than having a scalar penalty parameter for the entire inverse covariance matrix, we propose to use block-wise regularization parameters that can adapt to different levels of data dispersion in the variables, while avoiding performing data normalization.

In this approach, we first group the variables in the dataset based on their variabilities. Then, we select a baseline block in the inverse covariance matrix for which we find the scalar regularization parameter using a modified version of the robust selection (RS) algorithm (Cisneros-Velarde et al. (2020)). Lastly, the penalty parameters for other blocks are computed by properly scaling the baseline-block penalty parameter, with coefficients derived in a data-dependent way. Using our proposed method, different penalty amounts can be assigned to different blocks of entries in the inverse covariance matrix. In addition, it is no longer needed to find a search range for candidate penalty parameters. This is converted to searching for a significance level within $[0, 1]$, making it easier to tune a Gaussian graphical model across different applications. Furthermore, since our method incorporates the RS algorithm, it inherits the computational efficiency.

The organization of this chapter is as follows. In Section 2.1, we elaborate on our block-wise robust selection method. First, we describe the construction of the block-wise penalty parameters when the variable grouping is given. Second, a method based on change point detection is discussed for determining the variable grouping. A criterion proposed in change point detection literature selects the number of variable groups. Lastly, this section provides an alternative approach for choosing the block-wise regularization parameters. In Section 2.2, we provide the numerical experiments using our methods based on three simulated scenarios. The comparison with two robust selection variants is included in this numerical study. Two paleo-climate reconstruction applications using our tuning method are deferred until the next chapter.

2.1 Methodology

In this section, we describe our proposed block-wise robust selection (BRS). First, we discuss how to choose the regularization parameters given a grouping of the variables. Then, we discuss one way of obtaining the variable groups via binary segmentation. Lastly, we explain how to decide on the number of variable groups.

When identifying the zeros and nonzeros in the inverse covariance matrix (model selection), a scalar penalty parameter is used in most methods (e.g., Friedman et al. (2008); Peng et al. (2009); Khare et al. (2015)) and the variables are normalized. As a result, the same amount of regularization is applied to all elements in the inverse covariance matrix. However, estimating the inverse covariance matrix after normalization distorts the parameter values and renders the inferred parameter estimates unusable in downstream analysis. On the other hand, our method aims to find regularization parameters adaptively for different blocks of edges with different variance levels, avoiding data normalization.

In order to take the data dispersion into account and avoid data normalization, we propose to regularize the partial correlations with the same parameter $\tilde{\lambda}$. This resulting penalty term can be expressed as $\tilde{\lambda}|\rho_{ij}|$. Given the definition of partial correlations, i.e., $\rho_{ij} = -\frac{K_{ij}}{\sqrt{K_{ii}K_{jj}}}$, the penalty term can be rewritten as $\frac{\tilde{\lambda}}{\sqrt{K_{ii}K_{jj}}}|K_{ij}|$. In graphical lasso, $|K_{ij}|$ is penalized with λ_{ij} which can be expressed in terms of $\tilde{\lambda}$ as

$$\lambda_{ij} = \frac{\tilde{\lambda}}{\sqrt{K_{ii}K_{jj}}}.$$

As such, we can construct a penalty matrix whose entries are λ_{ij} for $i, j \in \{1, \ldots, p\}$, as follows:

$$\Lambda = \tilde{\lambda} \begin{pmatrix} \frac{1}{K_{11}} & \frac{1}{\sqrt{K_{11}K_{22}}} & \cdots & \frac{1}{\sqrt{K_{11}K_{pp}}} \\ \frac{1}{\sqrt{K_{22}K_{11}}} & \frac{1}{K_{22}} & \cdots & \frac{1}{\sqrt{K_{22}K_{pp}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{K_{pp}K_{11}}} & \frac{1}{\sqrt{K_{pp}K_{22}}} & \cdots & \frac{1}{K_{pp}} \end{pmatrix},$$

where $\tilde{\lambda}$ is the baseline regularization parameter and K_{ii} is the partial variance of the i^{th} variable, for $i \in \{1, \ldots, p\}$. This matrix can be formulated using an outer product as

$$\Lambda = \tilde{\lambda} v v^{\top}, \tag{2.1}$$

where $v = (\frac{1}{\sqrt{K_{11}}}, \ldots, \frac{1}{\sqrt{K_{pp}}})^{\top}$. Such a formulation of the regularization is useful for understanding inverse covariance estimation but the quantities can not be decided prior to the estimation procedure, as K is the unknown matrix that we are estimating.

Nevertheless, following such an idea of formulating the penalty parameter(s), we explain how we can obtain a block-wise penalty matrix. Specifically, the same penalty parameter is shared among entries corresponding to variables with a similar level of variances.

2.1.1 Choosing Regularization Parameters When Variable Grouping is Given

This part explains the block-wise regularization, assuming the variable grouping is given. The discussion on obtaining such a grouping is deferred in the later sections. Suppose that all variables can be partitioned into N groups, where $N \in \{1, \ldots, p\}$. Consequently, the inverse covariance matrix K and the penalty parameter matrix Λ will have N diagonal blocks and $(N-1)N/2$ upper off-diagonal blocks. In addition, we use I and J to denote the I^{th} and J^{th} variable group, where $I, J \in \{1, \ldots, N\}$. The number of variables in the J^{th} group is represented by p_J and $p = \sum_J p_J$. We then use a pair of upper-case letters (I, J) to index a block in a matrix, where the rows correspond to variables from group I and the columns from group J. The dimension of the sub-matrix indexed by (I, J) is thus $p_I \times p_J$. Consider a running example where all variables can be partitioned into $N = 2$ groups. In this example, the two diagonal blocks are indicated by $(1, 1)$ and $(2, 2)$, and the one off-diagonal block is indexed by $(1, 2)$.

Given a variable grouping, our goal is to find the penalty parameter matrix Λ that exhibits an adaptive amount of regularization for each block of edges. This is made possible by finding the scalar penalty parameters for each block of entries in the inverse covariance matrix and combining them appropriately. The scalar penalty parameter is the same for entries within each block, based on the assumption that the data dispersion of the variables being grouped together is similar. On the other hand, the regularization parameters are different for different blocks. Embedding this block-wise idea into Equation (2.1), we construct the block-wise penalty matrix as follows

$$\Lambda = \tilde{\lambda} v v^\top, \tag{2.2}$$

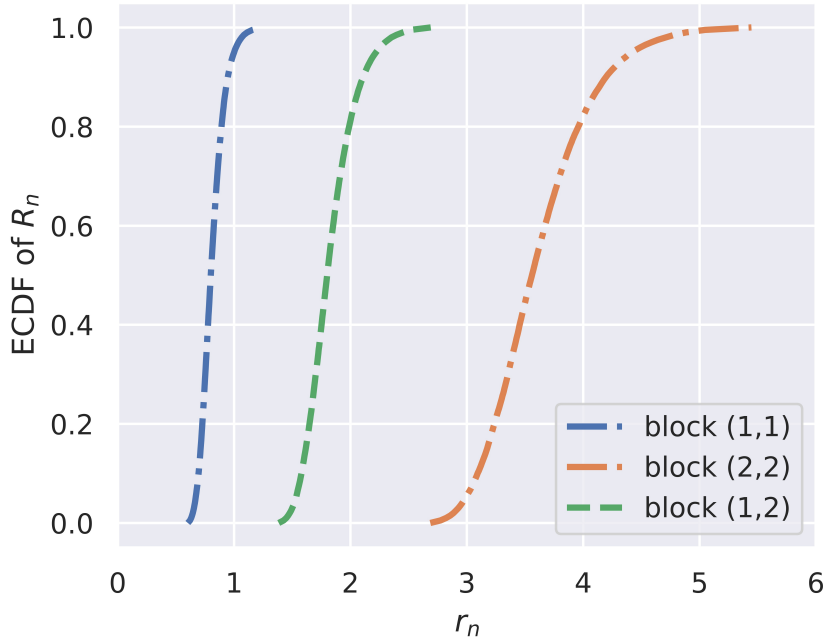

Figure 2.1: RWP functions for the running example where the variables are grouped into two blocks. In this case, there are two diagonal blocks and one off-diagonal block in the precision matrix.

where $vv^\top$ can be treated as a weight matrix scaling the baseline parameter $\tilde{\lambda}$ differently according to different blocks. Specifically, v is a p-dimensional vector with repeated entries for each group J, that is

$$v = (\underbrace{\gamma_1, \ldots, \gamma_1}_{p_1}, \ldots, \underbrace{\gamma_J, \ldots, \gamma_J}_{p_J}, \ldots, \underbrace{\gamma_N, \ldots, \gamma_N}_{p_N})^\top \in \mathbb{R}^p.$$

We use γ_J to represent the median of $\{1/\sqrt{K_{ii}}\}$ across all variable i in group J. Moreover, γ_J can be approximated by the median of $\{\sqrt{\Sigma_{ii}}\}$ across all variable i in group J, since Σ is the inverse of K if it is a diagonal matrix. Now the problem boils down to finding $\tilde{\lambda}$.

The selection of $\tilde{\lambda}$ is completed by the robust selection algorithm for a baseline block which needs to be decided before obtaining $\tilde{\lambda}$. In our implementation, we first obtain the

robust Wasserstein profile (RWP) functions for all the blocks and use the diagonal block whose median value of the RWP function is maximal as the baseline.[1] Suppose block (J, J) is selected as the baseline. We write its RWP function as $R_{n,(J,J)}$. Then, $\tilde{\lambda}$ is set to be the $\lfloor (B+1)(1-\alpha) \rfloor^{th}$ order statistic of $R_{n,(J,J)}$ divided by γ_J^2, i.e.,

$$\tilde{\lambda} = R_{n,(J,J)}^{\lfloor (B+1)(1-\alpha) \rfloor} / \gamma_J^2.$$

While calculating the RWP functions for any diagonal block, we discard the diagonal elements from the corresponding sub-matrix of A and K^{-1}.

With the proposed method, the arbitrary searching space of the regularization parameter $\tilde{\lambda}$ is restricted to a fixed range of the significance level parameter α, which spans from zero to one. Specifying the searching grid for the regularization parameter in methods like cross-validation requires expert knowledge to include an optimizer of certain out-of-sample metric, which is challenging and computationally inefficient. In contrast, adjusting α from the most conservative (zero) to the most flexible (one) makes it easy for practitioners to tune a Gaussian graphical model. This range is unchanged regardless of the magnitude and unit of the data. We refer the interested readers to the original RS paper for an exact definition of α and how it is used as a tolerance level in the distributionally robust optimization.

We illustrate the block-wise RWP functions computed for our running example. Figure 2.1 shows the respective RWP functions of the three blocks. The RWP functions for the two diagonal blocks are shown in blue and orange, and the RWP function for the off-diagonal block is shown in green. It can be seen that these RWP functions have different scales. In this case, we choose block $(2, 2)$ as the baseline block as its median value of the RWP function is the largest. The block-wise penalty matrix is computed by

[1] See Theorem 1.1 in Chapter 1 for the definition of robust Wasserstein profile (RWP) function based on A and K^{-1}.

Equation (2.2) with $\tilde{\lambda} = R_{n,(2,2)}^{\lfloor (B+1)(1-\alpha) \rfloor} / \gamma_2^2$.

2.1.2 Determining Variable Grouping

Given a set of variables, to apply block-wise robust selection, it is necessary to first assign them into suitable groups, based on their statistical characteristics (e.g., variance magnitudes). In this section, we use the binary segmentation algorithm from the change point detection literature to group the variables. We first assume that the target number of variable groups is provided in this section and will discuss how to determine this number in the next section.

Grouping a set of variables can be treated as a clustering problem, where variables in the same group should share similar characteristics. In contrast, variables from different groups should be more different. As BRS avoids normalization to preserve the original characteristics of the data, the desired clustering method should take the different variances of the variables into account to facilitate the selection of suitable penalty parameters. Since the data are assumed to be zero-mean throughout this chapter, we defer the handling of the heterogeneous means of variables until the next chapter.

One possible way to achieve this is to detect the abrupt changes in the sorted standard deviations of the variables. In other words, variables with similar variabilities should be assigned to the same group while those with different variabilities should be in different groups. This is similar to the multiple change-point detection problems in the literature (Scott and Knott (1974); Truong et al. (2020); Killick et al. (2012); Killick and Eckley (2014)).

Suppose the ordered sequence of sample standard deviations of the variables is denoted as $s_{(1)}, ..., s_{(p)}$. Given the target number of variable groups N, the required number of change points is $N - 1$. Let $\tau_i \in \{1, 2, \ldots, p-1\}$, where $i \in \{0, ..., N-1\}$, denote

the location of the i^{th} change point to be found. We further denote the location of the sequence starting point as $\tau_0 = 0$ and that of the ending point as $\tau_N = p$. Given the change points, the sequence is segmented into N chunks, with the change points at locations $\tau_1, \ldots, \tau_{N-1}$. Note that the change points are ordered, which means that $\tau_i < \tau_j$ if and only if $i < j$. The multiple change-point detection for grouping the variables can then be formulated as the following optimization problem:

$$\underset{\tau_1,\ldots\tau_{N-1}}{\operatorname{argmin}} \quad \sum_{i=1}^{N} \ell(s_{\left((\tau_{i-1}+1):\tau_i\right)}), \tag{2.3}$$

where $\ell(\cdot)$ is a loss function that measures the quadratic differences between consecutive points in a sub-sequence and $s_{\left((\tau_{i-1}+1):\tau_i\right)}$ denotes the sample standard deviations from location $\tau_{i-1} + 1$ to location τ_i.

To solve this optimization, we use the binary segmentation algorithm (Scott and Knott (1974)) from the change point detection literature. This is a conceptually simple yet effective method to detect one or multiple change points in the input sequence (Killick et al. (2012); Killick and Eckley (2014); Truong et al. (2020)). The sequence is greedily split by finding a new breaking point that maximally lowers the loss function and the procedure is performed until $N - 1$ change points are found. This is an approximate minimization process of Problem (2.3) as the detection of any new change location is conditional on the previously identified change points (Scott and Knott (1974); Truong et al. (2020); Killick and Eckley (2014)). The computation complexity for determining the variable grouping is $\mathcal{O}(p \log p)$ when using binary segmentation. We use the Python package *ruptures* (Truong et al. (2020)) for running the binary segmentation with a quadratic loss function.

2.1.3 Choosing the Number of Variable Groups

In this part, we discuss how to choose the number of groups, N, for the change point detection problem described previously. Multiple metrics have been proposed in the literature (e.g., Truong et al. (2020)) and widely used in practice (e.g., Killick and Eckley (2014)) to facilitate choosing the number of change points. Following the general guidelines of model selection, these metrics aim to balance between model flexibility and generalizability. With a fixed metric, the procedure for selecting N is given as follows. We first compute the segmentation for a sequence of values. Then, the segmentation quality is measured by this given metric as a function of N. Lastly, the N value that optimizes the metric is chosen as the the final decision. In our work, we utilize the Modified Bayes Information Criterion (MBIC) proposed by Zhang and Siegmund (2007) for deciding N, which maximizes the asymptotic approximation of Bayes factor.

When using MBIC, the data to be segmented is assumed to be Gaussian. In our context, we have

$$s_{(j)} \sim \mathcal{N}(\mu_i, 1), \text{ for } j \in \{\tau_i + 1, \ldots, \tau_{i+1}\}, \text{ and } i \in \{0, ..., N-1\},$$

where μ_i, $\forall i \in \{0, ..., N-1\}$ reflects the different mean-levels of the signal. Next, we provide the definition of MBIC in the following theorem.

Theorem 2.1 *(Theorem 1 in Zhang and Siegmund (2007)) Suppose $N-1$ change points are to be detected and the quadratic loss is used. Then, given that other assumptions are*

satisfied,

$$MBIC = \frac{1}{2}\sum_{i=1}^{N}(\tau_i - \tau_{i-1})\big(\frac{1}{\tau_i - \tau_{i-1}}\sum_{j=\tau_{i-1}}^{\tau_i - 1} s_{(j)} - \frac{1}{p}\sum_{j=1}^{p} s_{(j)}\big)^2 - \frac{1}{2}\Big(3(N-1)\log(p) + \sum_{i=1}^{N}\log(\frac{\tau_i - \tau_{i-1}}{p})\Big),$$

where the terms in the first row specify the log-likelihood of the data and those in the second are the penalty terms.

MBIC is similar to the traditional BIC metric because it contains a data likelihood term and a penalty term. However, the penalty term in MBIC uses a more complicated way to regularize the model complexity. As we can see, the penalty term reaches the maximum when τ_i's are evenly spaced. Thus, one characteristic of MBIC is that it favors evenly spaced change points. For more information regarding MBIC and its applications to genomic data, we refer readers to Zhang and Siegmund (2007).

2.1.4 Algorithm

The procedure of our proposed method, BRS, is summarized in Algorithm 1.

2.1.5 Alternative Way of Choosing Regularization Parameters

In this part, we describe an alternative approach for obtaining the block-wise regularization matrix by utilizing the RS algorithm, to which we refer as BRS*. In our main BRS algorithm discussed previously, we run RS once for the baseline block, and use the order statistic as well as γ_J's for constructing the block-wise penalty matrix. Unlike this way, in BRS*, we treat the blocks of edges independently and run the RS algorithm for

[2] $A_{n,(I,J)}$ stands for the sub-matrix of A where the rows consist of variables from the I^{th} group and the columns represent variables from the J^{th} group.

Algorithm 1: Block-wise regularization selection for Gaussian graphical model

Data: $X_{n\times p}$
Require: N groups of variables, B, α
Result: Λ
for *b in {1,..., B}* **do**
  Obtain X^b by sampling rows of X with replacement;
  Compute the covariance matrix A_n^b for X^b;
  for *block* (I, I) *where* $I \in \{1, \ldots, N\}$ **do**
    Compute bootstrap RWP function $R^b_{n,(I,I)} = \|\mathbf{vec}\ (A^b_{n,(I,I)} - A_{(I,I)})\|_\infty$;[2]
  end
end
Determine a baseline block (J, J) and obtain the order statistic $R^{\lfloor (B+1)(1-\alpha)\rfloor}_{n,(J,J)}$;
Set $\tilde{\lambda}$ to be $R^{\lfloor (B+1)(1-\alpha)\rfloor}_{n,(J,J)}/\gamma_J^2$;
Construct the block-wise penalty matrix Λ by Equation (2.2).

each block in order to get their respective penalty parameters. More precisely, the block-wise RWP function $R_{n,(I,J)}$ is constructed for any given block (I, J) $(I, J \in \{1, \ldots, N\})$. Then, the $\lfloor (B+1)(1-\alpha)\rfloor^{th}$ order statistic of the RWP function is returned as the regularization parameter for this block. Lastly, we construct the entire penalty matrix by applying the obtained scalar penalty parameter for all entries in each block (I, J).

2.2 Numerical Experiments

In this section, we conduct simulation experiments to evaluate the performance of the proposed method BRS. We compare BRS with the alternative block-wise approach BRS*, and with two variants of robust selection, RS1 and RS2, in terms of graph recovery. The difference between the two variants of robust selection is whether the data is normalized before parameter tuning and inverse covariance estimation. More specifically, no data normalization is performed in BRS, BRS* and RS1, whereas the column mean and variance of the data are removed in RS2. We summarize these four approaches in Table 2.1. Once the penalty parameter is determined, we use the QUIC algorithm (Hsieh et al. (2013))

Table 2.1: Simulation setups

Abbreviation	Tuning Method	Data Normalization
BRS	block-wise robust selection	no
BRS*	alternative way of BRS	no
RS1	robust selection	no
RS2	robust selection	yes

for estimating the inverse covariance matrix.[3]

2.2.1 Simulation Settings and Data

We conduct three simulation experiments with two kinds of underlying random graphs, i.e., a power-law graph and an Erdős-Rényi graph. The power-law graph is often considered as the underlying structure for many real-world graphs, such as social and genetic networks. The Erdős-Rényi graph is a commonly used random graph for simulation studies on inverse covariance estimation. The first and second simulation use the power-law graph and the third one uses the Erdős-Rényi graph. We refer to these three simulations as the power-law I, the power-law II, and the Erdős-Rényi simulations, respectively. The power-law graph has a varying distribution for the degree of each node, where the degree of a node is defined as the number of edges connected to it. In particular, the degree distribution follows a power law. The Erdős-Rényi graph is constructed by randomly including edges to connect the nodes. The probability for any edge inclusion is set to be 3%. Figure 2.2 shows the generated power-law graph and Erdős-Rényi graph with edge densities 5.85% and 3.22%, respectively.

Given a graph structure, the edge weights are sampled following the method in Peng et al. (2009) and consequently, we obtain a $p \times p$ correlation matrix Σ^{init}, the inverse of which K^{init} obeys the given graph structure. Next, we pre- and post-multiply K^{init} by a diagonal matrix D and denote the resulting matrix as K. In the Erdős-Rényi

[3]QUIC stands for QUadratic approximation of Inverse Covariance matrices.

and power-law I simulations, the non-zero entries in D ranges from 0.2 to 1.4 in an increasing order, while in the power-law II simulation, these entries follow a decreasing order. Finally, for each sample size $n \in \{0.5, 1, 2, 5\} \times p$, we sample $m = 100$ $i.i.d.$ datasets following the multivariate normal distribution with mean zero and covariance $\Sigma = K^{-1}$, i.e., $\mathcal{N}_p\left(0, K^{-1}\right)$. In all three simulations, the number of variables is $p = 200$ and the number of bootstrapped replicates is $B = 200$. Four significance levels are considered: $\alpha \in \{0.05, 0.35, 0.65, 0.95\}$.

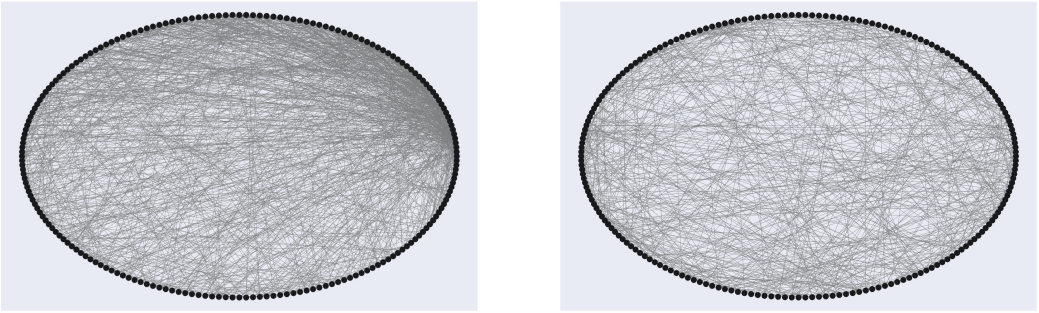

Figure 2.2: Graphs used in our simulation study: power-law graph (left) and Erdős-Rényi graph (right). Their respective edge densities are 5.85% and 3.22%.

2.2.2 Performance Metrics

The model selection performance of BRS, BRS*, RS1, and RS2 is assessed based on the correctness of the recovered graph structures. Since the underlying graphs are sparse, we adopt F_1 and Matthews correlation coefficient (MCC) as the evaluation metrics, both of which are informative measures for summarizing imbalanced classification results (see Chicco and Jurman (2020); Boughorbel et al. (2017)). Next, we provide formal definitions of F_1 and MCC. First, we denote the four entries of a confusion matrix as follows: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). In graph

estimation, we consider the presence of an edge as a "positive" instance and the absence of an edge as a "negative" case.

- Matthews correlation coefficient (MCC) is defined as

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

  and ranges from -1 to 1. It uses the four quantities from the confusion matrix, and essentially is Pearson correlation between the true and estimated binary classifications of the edge presence. The highest value 1 indicates that the graph estimation is perfect. The lowest value -1 suggests a total disagreement between the actual and estimated structures, and 0 means that the estimation is no better than flipping a coin. MCC becomes undefined if the estimated graph is empty, namely, when TP and FP are zero.

- F_1 score is defined as

$$F_1 = \frac{TP}{TP + 1/2(FP + FN)}$$

  and ranges from 0 to 1. It is the harmonic mean of precision and recall when they are considered to be equally important. F_1 score reaches the its best value at 1 when the estimation of graph structure is perfect and the worst value at 0 when either precision or recall is zero. One criticism of the score is that it ignores TN, but in cases where the actual and estimated edges are of greater importance, F_1 still provides useful information in addition to MCC.

2.2.3 Power-Law I Simulation

We discuss the results of power-law I simulation in this part. First, we compare the true and the estimated adjacency matrices in Figure 2.3, when $\alpha = 0.05$ and $n/p = 5$.

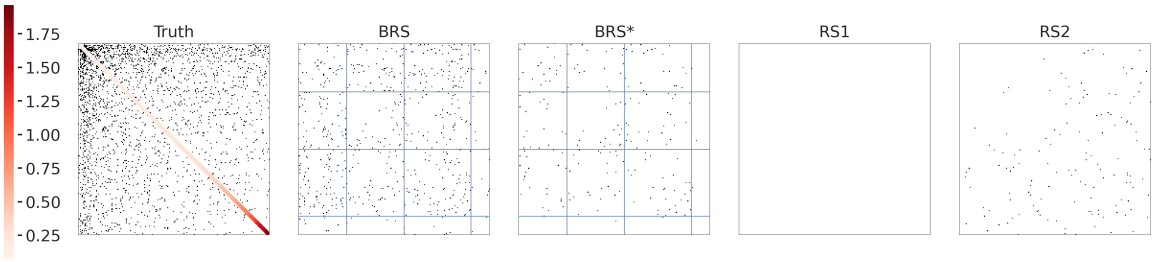

Figure 2.3: Adjacency matrices (from left to right) of the true graph, BRS graph, BRS* graph, RS1 graph, and RS2 graph in the power-law I simulation, with $\alpha = 0.05$ and $n/p = 5$. The graphs are estimated based on the same Gaussian dataset. The colorbar indicates the magnitudes of the actual partial variances. The grids in BRS and BRS* show the variable grouping with the number of blocks being four. Regularization parameters are tuned by BRS, BRS* and RS. Un-normalized data is used for BRS, BRS*, and RS1, and normalized data is used for RS2. The inverse covariance matrix is estimated using QUIC.

The true adjacency matrix is provided in the left-most figure, whose diagonal line is colored by the magnitude of the true partial variances. A lighter (darker) red color indicates a lower (higher) magnitude. The top-left corner in the true adjacency matrix shows higher node degrees than the bottom-right. The second, third, fourth, and fifth sub-figures (from left to right) show the four adjacency matrices estimated by BRS, BRS*, RS1, and RS2, respectively. The blue grids in the second and third sub-figures show the edge blocks in BRS and BRS*, respectively. These four graphs are estimated based on one of the *i.i.d.* datasets generated for power-law I simulation.

We see that the adjacency graph estimated by BRS is visually the most similar to the true graph, in terms of the quantity and locations of the edges. This is followed by BRS*, where the number of edges is less than that of BRS. In contrast, RS1 graph does not contain any edge and RS2 graph contains a relatively small number of edges and misses those in the top-left area where the true node degrees are high. The relative comparison of the four methods is consistently observed across the *i.i.d.* datasets as well as in other cases where α and n/p take different values.

Second, the four methods are compared quantitatively using MCC and F_1. Figure 2.4 and 2.5 show the boxplots of MCC and F_1 for power-law I simulation. Four α values are considered and the corresponding results shown in the sub-figures are (a) $\alpha = 0.05$, (b) $\alpha = 0.35$, (c) $\alpha = 0.65$, and (d) $\alpha = 0.95$. For each α value, four n/p ratios are used, as indicated by the colors in the sub-figures. In each sub-figure, the top row shows the boxplots for BRS, the second row for BRS*, the third row for RS1, and the bottom row for RS2. Boxes towards the right indicate higher F_1 and MCC scores.

In terms of both metrics, we see that BRS attains higher F_1 and MCC scores than the other methods, for all values of α and n/p. For example, when $\alpha = 0.95$ and $n/p = 5$, the median MCC of BRS is 0.59, and those of BRS*, RS1, and RS2 are 0.53, 0.13, and 0.49, respectively. The higher MCC and F_1 scores attained by BRS and BRS* result from that both methods allow more flexible regularization that is adaptive to different levels of variance of the variables. In contrast, the heterogeneous variances of the variables can not be taken care of by RS1, which in turn results in empty graphs. For instance, when $\alpha = 0.05$ and $n/p \leq 2$, no edges can be recovered by RS1 since the tuned penalty parameter is too large. The MCC and F_1 scores are undefined for such cases and thus the corresponding boxes are missing in Figure 2.4 (a) and 2.5 (a). In these two figures, the squeezed boxes for $n/p = 5$ are due to that only a small portion of datasets have non-empty graphs with very few edges. While RS2 can handle the heterogeneous variances, this process requires normalization which distorts data distribution, e.g., when the sample variances are not accurate. As such, RS2 has higher scores than RS1 but performs worse than BRS, as can be seen in the four sub-figures for each metric.

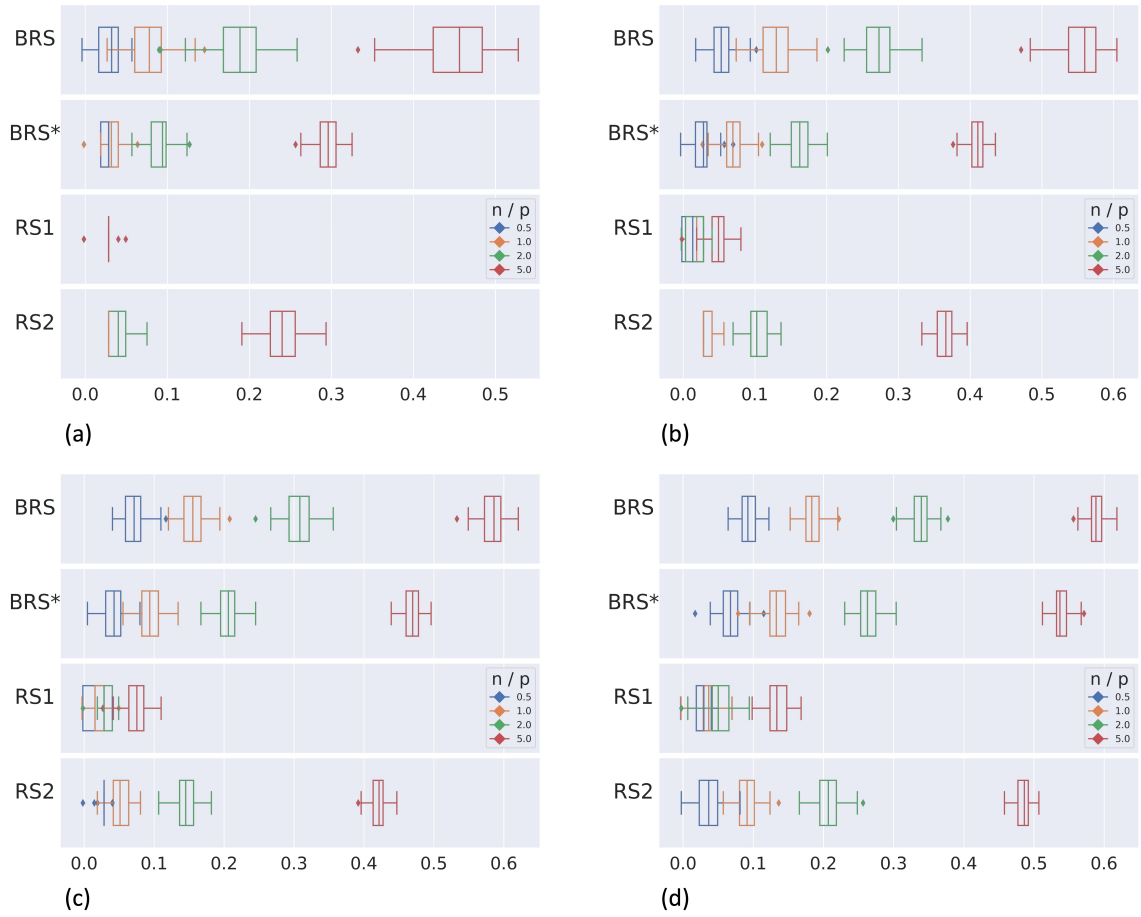

Figure 2.4: MCC boxplots of graph structure estimation in the power-law I simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the MCC of block-wise robust selection results with un-normalized data, the second row shows the MCC of alternative block-wise selection results with un-normalized data, the third row shows the MCC of robust selection results with un-normalized data, and the bottom row shows the MCC of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

2.2.4 Power-Law II Simulation

The results of power-law II simulation is provided in this part. The difference between this and power-law I simulation is the way to obtain K, as described in Section 2.2.1. Consequently, the magnitudes of the true partial variances in this case are in a decreasing

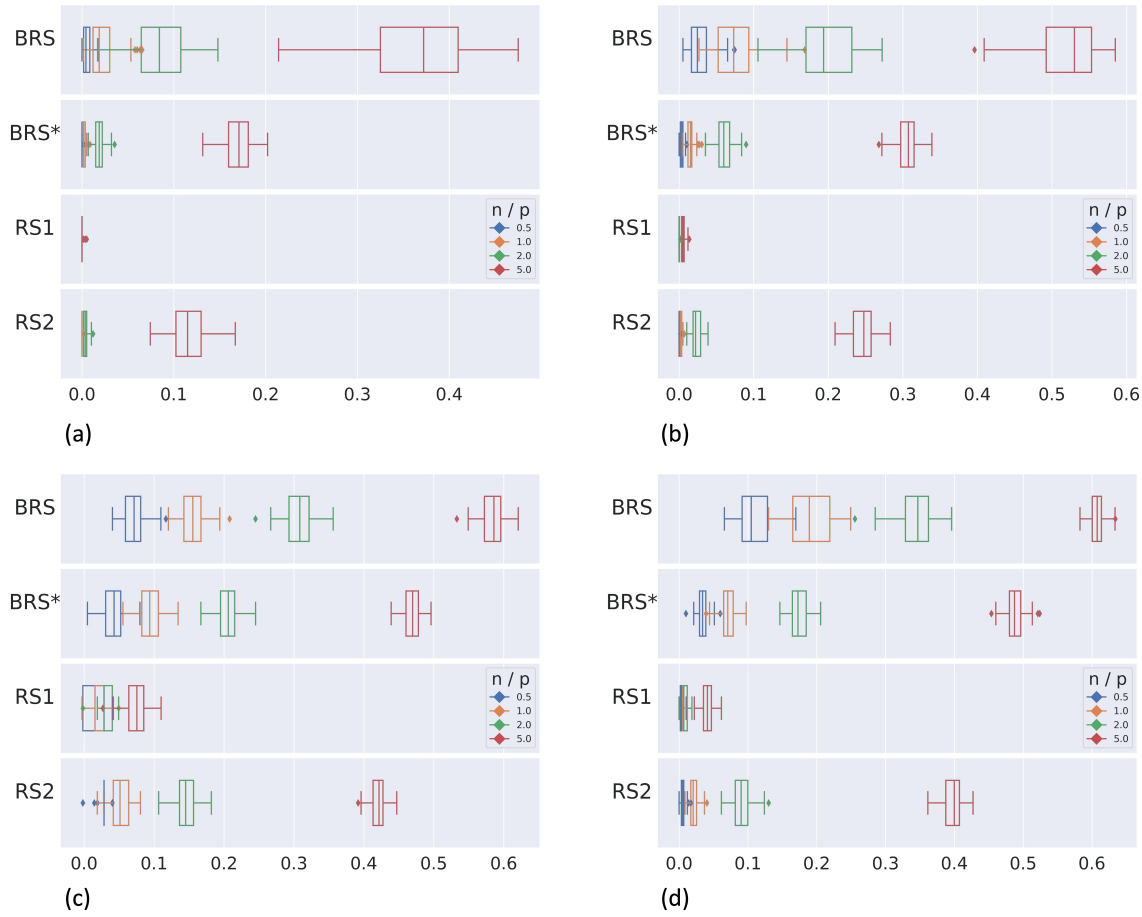

Figure 2.5: F_1 boxplots of graph structure estimation in the power-law I simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the F_1 of block-wise robust selection results with un-normalized data, the second row shows the F_1 of alternative block-wise selection results with un-normalized data, the third row shows the F_1 of robust selection results with un-normalized data, and the bottom row shows the F_1 of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

order, unlike in power-law I. This distinction can be directly seen in Figure 2.6, where we compare the true and the estimated adjacency matrices when $\alpha = 0.05$ and $n/p = 5$. The true adjacency matrix is presented in the left-most panel, whose diagonal line is colored by the magnitude of the true partial variances. A darker (lighter) red color shows a

higher (lower) magnitude. The remaining panels successively show the four estimated adjacency matrices by BRS, BRS*, RS1, and RS2. The blue grids in the second and third sub-figures indicate the edge blocks in BRS and BRS*, respectively. These four estimated graphs are obtained based on one of the *i.i.d.* datasets generated for the power-law II simulation. Visually, the number and locations of edges obtained via BRS are the most similar to the ground truth. Those for the BRS* are less similar to the ground truth. The RS1 graph only contains a very small number of edges in the bottom-right corner and the RS2 graph shows fewer edges than BRS. The patterns of the graph recovery of the four methods are similar across the *i.i.d.* datasets and in other cases where α and n/p are varied.

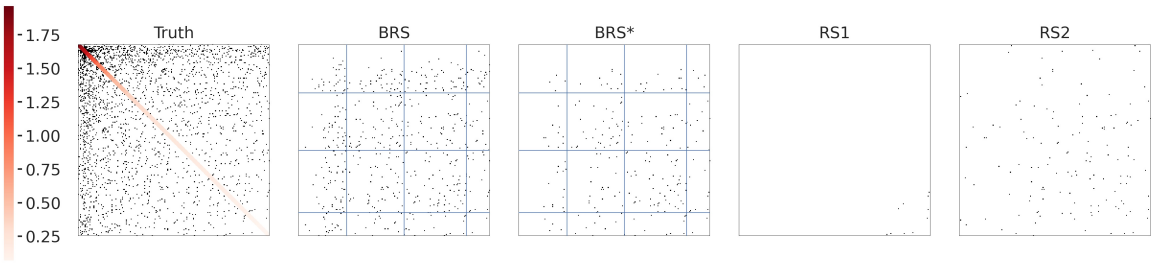

Figure 2.6: Adjacency matrices (from left to right) of the true graph, BRS graph, BRS* graph, RS1 graph, and RS2 graph in the power-law II simulation, with $\alpha = 0.05$ and $n/p = 5$. The graphs are estimated based on the same Gaussian dataset. The colorbar indicates the magnitudes of the actual partial variances. The grids in BRS and BRS* show the variable grouping with the number of blocks being four. Regularization parameters are tuned by BRS, BRS* and RS. Un-normalized data is used for BRS, BRS*, and RS1, and normalized data is used for RS2. The inverse covariance matrix is estimated using QUIC.

In addition to the visual inspection of edge recovery, we provide quantitative comparison of the three methods using MCC and F_1 in Figure 2.7 and 2.8. The two figures show the boxplots of MCC and F_1 for power-law II simulation, respectively. Four α values are considered and the corresponding results shown in the sub-figures are (a) $\alpha = 0.05$, (b) $\alpha = 0.35$, (c) $\alpha = 0.65$, and (d) $\alpha = 0.95$. Four n/p ratios are considered for each

α value, which is indicated by the colors in the sub-figures. In each sub-figure, the top row shows the boxplots for BRS, the second row from the top for BRS*, the third row for RS1, and the bottom row for RS2. Boxes towards the right indicate higher F_1 and MCC scores.

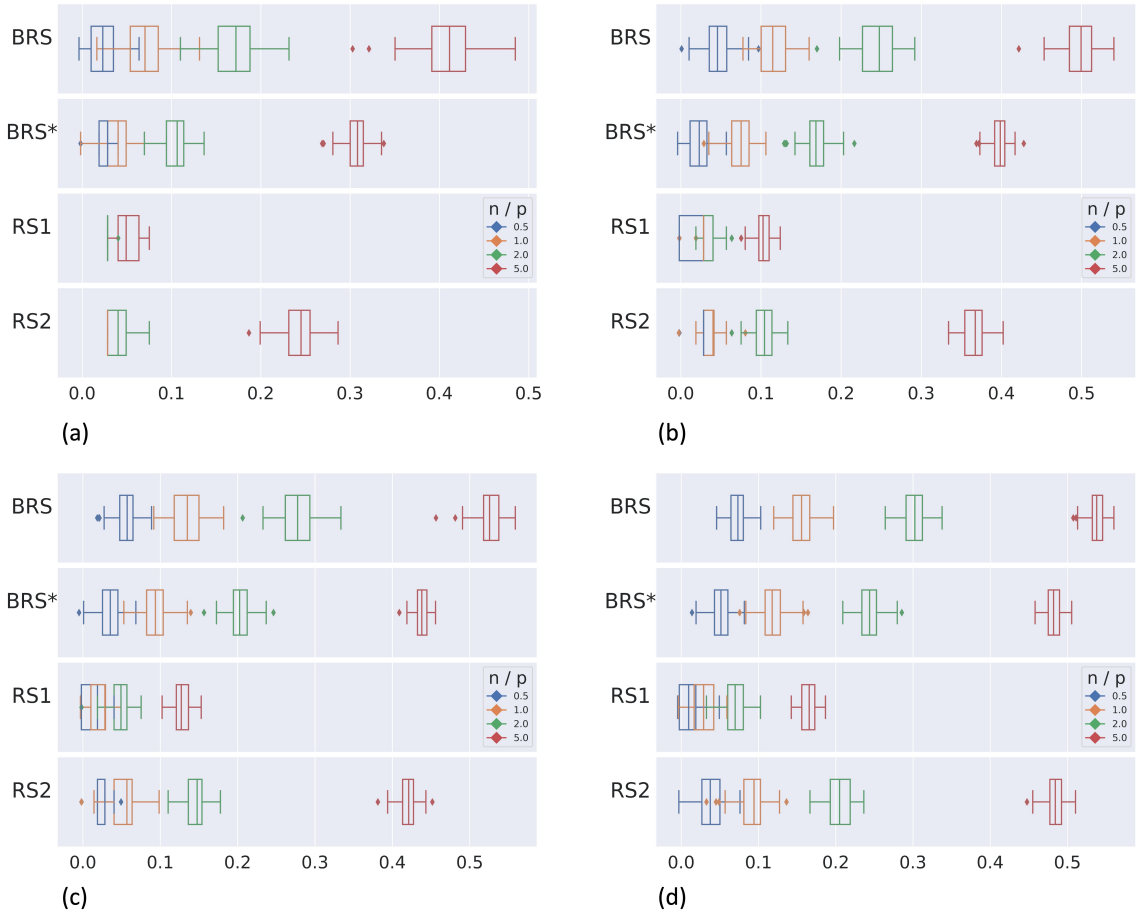

Figure 2.7: MCC boxplots of graph structure estimation in the power-law II simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the MCC of block-wise robust selection results with un-normalized data, the second row shows the MCC of alternative block-wise selection results with un-normalized data, the third row shows the MCC of robust selection results with un-normalized data, and the bottom row shows the MCC of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

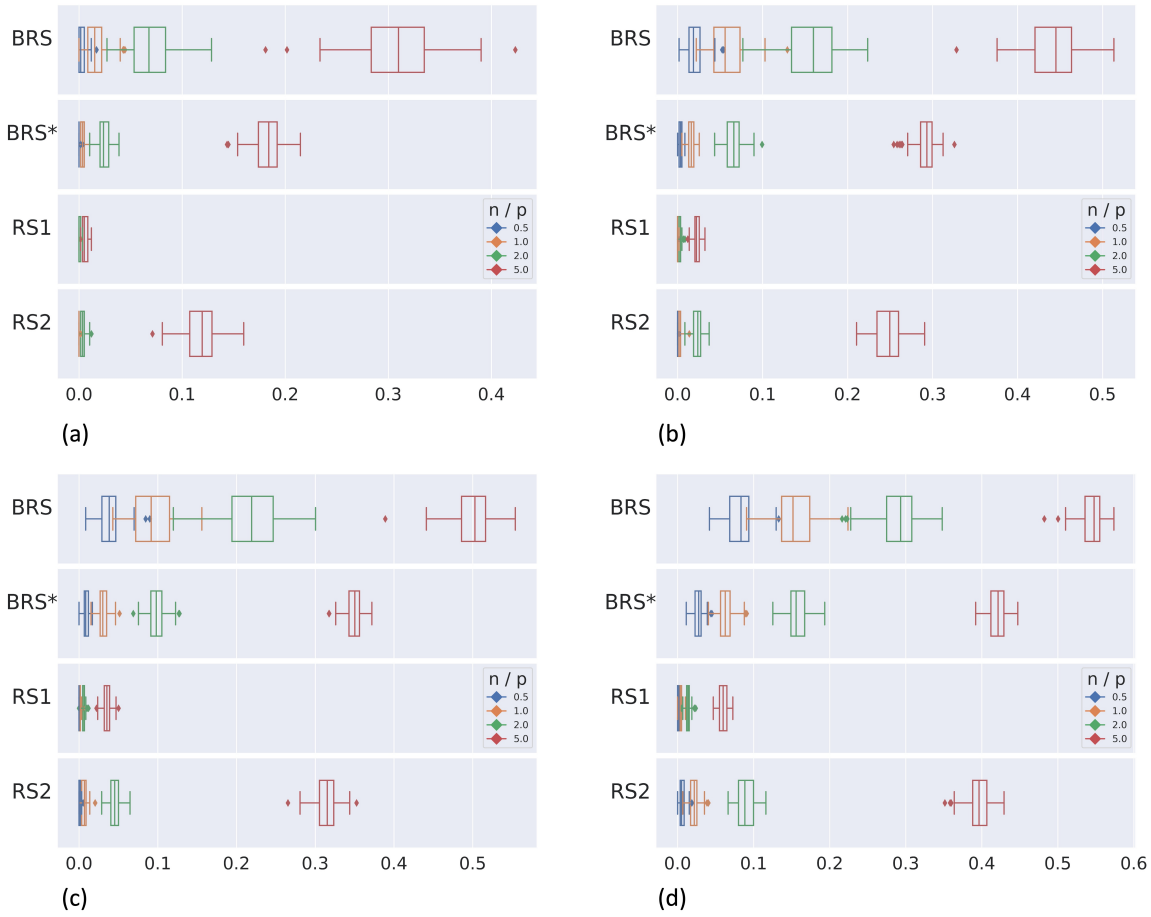

Figure 2.8: F_1 boxplots of graph structure estimation in the power-law II simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the F_1 of block-wise robust selection results with un-normalized data, the second row shows the F_1 of alternative block-wise selection results with un-normalized data, the third row shows the F_1 of robust selection results with un-normalized data, and the bottom row shows the F_1 of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

It is observed that BRS reaches larger MCC and F_1 scores than the other three methods, for all values of α and n/p. When α is controlled at 0.35 and n/p is 5, we report the median F_1 score for BRS, BRS*, RS1 and RS2, which are 0.45, 0.29, 0.02 and 0.25, respectively. Since the penalty parameters tuned by BRS can adapt to different

levels of variances present in the data, the estimated edges are more accurate. In contrast, RS1 does not take account of the heterogeneous variances, thus the obtained graphs are empty in several cases due to overly heavy regularization. Such graphs have undefined MCC and F_1 scores, which causes the absence of the corresponding boxes in Figure 2.7 and 2.8. For example, when $\alpha = 0.05$ and $n/p \leq 2$, no edges can be recovered by RS1. Thus, the corresponding boxes are missing in Figure 2.4 (a) and 2.5 (a). RS2 deals with the heterogeneous variances of data by column-wisely removing them and thus, consistently reaches higher MCC and F_1 scores than RS1. However, this normalization procedure changes the original distribution of data and is sensitive to the accuracy of the sample variances. It is thus not as robust as BRS. Consequently, the MCC and F_1 scores are lower than those of BRS.

2.2.5 Erdős-Rényi Simulation

In this part, we discuss the results of the Erdős-Rényi simulation. First, we compare the true and the estimated adjacency matrices in Figure 2.9, when α is controlled at 0.05 and n/p is 5. The left-most sub-figure is the true adjacency matrix where the brightness of the diagonal line varies based on the magnitude of the true partial variances. A darker (lighter) red color indicates a higher (lower) magnitude. The second, third, fourth, and fifth sub-figures (from left to right) show the four estimated adjacency matrices by BRS, BRS*, RS1, and RS2, respectively. In the second and third sub-figures, the blue grids indicate the edge blocks in BRS and BRS*, respectively. These four graphs are estimated based on one of the *i.i.d.* datasets generated for the Erdős-Rényi simulation. In terms of the quantity and locations of the estimated edges, BRS and BRS* are visually the most similar to the true graph. In contrast, the RS2 graph contains fewer edges whose locations match the true graph while the RS1 graph only contains a very small number

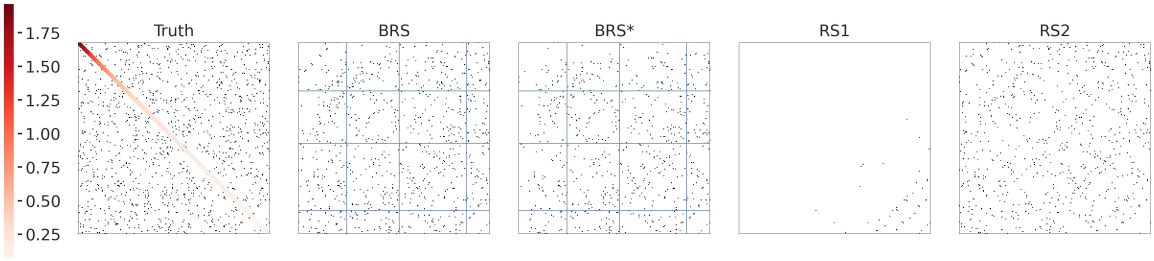

Figure 2.9: Adjacency matrices (from left to right) of the true graph, BRS graph, BRS* graph, RS1 graph, and RS2 graph in the Erdős-Rényi simulation, with $\alpha = 0.05$ and $n/p = 5$. The graphs are estimated based on the same Gaussian dataset. The colorbar indicates the magnitudes of the actual partial variances. The grids in BRS and BRS* show the variable grouping with the number of blocks being four. Regularization parameters are tuned by BRS, BRS* and RS. Un-normalized data is used for BRS, BRS*, and RS1, and normalized data is used for RS2. The inverse covariance matrix is estimated using QUIC.

of edges. These observations are consistent across the $i.i.d.$ datasets as well as in other cases where α and n/p take different values.

Next, we provide quantitative comparisons of the four methods using MCC and F_1. Figure 2.10 and 2.11 show the boxplots of MCC and F_1 for the Erdős-Rényi simulation. Four α values are considered and the corresponding results shown in the sub-figures: (a) $\alpha = 0.05$, (b) $\alpha = 0.35$, (c) $\alpha = 0.65$, and (d) $\alpha = 0.95$. Four n/p ratios are used for each α level, as indicated by the colors in the sub-figures. In each sub-figure, the top two rows show the boxplots for BRS and BRS*, the bottom two rows are for RS1 and RS2. Boxes towards the right indicate higher F_1 and MCC scores.

We see that when $\alpha = 0.05$ and 0.35, BRS achieves higher F_1 and MCC scores than the other three methods, for all values of n/p, as shown in Figure 2.10 (a), 2.10 (b), 2.11 (a), and 2.11 (b). The more accurate edge recovery performance of BRS and BRS* as compared to to the two variants of RS is due to that they allow adaptive regularization that incorporates different levels of variance of the data. In contrast, RS1 cannot handle the heterogeneous variance of the original data, and thus imposes a large scalar penalty

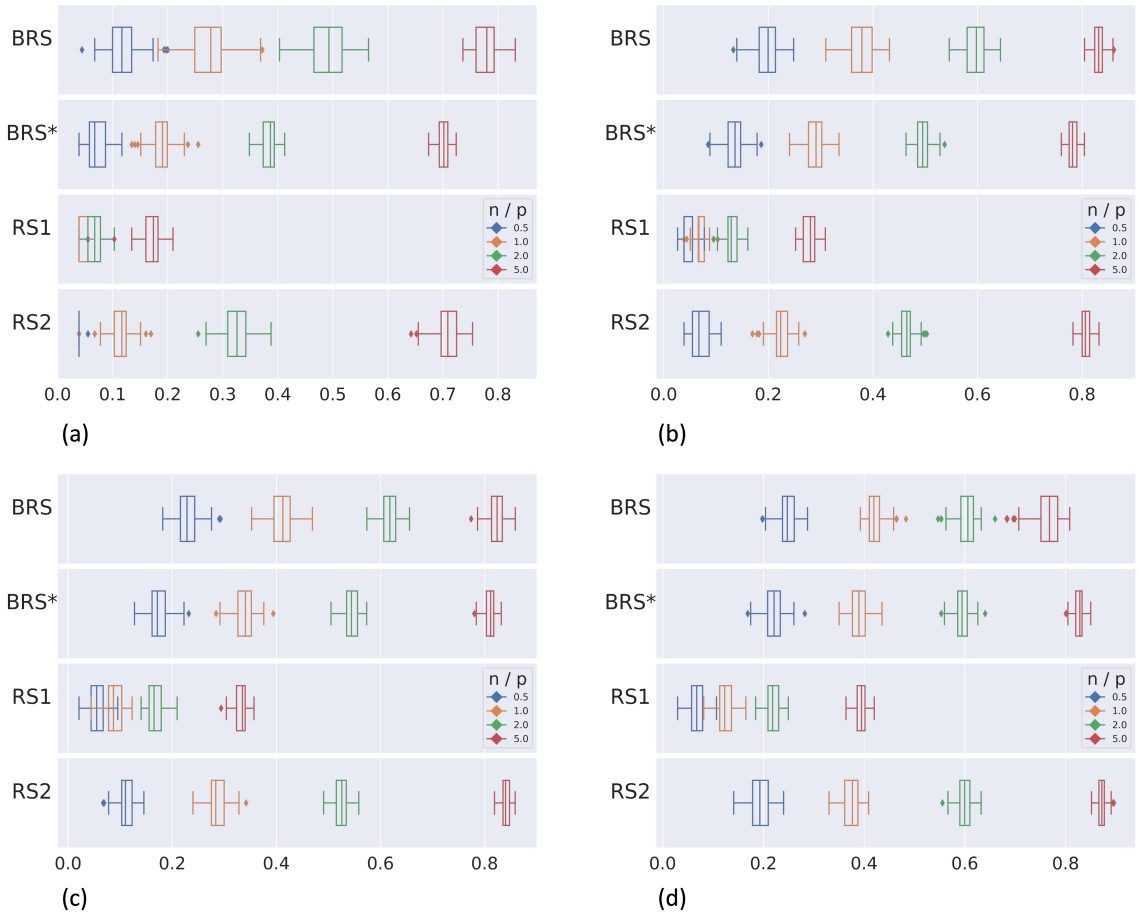

Figure 2.10: MCC boxplots of graph structure estimation in the Erdős-Rényi simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the MCC of block-wise robust selection results with un-normalized data, the second row shows the MCC of alternative block-wise selection results with un-normalized data, the third row shows the MCC of robust selection results with un-normalized data, and the bottom row shows the MCC of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

parameter, which leads to nearly edge-less graphs. For example, when $n/p = 0.5$ and $\alpha = 0.05$, no edges can be recovered by RS1 since the selected penalty parameter is too large. Such cases leave the MCC undefined and thus the corresponding box is missing in Figure 2.10 (a).

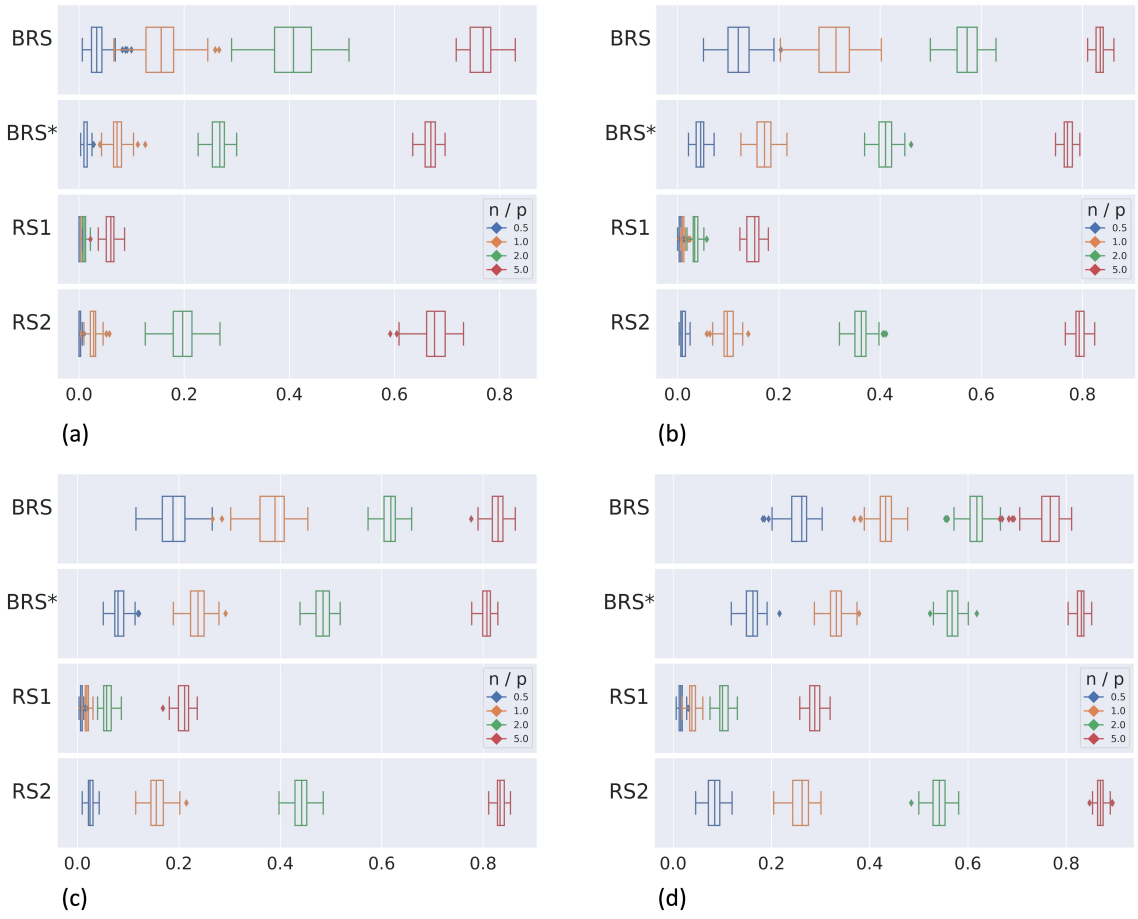

Figure 2.11: F_1 boxplots of graph structure estimation in the Erdős-Rényi simulation, when α is controlled at (a) 0.05, (b) 0.35, (c) 0.65, and (d) 0.95. In all sub-figures, the top row shows the F_1 of block-wise robust selection results with un-normalized data, the second row shows the F_1 of alternative block-wise selection results with un-normalized data, the third row shows the F_1 of robust selection results with un-normalized data, and the bottom row shows the F_1 of robust selection results with normalized data. The different colors indicate different n/p ratios. The number of variables is $p = 200$. Each box is computed over $m = 100$ Gaussian datasets. The inverse covariance is estimated using QUIC.

In addition, when data is deficient, i.e., $n/p \leq 1$, BRS outperforms BRS*, RS1, and RS2, in terms of both metrics, across all the α values, as can be seen in all sub-figures in Figure 2.10 and 2.11. While the tuning of penalty parameters can be challenging for Gaussian graphical models when data is deficient, BRS can still robustly recover the

true edges and achieve high F_1 and MCC scores in these cases. For instance, when $\alpha = 0.05$ and $n/p = 0.5$, RS1 and RS2 finds overly large penalty parameters for most of the 100 datasets, resulting in most of the graphs having no or very few edges. This causes the corresponding boxes to be missing or squeezed into a line in the figures. On the other hand, BRS still provides reasonable, non-empty graphs for all the datasets and achieve considerably higher F_1 and MCC scores as compared to these two methods. Furthermore, the accuracy of edge recovery in BRS benefits from the handling of the data dispersion via both variable grouping and the weight matrix when the sample is scarce. This explains the larger scores of MCC and F_1 attained by BRS when compared to BRS*.

The effect of data normalization can be seen by comparing RS1 and RS2. After normalization, the magnitudes of the variables become more comparable as the variances are homogenized. As such, the learned scalar penalty parameter can be applied to all the entries of the inverse covariance matrix. This is confirmed by observing that RS2 always achieves higher F_1 and MCC scores than RS1 in Figure 2.10 and 2.11, across all cases of n/p and α. However, as discussed earlier, the normalization procedure changes the data distribution. Moreover, when data is deficient, i.e., $n/p \leq 1$, the sample mean and variance can be unreliable for performing standardization in RS2, leading to less accurate graph estimation, as can be seen by comparing the blue and orange boxes in Figure 2.10 and 2.11.

When data is very sufficient and the significance level is low, i.e., $n/p = 5$ and $\alpha = 0.65$ or 0.95, the performance of RS2 improves and it achieves slightly higher F_1 and MCC scores than BRS. This is because, in this case, the sample mean and variance for each variable can more accurately reflect the true mean and variance, which allows RS2 to generate more accurate results. Note that in this case, while having slightly lower scores than RS2, our BRS still achieves reasonable F_1 and MCC scores and significantly

outperforms RS1.

2.3 Summary

We proposed block-wise robust selection for tuning the Gaussian graphical model in this chapter. The method finds the regularization parameters adaptively for different blocks of entries in the inverse covariance matrix. Our simulation results demonstrates that our proposed approach performs reasonably well in terms of edge recovery in different scenarios. This tuning method is also computationally efficient. We can utilize it for a wide range of applications that use Gaussian graphical models to characterize the relationship among variables. The next chapter studies environmental applications that utilize the tuned Gaussian graphical models for statistical inference.

Chapter 3

Paleo-Climate Reconstruction Using Block-wise Robust Selection

In this chapter, we take a comprehensive look into utilizing our proposed block-wise robust selection (BRS) to tune Gaussian graphical models for climate field reconstruction. The reconstruction problems can be viewed as imputing the missing values in the pre-instrumental period, based on the information learned from the post-instrumental years or another related reconstruction. We consider performing inference on the missing entries in two applications where the data comes from the HadCRUT4 model (Morice et al. (2012)) and the PAGES2k database (Consortium et al. (2017)). We use GraphEM (Guillot et al. (2015)) as the imputation framework, which embeds a Gaussian graphical model within a modified expectation-maximization (EM) algorithm. The Gaussian graphical model characterizes the spatial correlation structure over the global temperature field, which needs to be specified before the imputation. Towards this end, we apply our proposed BRS method discussed in the previous chapter to tune the graphical models. We further compare the BRS-based reconstruction results with those produced by other tuning methods.

The organization of this chapter is summarized as follows. In Section 3.1, we provide preliminaries of the datasets to be used. We use the ensemble and median data of

HadCRUT4.6 for the first application, and the aggregated yearly data of HadCRUT4.3 coupled with proxies in the PAGES2k database for the second application. In Section 3.2, we describe the imputation algorithm based on the Gaussian graphical models and show how we incorporate BRS in it. In Section 3.3, we present our first application where the missing values in HadCRUT4.6 ensemble data are imputed using two Gaussian graphical models, one is tuned by our proposed BRS algorithm and the other is manually tuned by domain knowledge. We compare the two tuning methods and the respective resulting reconstructed values. The second application of imputing the HadCRUT4.3 datasets leveraging the proxies is provided in Section 3.4, where we explore two variants of the BRS algorithm that use k-means and Gaussian mixture models for grouping the variables. The resulting reconstructions are compared with those obtained using the vanilla BRS with binary segmentation as the grouping method and using manually-selected regularization parameters, respectively.

3.1 Datasets

In this section, we introduce the two HadCRUT4 datasets (Morice et al. (2012)) to be imputed and one proxy dataset (Consortium et al. (2017); Neukom et al. (2019)) that is used to aid the reconstruction. The HadCRUT4 datasets provide monthly near-surface temperature anomalies over the globe from 1850 to the present with the reference period from 1961 to 1990. There are approximately 60% values missing in the dataset, which scatter in different spatial locations and time points. Our goal is to reconstruct these missing entries. The PAGES2k dataset (Consortium et al. (2017); Neukom et al. (2019)) provides the proxy records over some locations for the second application of yearly climate field reconstruction. The preserved proxy records are direct reflections of the climatic conditions in the past. Therefore, using them in the reconstruction enables the scientists

to study the climate changes over a longer historical period.

3.1.1 HadCRUT4 Datasets

HadCRUT4 data (Morice et al. (2012)) is the joint work of the Met Office Hadley Centre and the Climatic Research Unit at the University of East Anglia, which contains monthly time series temperature anomalies at a spatial resolution of $5^\circ \times 5^\circ$ over the globe. The temperature data spans from 1850 to the present, with the reference period from 1961 to 1990. Global coverage of temperatures is achieved by merging the land temperatures from CRUTEM4 (Jones et al. (2012)) and ocean temperatures from HadSST3 (Kennedy et al. (2011)). HadCRUT4 is constructed as an ensemble dataset which reflects the uncertainty in the temperature anomalies. In addition, a median dataset is constructed by computing the median temperature anomalies over the ensemble data at each spatial location. In our work, we use both the median and ensemble data while conducting the HadCRUT4.6 reconstructions, and use the aggregated yearly data for HadCRUT4.3 reconstructions with the aide of proxy data. Different versions in HadCRUT4 indicate different versions of the land-surface temperatures. HadCRUT4.3 uses the land-surface temperatures obtained from CRUTEM4.3 and HadCRUT4.6 uses CRUTEM4.6.

3.1.2 PAGES2k Database

It is known that the widespread availability of instrumental records for global surface temperatures began in 1850 (Brohan et al. (2006)). Due to the lack of climatic records, scientists use imprints created by past climate to interpret and study the paleo-climate. These imprints are known as proxies. Examples of climate proxies include tree rings, ice cores, corals, pollen grains, and ocean sediments. The proxies serve as an essential tool for temperature field reconstruction. They are the direct indicators of the paleo-climate,

which can provide insights into the climate history when instrumental observations are not available. Proxies have been used extensively for providing additional information about the trends of climate changes in the literature of climate field reconstruction (Guillot et al. (2015); Mann et al. (2007)), which facilitate the estimation and calibration of climate models.

PAGES2k database, which is provided by the PAGES2k consortium (Consortium et al. (2017); Ahmed et al. (2013); Neukom et al. (2019)), is a community-based record for temperature-sensitive proxies. In our second application that incorporates proxies in the reconstruction, we use proxy data originally obtained from PAGES2k database version 2.0.0 and later screened by a procedure guided by regional false discovery rate, as described in Neukom et al. (2019). The procedure also filters out records with temporal resolution lower than annual. Consequently, the resulting proxies are at annual or higher resolutions and cover 210 locations, including the ocean basins and all continental areas over the globe. In Figure 3.1, we show the availability of the proxy records and their types used in our second application. It can be seen that the proxies spread more in the northern hemisphere than in the southern. The majority of the proxies are trees (as indicated in brown) and corals (orange). As pointed out in Consortium et al. (2017), nearly half of the time series of the proxy data is significantly correlated with surface temperatures in HadCRUT4 version 4.2. This indicates that the PAGES2k proxy data can be helpful for temperature reconstruction in HadCRUT4 data.

3.2 Imputation via GraphEM

In this section, we provide a primer on using GraphEM (Guillot et al. (2015)) for missing value imputation based on Gaussian graphical models. The method uses graphical lasso (glasso) to estimate the inverse covariance matrix and a modified EM algorithm

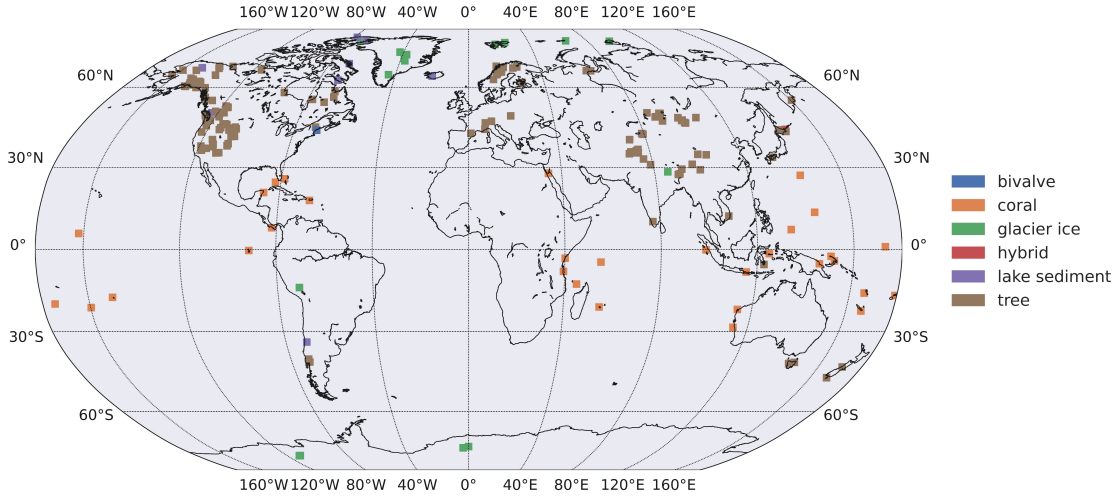

Figure 3.1: Available proxies in the PAGES2k database (Consortium et al. (2017)). Different colors indicate different types of proxy data. Trees are shown in brown, glacier ice is shown in green, coral is shown in orange, lake sediment is shown in purple, bivalve is shown in blue, and hybrid of tree and borehole is shown in red.

for imputation. We will use this algorithm for reconstructing the paleo-climate with the regularization parameters tuned by BRS. Specifically, the sparsity structure in the Gaussian graphical model indicates how the temperature at any location conditionally correlates with other locations. Our proposed BRS provides an efficient way to find the proper regularization in order to derive the proper sparsity pattern, which balances data likelihood and model complexity. Once the Gaussian graphical model is obtained, its adjacency graph is kept in the EM algorithm while estimating the missing entries. The procedure is formalized in the following discussion.

Suppose we have a p-dimensional random vector from the Gaussian distribution, denoted as $(X_1, \ldots, X_p)^\top \sim \mathcal{N}_p(\mu, \Sigma)$, where $\mu = (\mu_1, \ldots, \mu_p)^\top$ is the mean vector and Σ is the positive definite covariance matrix. Additionally, we assume that n $i.i.d$ samples from this distribution are collected and parts of the samples contain missing values. We denote the resulting data matrix as X. We provide an intuitive visualization of the missing and available data in Figure 3.2. Let us further denote a sample containing

missing values as $x = (x^m, x^a)$, consisting of the missing observations x^m for several variables and the available observations x^a for the rest. In our context of climate field reconstruction, the spatial grids over the globe are modeled by this Gaussian distribution. Each sample consists of observed global temperatures for a subset of the locations. We treat the temperatures at different times as $i.i.d$ in this study.

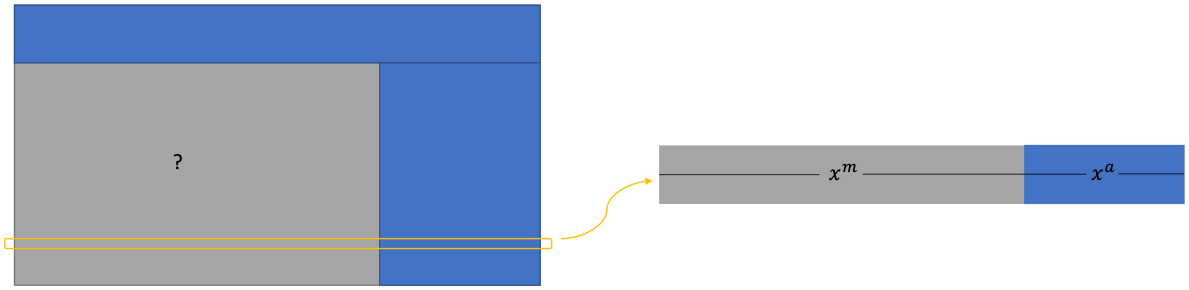

Figure 3.2: Data matrix with missing values. The missing values are indicated in gray and the available entries are in blue.

3.2.1 Sparse Graph Estimation

The first step of the algorithm is estimating a Gaussian graphical model and acquiring the adjacency graph. Based on the Gaussian assumption of the data, we use glasso (Friedman et al. (2008)) to obtain the inverse covariance matrix; see Chapter 1.1 for a detailed description of the estimator. We provide the formula for obtaining the estimate using the regularization parameter selected via BRS as follows:

$$K^{\Lambda} = \underset{K \succ 0}{\operatorname{argmax}} \ \log|K| - \operatorname{tr}(AK) - \|\Lambda \circ K\|_1, \tag{3.1}$$

where Λ is the block-wise penalty matrix tuned by BRS and A denotes the sample covariance matrix estimated from fully observed data from the reference period. The

graph G is then set to be the adjacency matrix of K^{Λ}, which is constructed by

$$G_{ij} = \begin{cases} 1, & \text{if } K^{\Lambda}_{ij} \neq 0 \\ 0, & \text{if } K^{\Lambda}_{ij} = 0 \end{cases}.$$

This estimated graph G will be kept in the modified EM algorithm for finding the relevant predictors while calculating estimates of the missing entries, as we shall discuss next.

3.2.2 EM Algorithm

In this part, we describe the modified EM algorithm used for imputation as proposed in Guillot et al. (2015). The discussion of a variant of this algorithm proposed by Vaccaro et al. (2021) will be deferred until Section 3.3. The modified EM algorithm works by 1) regressing the missing data to the available ones adjacent in the graph G, and 2) updating the mean and covariance estimates. These two steps are repeated until convergence. We provide a more accurate description of the procedure as follows.

Recall that we use $x = (x^m, x^a)$ to denote a sample that is partially observed, where the subscript m stands for "missing" and a for "available". Accordingly, we decompose the mean vector and the covariance matrix as $\mu = (\mu^m, \mu^a)^\top$ and

$$\Sigma^G = \begin{pmatrix} \Sigma^{G,mm} & \Sigma^{G,ma} \\ \Sigma^{G,am} & \Sigma^{G,aa} \end{pmatrix}.$$

For the covariance matrix, we use the notation Σ^G to indicate that it will be estimated based on the graph G, which distinguishes the algorithm from the classical EM algorithm in missing value imputation literature (Little and Rubin (2019)). The sub-matrix $\Sigma^{G,ma}$ stands for the covariance matrix computed between x^m and x^a. Note that the EM

procedure described below imputes the missing values of each sample separately at every iteration h, for $h \geq 0$. Then $\mu_{(h)}$ and $\Sigma^G_{(h)}$ are computed once all samples containing missing values are imputed.

Initialization: The initialization of the EM algorithm requires an initial guess of the missing values that completes the data matrix for computing $\mu_{(0)}$ and $\Sigma^G_{(0)}$. This can be done by replacing the missing values with the mean over the available records. As such, $\mu_{(0)}$ is set to be the average of all samples and $\Sigma^G_{(0)}$ is taken to be the sample covariance matrix.

E-step: Given the current estimates of $\mu_{(h)}$ and $\Sigma^G_{(h)}$, the missing values x^m are computed by

$$(x^m - \mu^m_{(h)}) = B_{(h)}(x^a - \mu^a_{(h)}), \tag{3.2}$$

where $B_{(h)}$ is the regression coefficient matrix that is calculated by

$$B_{(h)} = \left(\Sigma^{G,ma}_{(h)}\right)\left(\Sigma^{G,aa}_{(h)}\right)^{-1}.$$

The imputed matrix is denoted as $X_{(h+1)}$, after performing the E-step for each sample with missing values.

M-step: With the imputed data matrix $X_{(h+1)}$ and the graph G, we obtain the updated mean $\mu_{(h+1)}$ and covariance $\Sigma^G_{(h+1)}$. The mean for the j^{th} variable is updated by

$$\left(\mu_{(h+1)}\right)_j = \frac{1}{n}\sum_{k=1}^{n}\left(X_{(h+1)}\right)_{kj}, \tag{3.3}$$

where $\left(X_{(h+1)}\right)_{kj}$ means the entry at the k^{th} row and j^{th} column in $X_{(h+1)}$. The covari-

ance matrix $\Sigma^G_{(h+1)}$ is updated by

$$\Sigma^G_{(h+1)} = \underset{\Sigma=K^{-1};\ K_{ij}=0 \text{ if } G_{ij}=0}{\operatorname{argmax}} \log|K| - \operatorname{tr}\left(A_{(h+1)}K\right), \tag{3.4}$$

where $A_{(h+1)}$ is the sum of the sample covariance matrix of $X_{(h+1)}$ and the covariance of residuals, $R_{(h+1)}$, i.e.,

$$A_{(h+1)} = \frac{1}{n}\Big(X^\top_{(h+1)} - \mu_{(h+1)} \otimes \mathbf{1}_n^\top\Big)\Big(X_{(h+1)} - \mathbf{1}_n \otimes \mu^\top_{(h+1)}\Big) + R_{(h+1)}, \tag{3.5}$$

where $\mathbf{1}_n = (1, \ldots, 1)^\top$ is the n-vector whose entries are all one's and

$$R_{(h+1)} = \begin{pmatrix} 0 & 0 \\ 0 & \Sigma^{G,mm}_{(h)} - \Sigma^{G,ma}_{(h)}\left(\Sigma^{G,aa}_{(h)}\right)^{-1}\Sigma^{G,am}_{(h)} \end{pmatrix}.$$

3.3 Climate Reconstruction Using BRS

In this section, we reconstruct the HadCRUT4.6 ensemble anomalies by incorporating BRS into the GraphEM algorithm. We follow the workflow of GraphEM proposed in Vaccaro et al. (2021) for this application. First, the sparse Gaussian graphical model is estimated based on the short hybrid reconstruction (version 2.0) of HadCRUT4 by Cowtan and Way (2014), using data spanning from 1979 to 2019. Once the graph is obtained, the covariance matrix used for initializing the EM algorithm is obtained by Algorithm 17.1 in Friedman et al. (2001). Next, the initial guess of the missing values is provided by a long reconstructed dataset from Cowtan and Way (2014). The short and long reconstructed datasets provided by Cowtan and Way (2014) help rectify the bias in the computed global temperature trends from HadCRUT4.6, which is due to the handling of the unobserved regions at the poles and in Africa; we refer interested readers

to the original paper for more details on the discovered reconstruction bias caused by the uneven sampling of high-latitude regions. The workflow diagram of GraphEM is available in Section 2(d) and Figure 3 of Vaccaro et al. (2021).

In this GraphEM workflow, we apply our block-wise robust selection (BRS) described in Chapter 2 to tune the regularization parameters in the sparse graph estimation. In the original paper of Vaccaro et al. (2021), this tuning was conducted on a scalar penalty parameter to achieve a set of target densities.

3.3.1 Results

This part shows the reconstruction results by utilizing graphs obtained via BRS. First, we determine the number of variable groups by using the modified Bayes information criterion (MBIC) (discussed in Section 2.1.3). After evaluating a sequence of values from 1 to 30, we select 4 as it minimizes the MBIC. We then use four groups of variables when running the binary segmentation algorithm.

We show the resulting grouping of all temperature anomalies and their standard deviations in Figure 3.3. Each grid cell is colored by the standard deviation and numbered by the group's index. In terms of the numbering of the groups (zero, one, two, and three), groups zero and three correspond to the lowest and highest levels of dispersion, respectively. When two cells have the same number, their corresponding temperatures are assigned to the same group in BRS; two cells with different numbers have their respective temperatures in two different groups. This map shows that the grid cells belonging to the same group are most likely to be spatially nearby. For instance, the temperature anomalies in the southern Atlantic Ocean near the equator and southern America belong to the same group. In addition, most group transitions across neighboring grids are incremental, i.e., the transition happens between two consecutive groups. For instance,

group zero usually transitions to group one and rarely to group three. This is because the physical behavior of temperatures is locally smooth.

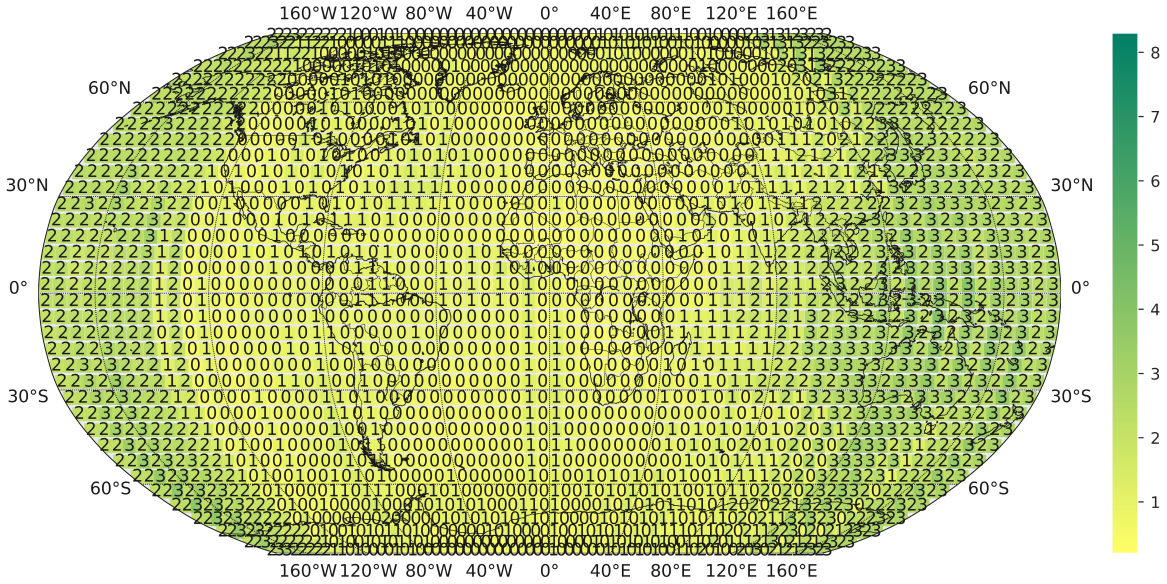

Figure 3.3: Grid map of standard deviations and group indices of temperature anomalies. The color bar indicates the magnitude of standard deviations. The number in each grid indicates the group index. Cells with the same number have their temperatures within the same group and different numbers indicate that the temperatures of the corresponding grids fall into different groups.

Next, we construct graphs using BRS with different significance levels, namely, $\alpha \in \{0.01, 0.05, 0.1, 0.4\}$. The final α value is determined based on the kernel density estimates of the anomaly probabilities, following a technique proposed by Vaccaro et al. (2021). More specifically, we draw the kernel density of the reconstructed anomalies for each α and that of the raw HadCRUT4.6 median. The significance level is set to be the one that results in the most similar kernel density (from imputed values) to that of the raw HadCRUT4.6 median. Moreover, since we follow the workflow of Vaccaro et al. (2021), we also compare the graph obtained with their method. Their best graph is the one with the most similar kernel density to the raw HadCRUT4.6 median, selected from several candidates obtained based on the target density criterion. In particular, this best graph

has a target density of 0.6%.

Figure 3.4 shows the kernel densities (log scale) for the raw HadCRUT4.6 median, the best graph obtained by Vaccaro et al. (2021), and graphs obtained with BRS using different α values. We see that BRS with $\alpha = 0.05$ (green) has the most similar kernel density to that of the raw HadCRUT4.6 median (brown). The kernel density of BRS with $\alpha = 0.01$ (orange) almost overlaps with that of BRS with $\alpha = 0.05$ (green) but deviates from the raw HadCRUT4.6 (brown) more than BRS with $\alpha = 0.05$ (green) near the two tails. In addition, the kernel density from the graph obtained in Vaccaro et al. (2021) (blue) almost overlaps with the BRS density with $\alpha = 0.4$ (purple) for the negative anomalies. Even though BRS density with $\alpha = 0.4$ (purple) has the smallest difference from the density of the HadCRUT4.6 median for the negative anomalies, the difference becomes large for the positive anomalies. Therefore, We pick the graph obtained by setting $\alpha = 0.05$ in BRS. The actual edge density is 0.47% in the BRS graph and 0.61% in the graph from Vaccaro et al. (2021).

We first compare the respective reconstructions for the median anomalies data when using the BRS graph and the graph from Vaccaro et al. (2021), based on the mean temperatures over the globe. To better illustrate the deviation of the two reconstructions to the raw HadCRUT4.6 median, we compute the difference between each reconstruction to the raw HadCRUT4.6 median. We show the respective difference time series in Figure 3.5, where orange stands for the BRS-based reconstruction and blue for the graph from Vaccaro et al. (2021). We see that the two reconstructions are similar to each other. The variabilities over time of the two time series closely match as the fluctuations are of similar magnitudes and occur at around the same time instances. In addition, the standard deviation of the difference time series is 0.15 for Vaccaro's reconstruction and 0.17 for our BRS-based reconstruction. This further indicates the similarity between these two reconstructions.

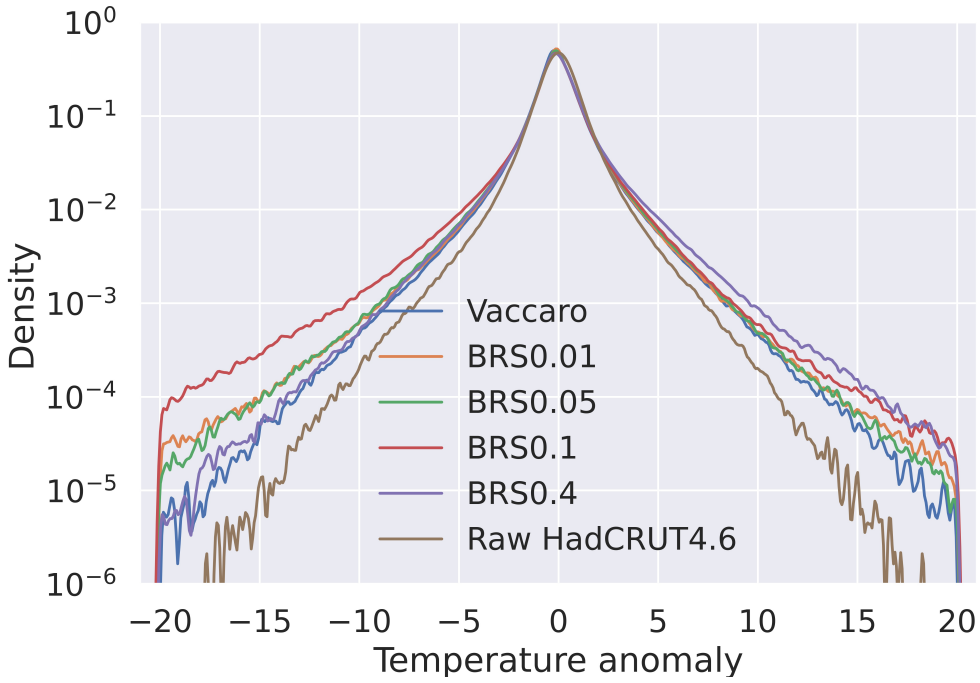

Figure 3.4: Kernel density estimates of temperature anomalies in different HadCRUT4.6 reconstructions. The extreme anomalies that exceed ±20 are excluded. The brown curve shows the kernel density of the raw HadCRUT4.6 median. The orange, green, red, and purple curves show the kernel densities of BRS-based reconstructions with $\alpha = 0.01$, 0.05, 0.1, and 0.4, respectively. The blue curve shows the kernel density of the HadCRUT4.6 reconstruction based on the graph from Vaccaro et al. (2021) with a target density of 0.61%. The curves are shown in log scale.

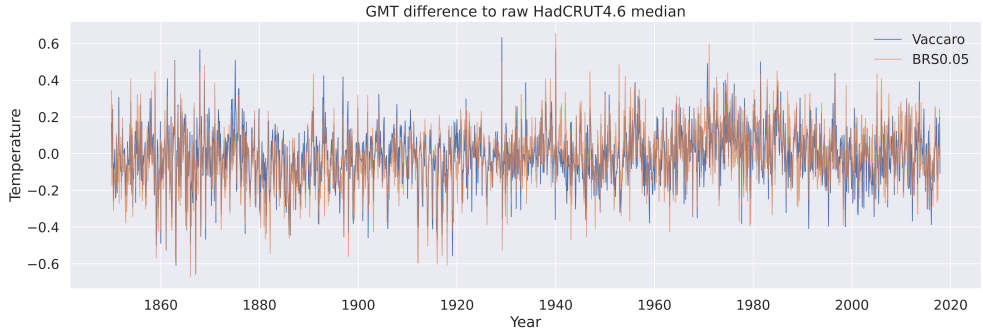

Figure 3.5: Difference of global mean temperatures to the raw HadCRUT4.6 median. The orange curve shows the difference based on reconstruction using BRS graph with $\alpha = 0.05$. The blue curve shows the difference based on reconstruction using Vaccaro's graph (Vaccaro et al. (2021)).

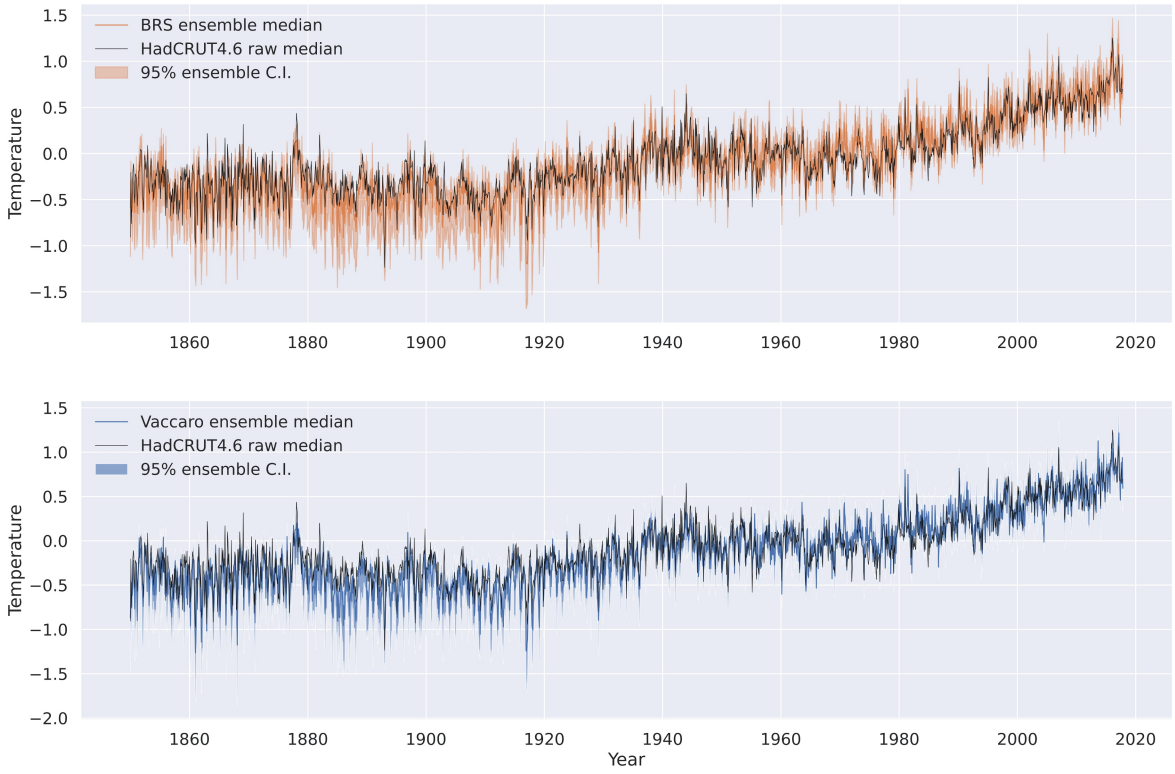

Figure 3.6: Median and 95% confidence interval (CI) of global mean temperatures based on ensemble HadCRUT4.6 reconstructions. The graph used for the top-panel reconstructions is BRS with $\alpha = 0.05$. The bottom-panel reconstructions use Vaccaro's graph (Vaccaro et al. (2021)). The solid black line in both figures indicates the raw HadCRUT4.6 median.

Next, we use the BRS graph with $\alpha = 0.05$ and Vaccaro's graph for reconstructing the 100-ensemble of HadCRUT4.6. We compute the median and 95% confidence interval (CI) for the global mean temperature for both sets of reconstructions. Figure 3.6 shows a comparison between the result of each set of the reconstructions and the raw HadCRUT4.6 median. Both figures show that the secular warming trend as of 1920 is consistent with the raw HadCRUT4.6 median. Seasonal trends that match with the raw HadCRUT4.6 median can also be seen in both time series.

In order to have a more direct comparison between our BRS-based reconstruction and the reconstruction using Vaccaro's graph, we draw scatter plots between their medians

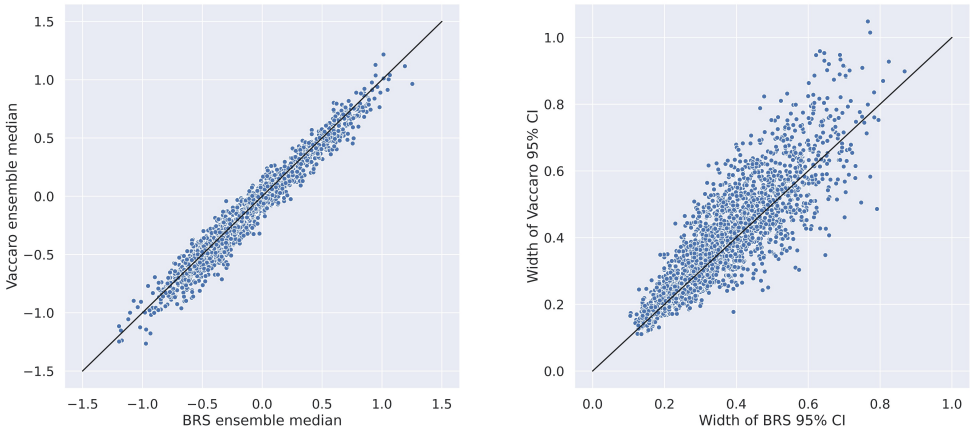

Figure 3.7: Left: per-time-point median of global mean temperatures from HadCRUT4.6 ensemble reconstructions by using the graph obtained via BRS with $\alpha = 0.05$ (x-axis) and Vaccaro's graph (y-axis). Right: per-time-point width of 95% confidence interval of global mean temperatures from HadCRUT4.6 ensemble reconstructions by using the graph obtained via BRS with $\alpha = 0.05$ (x-axis) and Vaccaro's graph (y-axis).

and 95% CIs, respectively. The left panel in Figure 3.7 shows the median comparison based on Vaccaro's graph (y-axis) and BRS graph (x-axis), and the right panel shows the comparison of the widths of the two CIs. We see that our reconstructions are similar to those based on Vaccaro's graph from both figures. In addition, 64% of our CI widths are smaller than Vaccaro's as the points are above the diagonal line.

In summary, we find these results encouraging, as the reconstruction obtained by using our BRS-tuned graph closely matches with a reconstruction based on a graph selected by climate scientists using a complex procedure. Our graph requires nearly 1/4 fewer edges to achieve a similar reconstruction. In addition, our BRS-based tuning of the graphical model is computationally efficient. For instance, in the case where $\alpha = 0.05$, it takes 80 seconds for the entire BRS procedure to complete on a Linux machine with four CPUs running at 2 GHz. As a comparison, it takes 27 minutes and 51 seconds for obtaining the target density graph, and the timing highly depends on how fine-grained

the search range is.

3.4 Climate Reconstruction Using Variants of BRS

This section explores different methods for determining the grouping of variables in BRS and evaluates their effects by examining the reconstructed temperature fields. The dataset used for this application is a hybrid dataset that combines the aggregated yearly anomalies in HadCRUT4.3 and the proxy records in the PAGES2k database.

First, we apply the original BRS described in Chapter 2, where we utilize the binary segmentation algorithm to group the temperatures and proxies based on their sample standard deviations. Then, since the temperature anomalies and proxy records have different means, we further consider incorporating this information to obtain the variable groups. We apply k-means clustering and Gaussian mixture model to the means and standard deviations of the temperature anomalies and proxies. Once we obtain the grouping by using each method, we compute the block-wise regularization parameters in the BRS algorithm. The Gaussian graphical model is then estimated and the imputation is performed based on the GraphEM algorithm. In addition, we provide two baseline reconstructions that use the target density criterion for tuning the Gaussian graphical model. Note that the regularization parameter tuning by any of our BRS variants is computationally efficient.

3.4.1 Grouping Using K-Means and Gaussian Mixture Model

In the original BRS discussed in the previous chapter, binary segmentation is used to group the variables based on their ordered sample standard deviations. This is based on the assumption that the central tendency of the variables is homogeneous. For example, zero-mean is assumed for the variables without loss of generality in the previous chapter.

However, the homogeneous central tendency is often not the case in paleo-climate reconstruction, as temperature anomalies and proxies represent different physical quantities and thus, have different means levels. Therefore, it is important to consider the means when finding the variable groups.

We construct a two-dimensional feature array that combines the sample means and sample standard deviations of the temperature anomalies and proxies to cluster the variables. The array is then fed into the clustering algorithm. Each record in this feature array is a pair of the sample mean and standard deviation of a temperature anomaly or a proxy. Figure 3.8 shows this feature array in a scatter plot. We can see that the proxy means range from -43 to 20, while the means of the temperature anomalies are around zero. In order to group the variables based on these two features, we apply two commonly used clustering methods, k-means clustering and Gaussian mixture model (Friedman et al. (2001); Murphy (2012)).

First, we describe how we use the k-means algorithm for determining the grouping for temperature anomalies and proxies. In Figure 3.8, we see that the means and standard deviations of proxies show a much larger variation as compared to those of the temperature anomalies. As such, directly applying k-means to the 2D feature array would cause the clustering result to be dominated by the characteristics of the proxies. As such, we use a two-step approach to group the temperature anomalies and proxies separately. More specifically, we first group the temperature anomalies using k-means. Next, we assign each proxy to the nearest cluster center of temperature anomalies. We use the Euclidean distance as the metric for measuring the distance.

The second method we use for grouping the temperature anomalies and proxies is the Gaussian mixture model. The temperature anomaly and proxy data points are assumed to be generated from a mixture of Gaussian distributions with unknown parameters. Compared to k-means, which returns hard-decision grouping of temperature anomalies,

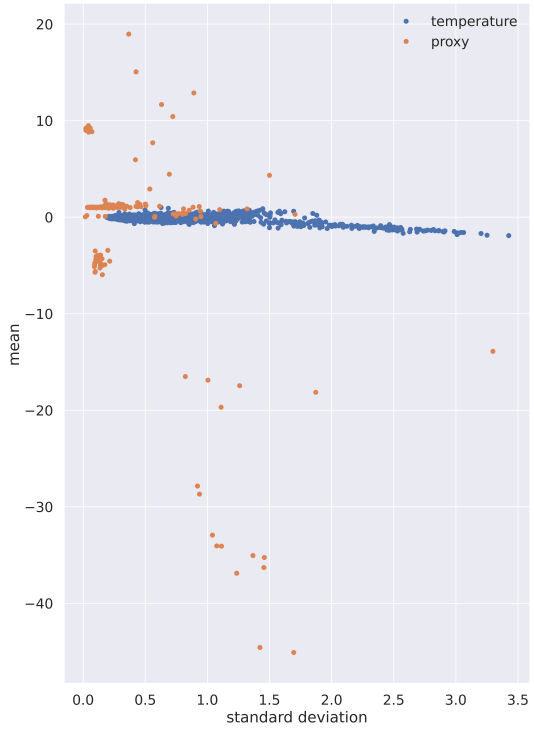

Figure 3.8: Scatter plot of the mean (y-axis) and standard deviation (x-axis) of temperature anomalies (blue) and proxies (orange) in HadCRUT4.6 application.

the Gaussian mixture model can return overlapping "soft" clusters containing a mixture of temperature anomalies and proxies.

We follow the standard ways of selecting the number of clusters for both k-means and the Gaussian mixture model. We use the Calinski-Harabasz score and the Silhouette coefficient for determining the number of clusters for both clustering methods. We also use the within-cluster sum of squares for k-means and BIC for Gaussian mixture model. For each metric except for the within-cluster sum of squares, the best number of clusters is the one that optimizes the corresponding metric. When using the within-cluster sum of squares as a metric for k-means, the best number of clusters corresponds to the elbow of the within-cluster sum of squares as this metric monotonically decreases as k increases.

3.4.2 Experiments

We present the reconstruction results of HadCRUT4.3 based on the hybrid data using different clustering methods for variable grouping in BRS. We utilize the target field provided by Neukom et al. (2019), which consists of 2592 temperature anomalies over the globe and 210 proxy records, spanning from 1881 AD to 1995 AD. There are no missing values in this target field; the missing values in the proxy records are filled using DINEOF (Neukom et al. (2019); Taylor et al. (2013)) and the missing temperature anomalies are filled using GraphEM. The temperature anomalies in this target field are the annual aggregated values over the April-to-March season window, based on the HadCRUT4.3 dataset. We use this target field to obtain the graph and assess the reconstruction performance. Specifically, 1911 AD to 1995 AD and 1881 AD to 1910 AD are used as the calibration and validation periods, respectively, as suggested by the authors.

In our experiments, the graphs are estimated based on the calibration-period data. The temperature anomalies of the validation period is treated as unknowns and reconstructed using GraphEM. For each variable grouping method, we consider three significance levels $\alpha = \{0.1, 0.4, 0.7\}$ and four numbers of groups $N = \{5, 8, 10, 12\}$. The reconstruction performance is evaluated based on three metrics, MSE, RE, and CE following Guillot et al. (2015). These metrics are computed over the validation period.

Let $X_{t,l}$ denote the temperature anomaly for the t^{th} year at the l^{th} location and $\hat{X}_{t,l}$ denote its reconstructed value. The evaluation metrics for each location are described as follows:

- Mean squared error (MSE) at a specific location l is defined as

$$(MSE)_l = \frac{1}{T} \sum_t \left(X_{t,l} - \hat{X}_{t,l} \right)^2,$$

where T is the number of validation years. The metric quantifies the average of the squared errors per location. When it equals zero, the reconstructed value equals the true value.

- Reduction of error (RE) compares the quality of the reconstruction to a mean reconstruction obtained via the calibration-period data. Formally, it is

$$(RE)_l = 1 - \frac{(MSE)_l}{(MSE)_l^{calib}},$$

where $(MSE)_l^{calib}$ is the MSE of a constant reconstruction which replaces the missing values with the mean from the calibration period. RE has a maximum of one and is not bounded from below. When $RE = 1$, the reconstruction is 100% better than the calibration-mean reconstruction, which happens when $(MSE)_l = 0$. When $RE = 0$, the assessed reconstruction has the same performance as the calibration-mean reconstruction in terms of MSE.

- Coefficient of error (CE) compares the quality of the reconstruction to a mean reconstruction that is obtained via the validation-period data. The metric is given by

$$(CE)_l = 1 - \frac{(MSE)_l}{(MSE)_l^{valid}},$$

where $(MSE)_l^{valid}$ is the MSE of the mean reconstruction that replaces the missing values by the average of the validation-period data. CE has a maximum of one and is not bounded from below. When $CE = 1$, the reconstruction is 100% better than the validation-mean reconstruction, which happens when $(MSE)_l = 0$. When $CE = 0$, the assessed reconstruction has the same performance as the validation-mean reconstruction in terms of MSE.

Table 3.1: Median of MSE, RE and CE over the globe for reconstructions with different N and α values in the BRS graphs. The variable grouping is obtained by binary segmentation (BS), k-means clustering (KM), and Gaussian mixture model (GMM), respectively. The best number for each metric is highlighted in bold.

Metric	α	Grouping	$N = 5$	$N = 8$	$N = 10$	$N = 12$
MSE	0.1	BS	0.29	0.29	0.29	0.34
		KM	0.31	0.31	0.31	0.33
		GMM	0.34	0.33	0.31	0.28
	0.4	BS	0.30	0.31	0.30	0.34
		KM	0.29	0.29	0.31	0.34
		GMM	0.35	0.31	0.29	0.31
	0.7	BS	0.30	0.32	0.32	**0.2**
		KM	0.29	0.30	0.34	0.36
		GMM	0.33	0.29	0.32	0.33
RE	0.1	BS	0.21	0.19	0.24	0.00
		KM	0.12	0.11	0.11	0.00
		GMM	0.00	0.08	0.16	0.14
	0.4	BS	0.16	0.16	0.19	0.00
		KM	0.21	0.19	0.17	0.00
		GMM	0.01	0.16	0.25	0.09
	0.7	BS	0.19	0.12	0.10	**0.49**
		KM	0.11	0.19	0.08	0.00
		GMM	0.08	0.22	0.13	0.03
CE	0.1	BS	-0.46	-0.48	-0.74	-0.78
		KM	-0.54	-0.58	-0.55	-0.70
		GMM	-0.79	-0.64	-0.55	-0.56
	0.4	BS	-0.53	-0.54	-0.46	-0.81
		KM	-0.47	-0.51	-0.54	-0.75
		GMM	-0.74	-0.52	-0.50	-0.64
	0.7	BS	-0.56	-0.64	-0.54	**0.24**
		KM	-0.58	-0.58	-0.83	-0.88
		GMM	-0.67	-0.51	-0.69	-0.75

Table 3.1 shows the median for each evaluation metric, when each grouping method is used for BRS under different α and N settings. The best performance numbers (lowest MSE, highest RE and CE) are highlighted in bold. The number of edges in each graph is summarized in Table 3.2, where the bold number corresponds to the best number in Table 3.1. Across all tested scenarios, the reconstruction using binary segmentation

Table 3.2: Number of estimated edges in each BRS graph with different N and α values. The variable grouping is obtained by binary segmentation (BS), k-means clustering (KM), and Gaussian mixture model (GMM), respectively. The highlighted quantity corresponds to the best performance in Table 3.1 across all α, N, and grouping methods.

α	Grouping	$N = 5$	$N = 8$	$N = 10$	$N = 12$
0.1	BS	25,652	28,110	30,771	462,201
	KM	21,128	24,394	24,940	136,473
	GMM	4,036	18,618	18,701	246,192
0.4	BS	28,110	28,901	34,292	568,368
	KM	25,553	27,590	28,224	262,056
	GMM	5,943	25,841	25,984	611,001
0.7	BS	32,551	40,369	44,216	**889,550**
	KM	30,188	31,694	32,381	428,982
	GMM	9,058	32,953	34,444	1,065,440

with $N = 12$ and $\alpha = 0.7$ achieves the lowest MSE and the highest RE and CE. The majority of the combinations of (α, N, Grouping) produce similar performance values. This indicates that it is flexible to use different variable grouping methods in BRS.

In Table 3.1, we observe that among all the cases using k-means, the median MSE gets higher, and median RE and CE get lower as N increases. The best performance is achieved when $\alpha = 0.4$ and $N = 5$. When using Gaussian mixture model as the grouping method, the best performance is achieved when $\alpha = 0.4$ and $N = 10$. When $N = 5$, the median RE and CE are low and MSE is high across all α levels. This is because the reconstruction quality at any location depends on the neighboring sites. When the number of neighbors (i.e., the estimated edges) are too few (see Table 3.2), the reconstruction cannot leverage correlation and provides results similar to the constant reconstruction based on the calibration-period mean; thus RE is close to zero.

Figure 3.9 shows the RE maps of the respective reconstructions when using the three different variable grouping methods in BRS. The top row shows the results of our main BRS approach with binary segmentation (BS), the middle shows the results of the variant

with k-means (KM), and the bottom shows the results of the variant using Gaussian mixture model (GMM). The α and N setting for each variant is selected based on the lowest MSE and highest RE and CE in Table 3.1.

Table 3.3: Median of MSE, RE and CE over the globe for reconstructions with target density graphs of 0.9% and 2% edge densities.

Metric	0.9%	2%
MSE	0.27	0.36
RE	0.28	0.02
CE	-0.3	-0.89

In Table 3.3, we provide the same set of performance evaluations for reconstructions obtained using graphs selected via domain expertise. This is done by first searching over a wide range of penalty parameter values and then determining the parameter that leads to a graph with the closest edge density to a given desired value. We follow Neukom et al. (2019) where the scientists use 2% as the target edge density for the graph estimation. In addition, we also experiment with a graph with 0.9% edge density. The resulting numbers of edges for the 0.9% and 2% graphs are 35,995 and 67,016, respectively. The 2% graph shows a higher median MSE and a lower CE when compared to the best performance of BRS and its variants. The 0.9% graph shows better performance with smaller MSE and higher RE and CE. However, this approach has the drawback of requiring an expensive search for the penalty parameter, as we elaborate next.

Table 3.4: Timing comparison for graphical model tuning using BRS and domain expertise. For BRS, we use binary segmentation for variable grouping.

Tuning Method	Wall Time
BRS ($\alpha = 0.7$, $N = 12$)	78 seconds
Target density 2%	11 hours and 15 minutes 3 seconds
Target density 0.9%	41 minutes 12 seconds

While the manually-tuned graph (target density 0.9%) achieves similar performance values as BRS and its variants under most α and N settings, our proposed BRS is more

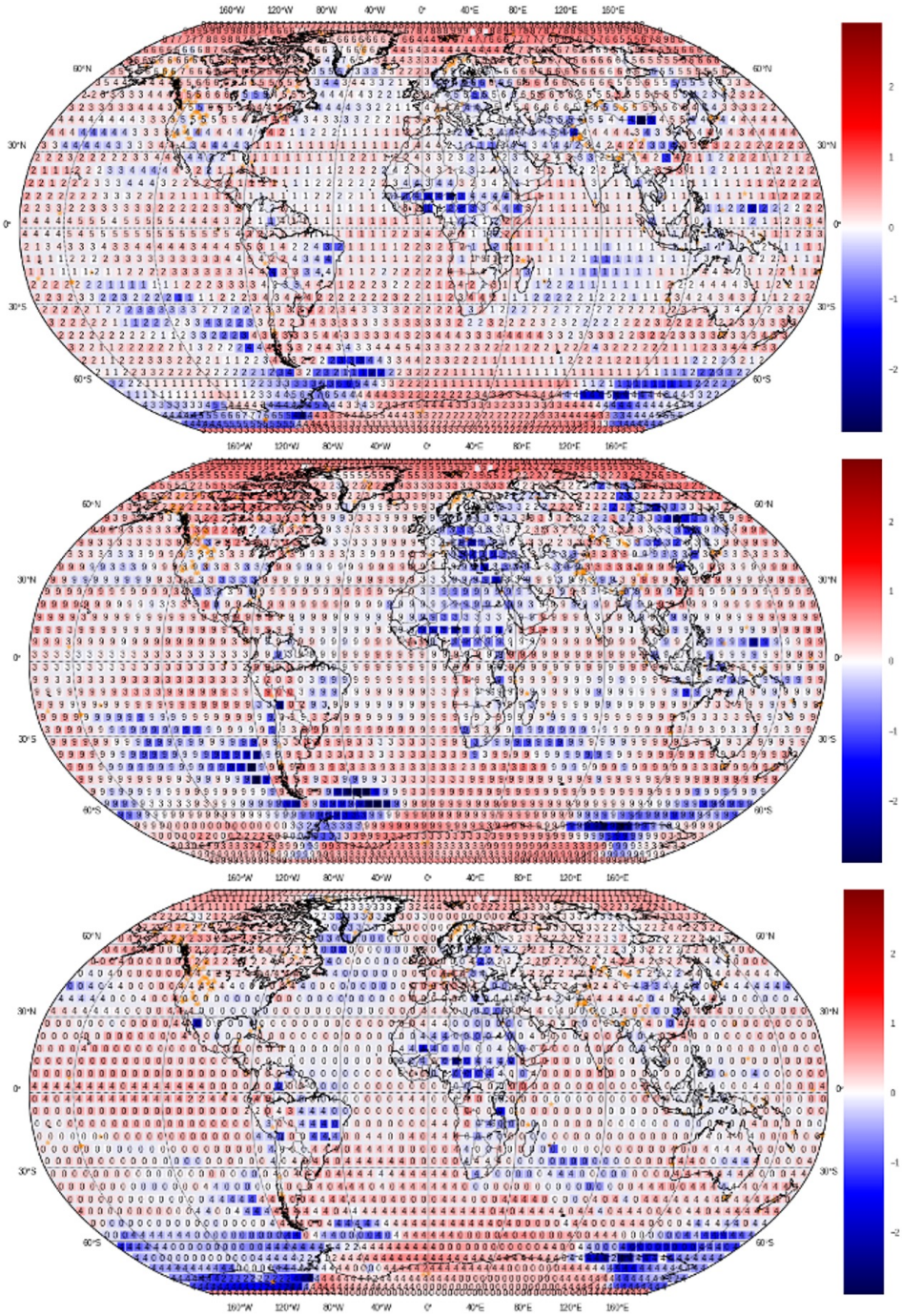

Figure 3.9: *RE* map of the reconstructions obtained by our main BRS approach with binary segmentation (top) and the variants with k-means (middle) and Gaussian mixture model (bottom). The α and N values for each case are as follows: for BRS with binary segmentation, $\alpha = 0.7$ and $N = 12$, for the k-means variant, $\alpha = 0.4$ and $N = 5$, and for the Gaussian mixture variant, $\alpha = 0.7$ and $N = 10$. The color indicates magnitude of the *RE* score for each grid. The number in each cell indicates the block assignment.

computationally efficient. Table 3.4 shows a wall-time comparison for BRS with binary segmentation and the target-density-based approach, which tunes graphs with domain expertise. It can be seen that computation times for tuning the target density graphs are significantly higher than that of BRS. The timing experiments are run on a Linux machine with four CPUs running at 2 GHz.

3.5 Summary

In this chapter, we provided two real-life climate field reconstruction applications using block-wise robust selection. Both applications utilized the tuned graphical models for down-stream tasks of imputing missing temperature values. We discussed the reconstruction results based on our tuning method and compared them with others obtained by tuning methods that require domain knowledge. We also explored using different clustering techniques for grouping the variables in our block-wise robust selection approach.

Chapter 4

Emulation of RHESSys

In a wide range of scientific areas, simulators have been extensively utilized to describe complex physical systems. A simulator is a mathematical model implemented in computer codes that captures the underlying physics (e.g., via partial differential equations) of the process to be simulated. In this chapter, we consider a watershed-level mechanistic model, Regional Hydro-Ecologic Simulation System (RHESSys); see Tague and Band (2004) for more details. RHESSys is a simulator that takes specific environmental parameters as input and generates the resulting environmental phenomena (Kennedy et al. (2017); Tague and Grant (2004); Tague and Peng (2013)). This allows researchers to generate different environment scenarios and study the sensitivity of the environment to certain factors (Zierl et al. (2007)). In the field of hydrology, researchers run RHESSys with different soil inputs in order to understand how soil affects streamflow (Tague et al. (2004)). In this chapter, we study how subsurface drainage and storage parameters of soil, as described in Table 4.1, impact streamflow. For efficiency, we will refer to them as "soil" in the remaining discussion of the chapter.

While RHESSys enables researchers to explore different scenarios and understand the associations between environmental factors, including under diverse scenarios where real-world measurements are not available, it is computationally expensive to run. For instance, a single 37-year simulation on a high-performance computing cluster with 1200

Table 4.1: Input soil conditions in RHESSys simulator

Input	Explanation
m	controlling decay of hydraulic conductivity with depth
K	hydraulic conductivity at the surface
po, pa	controlling soil storage capacity
$gw1, gw2$	controlling bypass flow to and from deeper aquifers

processors (20 nodes) and 3.06 TB of RAM took 137 hours for a 565 km^2 watershed, composed of 189,872 patches. In order to address such computational challenges, efficient surrogate models have been proposed to approximate the results of these physics-based simulators, which are referred to as statistics-based emulators. An emulator is designed to replicate the input-output behavior of a simulator of interest while being much faster to run. In addition, the emulator provides an uncertainty quantification of the output, thus capturing the sensitivity of the results, which would have only been obtained by running the simulator multiple times. Researchers can perform large-scale numerical experiments to understand complex environmental relationships with significantly reduced computation time and resources by employing such an emulator.

We propose to use a Gaussian Process (GP)-based model with temporal seasonality to emulate RHESSys. We first define a Gaussian Process.

Notation: For a random process $\mathcal{Y}$ over region $\mathcal{D} \subset \mathbb{R}^p$, let $Y(\boldsymbol{w})$ represent this process considered at the single location $\boldsymbol{w}$ within the support of this random process; i.e., $Y(\boldsymbol{w})$ is the corresponding random variable at p-dimensional point $\boldsymbol{w}$ within region $\mathcal{D}$.

Gaussian Process Definition: A random process $\mathcal{Y}$ over region $\mathcal{D} \subset \mathbb{R}^p$, is a Gaussian process (GP)

if and only if the following holds:

for each finite $M \in \{1, 2, \ldots\}$, and every set of M points $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \ : \ \boldsymbol{w}_\iota \in \mathcal{D} \ \ \forall \ \iota = 1, \ldots, M\}$, if $\boldsymbol{Y}^* \doteq [\ Y(\boldsymbol{w}_1), \ldots, Y(\boldsymbol{w}_M)\ ]^T$, then $\boldsymbol{Y}^* \sim \mathcal{N}_M(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$. †

† i.e., random vector $\boldsymbol{Y}^*$ follows M-dimensional multivariate normal distribution, with mean $\boldsymbol{\mu}^*$ a column vector of length M having i^{th} element $E[Y(\boldsymbol{w}_i)]$, and $M \times M$- dimensional covariance matrix $\boldsymbol{\Sigma}^*$ having $(i,j)^{th}$ element $Cov(Y(\boldsymbol{w}_i), Y(\boldsymbol{w}_j))$.

GP Notation: If $\mathcal{Y}$ is a Gaussian Process over region $\mathcal{D} \subset \mathbb{R}^p$ then we write $\mathcal{Y} \sim GP(\mu(.),\ C(.,.))$, where $\mu(.)$ is the mean function, with $\mu(\boldsymbol{w}) = E(Y(\boldsymbol{w}))$, and $C(.,.)$ is the covariance function with $C(\boldsymbol{w}, \boldsymbol{w'}) = Cov(Y(\boldsymbol{w}), Y(\boldsymbol{w'}))$, for each $\boldsymbol{w}$ and $\boldsymbol{w'} \in \mathcal{D}$.

In this work, our goal is to emulate the hydrological behavior of RHESSys under different values of soil inputs (representing different soil property scenarios) under each of two contrasting climates. For the latter, we study two watersheds: Rattlesnake Canyon and Sagehen Creek described later. To build this statistics-based emulator for each watershed, we need a training set that contains several sample simulator runs. More specifically, each element in our training set consists of the values of soil input variables for one simulator run (one value for each soil input variable) and the resulting time series output for this simulator run. We denote the input as $\mathbf{x}_i = (x_{i1}, ..., x_{id})$, for the i^{th} simulator run with $i \in \{1, ..., n\}$, where n is the total number of simulator runs in the training set and d is the dimension of the soil input. Soil inputs do not change over time within a simulator run. In particular, x_{iq} denotes a scalar for the q^{th} input in the i^{th} simulator run. We denote the design matrix of the training input settings as $\mathbf{X} \in \mathbb{R}^{n \times d}$, namely the matrix with i^{th} row $\mathbf{x}_i$, and q^{th} column X_q. We use $\mathbf{t} = (t_1, \ldots, t_j, \ldots, t_l)$ to denote a regular grid of time points where t_j is the j^{th} time point for $j \in \{1, \ldots, l\}$. We use $\mathbf{y}_i = (y_i^1, \ldots, y_i^l)^\top \in \mathbb{R}^l$ to denote the resulting time series output for $\mathbf{x}_i$ and $\mathbf{y}^j = (y_1^j, \ldots, y_n^j)$ denotes the outputs at time t_j across the n simulator runs, for each $j \in \{1, \ldots, l\}$. Let $\mathbf{Y} = (\mathbf{y}^1, \ldots, \mathbf{y}^l)^\top \in \mathbb{R}^{(ln)}$ denote the stacked time series outputs over $\mathbf{y}^j$. RHESSys was also run to construct a test set to assess the performance of our emulator for each watershed. We use * to denote the inputs and outputs in the test set, i.e., $\mathbf{x}_i^*$ and $\mathbf{y}_i^*$ are the input and output of the i^{th} test simulator run. In addition, we use $\hat{\mathbf{y}}_i^*$ to denote the predicted simulator output by the emulator for the i^{th} run (i.e., the prediction from the emulator when we provide the i^{th} test simulator run's input, $\mathbf{x}_i^*$, to the emulator). These notations are summarized in Table 4.2.

GP models inherit rich theory from well-established studies of Gaussian Processes and of the multivariate normal distribution (see for example Rasmussen (2003)). This provides rigorous theoretical support for building GP-based emulators and using them

for statistical inference. In addition, as the number of training samples increases, one can enhance the flexibility of GP-based models. Therefore GP-based models form a rich class within which we develop emulators for RHESSys. Furthermore, to capture the seasonality of the environmental behavior in the RHESSys output, we embed a time-varying mean function in the GP-based emulator that can characterize such seasonal trends. By applying our GP with seasonality emulator to the simulator runs for each of the two watersheds, we show that our proposed emulator can realistically predict/approximate many desirable features of the simulator output. Moreover, by leveraging the proposed emulator, we conduct a global sensitivity analysis, which reveals relationships between the input and output, e.g., how sensitive the hydrological output is to a variety of soil conditions (i.e., to different soil scenarios). Based on this analysis, we further identify the most important input variables (amongst RHESSys soil inputs), to which the RHESSys output variables exhibit sensitivity.

The remainder of this chapter is organized as follows. In Section 4.1, we describe the Gaussian Process-based methodology used to efficiently emulate time series computer-model (simulator) output given scalar-type inputs. The global sensitivity analysis using quasi-random sampling and variance-based indices are presented in Section 4.2. The analysis provides insights into how the variation in the model output at each time step can be factored into components of variation associated with different inputs. Section 4.3 describes the RHESSys simulator and challenges in evaluating it for various experiments, followed by a detailed description of the two datasets used in our numerical analysis. We then provide two sets of emulation results in Section 4.4 using our proposed method. Every output variable is studied at two experimental watersheds dominated by winter precipitation. One is Rattlesnake Canyon, where the climate is usually dry for most of each year (winter rainfall area). The other is Sagehen Creek in the Sierra Nevada, where the climate is wet, cold in winters with snow accumulation. The comparison of the two

Table 4.2: List of variables pertaining to the training and testing simulator runs. For simplicity, we use the same notation for both watersheds.

Symbol	Meaning
x_{iq}	The q^{th} input in the i^{th} training simulator run
$\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$	Vector of soil inputs for the i^{th} simulator run
$\mathbf{X}_{n\times d}$	Training design matrix
n	Number of simulator runs for training the emulator
d	Dimension of soil input space for the simulator
X_q	The q^{th} input
X_{-q}	All inputs except for the q^{th} input
$\mathbf{t} = (t_1, \ldots, t_l)$	A regular grid of time points of length l
l	Number of time points
t_j	The j^{th} time point, with $j \in \{1, ..., l\}$
$\mathbf{y}_i = (y_i^1, \ldots, y_i^l)^\top$	Observed time series simulator output for $\mathbf{x}_i$ over $\mathbf{t}$, $\mathbf{y}_i \in \mathbb{R}^l$
$\mathbf{y}^j = (y_1^j, \ldots, y_n^j)$	Time-series output at t_j across n simulator runs, $j \in \{1, \ldots, l\}$
$\mathbf{Y} = (\mathbf{y}^1, \ldots, \mathbf{y}^l)^\top$	Stacked time series outputs over $\mathbf{y}^j$, $\mathbf{Y} \in \mathbb{R}^{(ln)}$
x_{iq}^*	The q^{th} input of the i^{th} testing simulator run
$\mathbf{x}_i^*$	Vector of soil inputs for the i^{th} testing simulator run
n^*	Number of simulator runs for testing the emulator
$\mathbf{y}_i^*$	Observed time series output from the simulator for $\mathbf{x}_i^*$
$\hat{\mathbf{y}}_i^*$	Predicted time series output from the emulator for $\mathbf{x}_i^*$

watershed sites is also discussed in this section.

4.1 Gaussian Process Emulator with Seasonality

This section presents our proposed emulator that utilizes Gaussian Process (GP) with embedded seasonality modeling. We first discuss how we specify the mean function capturing seasonality and the covariance function for the GP. We then describe the estimation of the parameters in the emulator model and the efficient prediction/approximation of the simulator output using the derived emulator. Table 4.3 summarizes the notation used here.

Table 4.3: List of variables pertaining to model specification. Here "spatial" refers to d-dimensional soil input space; each soil input vector represents a d-dimensional coordinate within this d-dimensional soil inputs space.

Symbol	Meaning
$\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^l)^\top$	Mean vector for stacked outputs $\mathbf{Y}$, $\boldsymbol{\mu} \in \mathbb{R}^{(ln)}$
$\boldsymbol{\mu}_i = (\mu_i^1, \ldots, \mu_i^l)^\top$	Mean vector for $\mathbf{y}_i$, where $\mu_i^j = \mathbb{E}(y_i^j)$
$\boldsymbol{\mu}_i^* = \left((\mu_i^1)^*, \ldots, (\mu_i^l)^*\right)^\top$	Mean vector for $\mathbf{y}_i^*$
$\mathbf{s} = (s^1, \ldots, s^j, \ldots, s^l)^\top$	Seasonal process, s^j is the seasonal component at $t_j \in \mathbf{t}$
k	Index of the periodic process, $k \in \{1, \ldots, r\}$
r	Total number of periodic processes
ω_k	Frequency of the k^{th} trigonometric functions
U_k, V_k	Coefficient of the k^{th} cosine and sine function, respectively
β_q	Regression coefficient of the q^{th} input
$\boldsymbol{\Sigma}$	"Spatio"-temporal covariance matrix of $\mathbf{Y}$
$\boldsymbol{\Sigma}_t = (\sigma^{ij})_{i,j}$	Temporal covariance matrix of the process over $\mathbf{t}$
$\boldsymbol{\Sigma}_X = (\zeta_{ij})_{i,j}$	"Spatial" covariance matrix of the process over inputs $\mathbf{X}$
$\boldsymbol{\Sigma}_i^*$	"Spatio"-temporal covariance matrix of $\mathbf{y}_i^*$
$\Sigma_{\mathbf{x}_i^* X}$	Covariance matrix between the process at inputs $\mathbf{x}_i^*$ and $\mathbf{X}$
ρ	First-order temporal auto-correlation (lag 1 unit)
v	Partial sill
ϕ_q	Range parameter for the q^{th} input
τ	Nugget
δ	Dirac function

4.1.1 Gaussian Process

We model the stacked emulator outputs of the training runs as being generated from a Gaussian process, analogously to the emulator in Olson et al. (2018), referred to as the *stilt emulator*. Our emulator will interpolate the simulator training set output at the design matrix of training set input values. For simplicity, we, therefore, use notation $\mathbf{Y}$ also for the emulator output at the training set inputs. The simulator is deterministic, but in order to predict/approximate the simulator output using a statistical emulator, we assume the corresponding emulator values $\mathbf{Y}$ are random. We assume the random emulator process $\mathcal{Y}$ across space and time is a Gaussian Process (mean and covariance functions defined later). This implies that to develop the emulator, we assume the finite

set of emulator output at the training time points and soil inputs follow a multivariate normal distribution:

$$\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{4.1}$$

where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix that characterizes the input-output relationship. For example, when using a distance-based covariance matrix, the outputs should be correlated when the soil input vectors are similar (i.e., relatively close together in soil input space). Each GP can be fully specified by its mean and covariance function.

Within this GP-based emulation framework, we propose a novel way to characterize the seasonality in the mean function, which was not considered in the previously cited *stilt* work where the mean function is modeled as the linear combination of the simulator inputs. Next, we provide detailed descriptions of the model specification.

In practice, we may transform the original simulator runs and develop our emulator using a GP on the transformed scale in order to make the GP assumptions more reasonable. For simplicity, $\mathbf{Y}$ represents transformed simulator output and emulator values on the transformed scale in such cases, unless otherwise specified.

4.1.2 Mean Function

Time series data often exhibit periodic or seasonal patterns that reflect the underlying physical process. For instance, the streamflow in a hydrological system shows seasonal patterns, wherever seasonal weather patterns highly influence the flow of water.

A motivating time series example is provided here. Figure 4.1 shows one randomly selected time series of streamflow obtained from the RHESSys simulator (top panel) and an additive decomposition of this simulator output with three components: trend (second row), seasonal component (third row), and a remaining portion (fourth row) that cannot

be explained by the first two terms. The decomposition was performed after applying a variance-stabilizing transformation to the raw time series. The transformed daily data are plotted day-to-day for ten consecutive years. The decomposition includes a seasonal pattern repeating every year (365 days) with two local maxima, the first a low peak with a maximum of approximately 0 and the second dominant peak reaching approximately 1.5. This pattern summarizes streamflow variation across different seasons in a year.

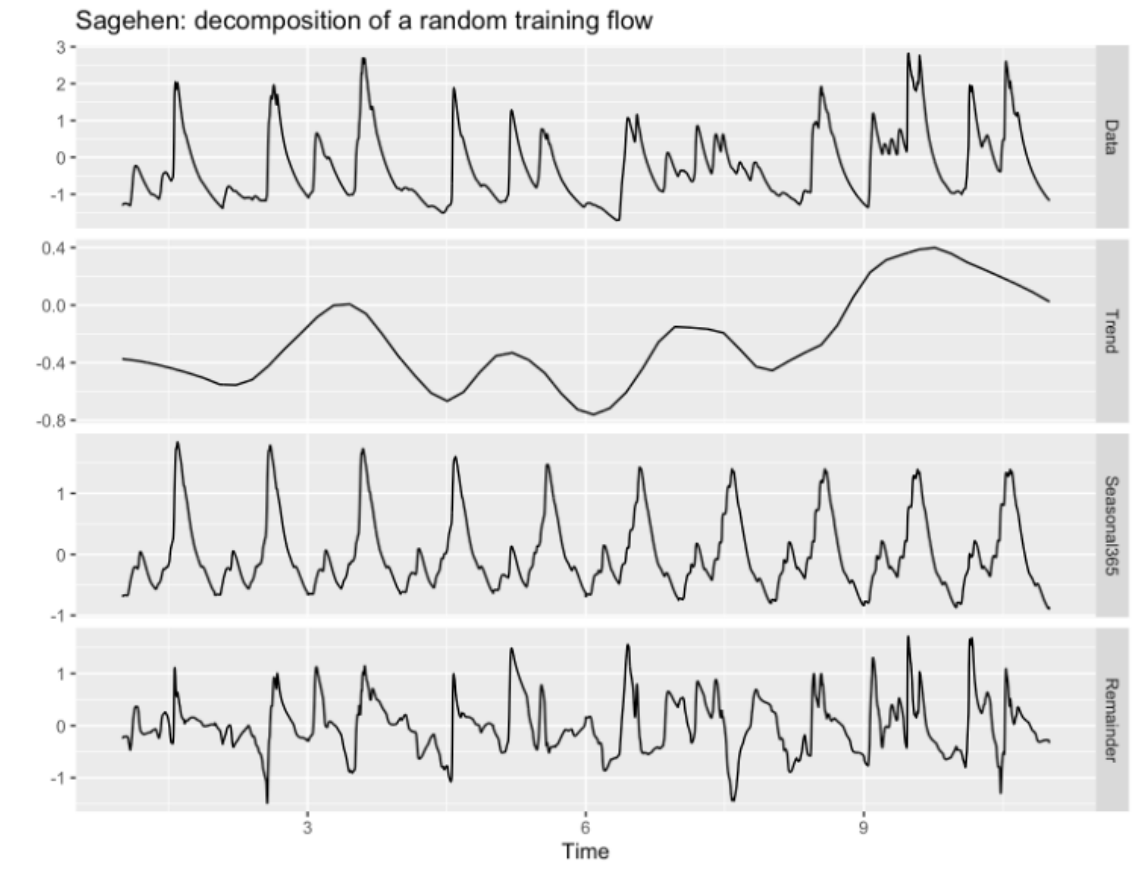

Figure 4.1: An example time series of transformed streamflow output from RHESSys for a single vector of soil input values (top), and an additive decomposition consisting of the trend (second row), the seasonal component (third row), and the remaining part (fourth row). The experimental site is Sagehen Creek and the data is a randomly selected realization of the hydrological simulator RHESSys. The decomposition is performed by using *stl* in R (R Core Team (2021))

In our modeling, we assume that the dominant temporal variability is a seasonal

pattern for all RHESSys time series outputs (e.g., due to seasonal weather changes). We model this seasonal behavior s^j for $t_j \in \mathbf{t}$ using a mixture of periodic processes with different amplitudes and frequencies, as follows:

$$s^j = \sum_{k=1}^{r} \Big(U_k \cos(2\pi\omega_k t_j) + V_k \sin(2\pi\omega_k t_j) \Big), \tag{4.2}$$

where U_k and V_k are independent random variables that have mean zero and variance σ_k^2, and ω_k denotes the frequency of the k^{th} process. r is the number of periodic components in this model, which can be determined using standard model selection methods.

It is further assumed that the d-dimensional input vector contains a value for each soil input dimension (variable) that was selected separately from the values of the other soil input dimensions. As such, for the i^{th} model run and the time instance t_j, the mean function for y_i^j is expressed as follows:

$$\mu_i^j = s^j + \sum_{q=1}^{d} \beta_q x_{iq}, \tag{4.3}$$

where s^j is defined in Equation (4.2) and the second term is a linear combination of the d-dimensional input with coefficient β_q for the q^{th} input x_{iq} in the i^{th} training simulator run. Note that since s^j characterizes a seasonal pattern shared by all the time series outputs, it does not carry the index of a specific simulator run. In addition, the second term does not depend on time. Therefore, it is not indexed by t_j.

Based on Equation (4.3), we can now specify the mean function as $\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^l)^\top$, which is the expectation of the stacked outputs $\mathbf{Y}$.

4.1.3 Covariance Function

To define the covariance function, we follow the specification of the stilt emulator from Olson et al. (2018), which is built upon the separability assumption from Rougier (2008). Specifically, for Gaussian Process $\mathcal{Y}$, we assume the covariance function is separable, in the sense that the GP covariance function $C(.,.)$ evaluated for any two d-dimensional soil input vectors (i.e., for any two points in soil parameter input space; here denoted $\mathbf{x}$ and $\mathbf{x}'$) and for any two time points (here denoted t and t') can be separated into the product of a spatial component and a temporal component. Specifically, we assume the covariance between GP $\mathcal{Y}$ at any domain value $(\mathbf{x}, t)$ and at any other domain value $(\mathbf{x}', t')$ can be decomposed as the product of a temporal covariance function and a spatial covariance function evaluated at these domain values, i.e.,

$$C\big((\mathbf{x}, t),\ (\mathbf{x}', t')\big) = C_t\big(t, t'\big) C_X\big(\mathbf{x}, \mathbf{x}'\big)$$

.

This separability implies that, when the GP covariance function is evaluated at every pair of RHESSys time points and every pair of soil parameter training set input vectors, the resulting covariance matrix can be written as the Kronecker product of two matrices; the first represents the covariance component across time points and the second the covariance across simulator inputs in d-dimensional soil parameter space. Specifically, we write the spatio-temporal covariance matrix $\mathbf{\Sigma}$ as follows:

$$\mathbf{\Sigma} = \mathbf{\Sigma}_t \otimes \mathbf{\Sigma}_X, \tag{4.4}$$

where $\mathbf{\Sigma}_t$ is the temporal covariance matrix computed between all pairs of time points and $\mathbf{\Sigma}_X$ contains the covariances between every pair of input settings in the training set.

The covariance across time is assumed to follow an AR(1) structure. As such, the $(i,j)^{th}$ element in $\mathbf{\Sigma}_t$ is given by

$$\sigma^{ij} = \frac{\rho^{|t_i - t_j|}}{1-\rho^2}\ , \tag{4.5}$$

where $\rho > 0$ is the first-order auto-correlation (i.e., the auto-correlation between the process at adjacent time points), and $|t_i - t_j|$ is the absolute difference (i.e., number of time units) between any two discrete time points t_i and t_j, for $i, j \in \{1, \ldots, l\}$. In our study, the units of time are days, so a time-lag of 1 corresponds to consecutive days; therefore for each $i \neq j$, $|t_i - t_j|$ is a strictly positive integer number of days. Since ρ is less than one, σ^{ij} decays rapidly if t_i and t_j are far from each other.

The covariance function of the process across the input domain (i.e., across d-dimensional input space) is modeled as a squared exponential and captures the output correlation as a function of input similarity. Specifically, when evaluated at the finite set of input values, the $(i,j)^{th}$ element in $\mathbf{\Sigma}_X$ is given by

$$\zeta_{ij} = v \exp\left(-\sum_{q=1}^{d} \frac{(x_{iq} - x_{jq})^2}{\phi_q}\right) + \tau\delta(i,j), \tag{4.6}$$

where v denotes the partial sill, ϕ_q is the range parameter for the q^{th} input, τ is the nugget; we refer readers to the original paper of the stilt emulator (Olson et al. (2018)) for detailed descriptions of these parameters. $(x_{iq} - x_{jq})^2$ is the squared Euclidean distance between the values of the q^{th} input variable in the i^{th} and j^{th} training runs. $\delta(i,j)$ is the Dirac delta function defined as follows:

$$\delta(i,j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}.$$

In summary, our model is an extension of the stilt emulator that additionally takes the seasonality of the physical process into account. Our emulator is characterized by parameters that describe the mean and the covariance functions, which are listed as follows:

$$\begin{aligned}\Pi = \{&\omega_k, \text{ for } 1 \leq k \leq r;\ U_k, \text{ for } 1 \leq k \leq r;\ V_k, \text{ for } 1 \leq k \leq r;\\ &\beta_q, \text{ for } 1 \leq q \leq d;\ \phi_q, \text{ for } 1 \leq q \leq d;\ \rho;\ v;\ \tau\}.\end{aligned} \tag{4.7}$$

In the next part, we describe how to estimate these model parameters based on the training simulator runs.

4.1.4 Emulator Parameter Estimation

Constructing our proposed emulator requires estimating the model parameters in Equation (4.7). First, we estimate the seasonal term in the mean function, as described in Equation (4.2). More specifically, we take the frequencies as multiples of 1/365, i.e., $w_k = k/365$. We then determine the number of such frequencies, r, using the Akaike's information criterion (AIC).

Given the frequencies $\omega_k's$, we continue to estimate the remaining parameters by following a Maximum Likelihood Estimation procedure that was used for estimating the stilt emulator (Olson et al. (2018)). The log-likelihood function of the parameters can be written as follows:

$$\ln L(\Pi \mid \mathbf{Y}) \propto (\mathbf{Y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{Y} - \boldsymbol{\mu}) + \ln |\boldsymbol{\Sigma}| \tag{4.8}$$

with the same uniform prior imposed on all the parameters. Recall that $\mathbf{Y}$ is the stacked outputs of the training simulator runs, $\boldsymbol{\mu}$ is a function of the input variables, and $\boldsymbol{\Sigma}$ is a function of the pairwise distances of input variables and the pairwise time intervals.

Note that it is computationally challenging to directly evaluate this likelihood due to the inversion of the covariance matrix, i.e., $\mathbf{\Sigma}^{-1}$, the complexity of which is $\mathcal{O}((ln)^3)$. As such, we utilize the work-around method proposed in Olson et al. (2018), which converts the inversion of the $nl \times nl$ matrix to the inversion of two smaller matrices, i.e., inverting a $l \times l$ matrix and a $n \times n$ matrix. This significantly reduces the computational cost.

4.1.5 Emulator Prediction Given New Input

In this part, we describe how to utilize our built emulator to predict the simulator output when a new, unseen test input is given. Suppose that the test input is $\mathbf{x}_i^*$. The corresponding l-dimensional simulator output $\mathbf{y}_i^*$ is approximated by a multivariate Gaussian distribution:

$$\mathbf{y}_i^* \sim \mathcal{N}(\boldsymbol{\mu}_i^*, \mathbf{\Sigma}_i^*), \tag{4.9}$$

where $\boldsymbol{\mu}_i^* \in \mathbb{R}^l$ is the GP mean function evaluated at this test input, with the j^{th} element $(\mu_i^j)^*$ standing for the mean at t_j ($j \in \{1, \ldots, l\}$) and computed by

$$(\mu_i^j)^* = \ s^j + \sum_{q=1}^{d} \beta_q x_{iq}^* + \Big[\left(\mathbf{I}_{l\times l} \otimes \Sigma_{\mathbf{x}_i^* X} \mathbf{\Sigma}_X^{-1} \right) (\mathbf{Y} - \boldsymbol{\mu}) \Big]^j. \tag{4.10}$$

It can be seen that the mean at t_j of the test output consists of the seasonal component, the linear combination of the input variables, and a third term that utilizes the training outputs whose inputs are similar to this test input.

The covariance matrix $\mathbf{\Sigma}_i^*$ is given by

$$\mathbf{\Sigma}_i^* = (v + \tau)\mathbf{\Sigma}_t - \mathbf{\Sigma}_t \otimes \Sigma_{\mathbf{x}_i^* X} \mathbf{\Sigma}_X^{-1} \Sigma_{\mathbf{x}_i^* X}^\top. \tag{4.11}$$

Once the mean and covariance functions are obtained according to the above equa-

tions, the predicted simulator output for the given test input is then fully specified by the Gaussian random variable in Equation (4.9). Note that when computing the inverse of $\mathbf{\Sigma}_X^{-1}$, the same linear algebra technique used in Equation (4.8) is again utilized to reduce computation costs.

4.2 Variance-based Sensitivity Analysis

We conduct sensitivity analysis to understand how the variability in the simulator output can be attributed to the different inputs. Specifically, we would like to quantify the individual contribution of each simulator input to the uncertainty in the output, as well as the contribution of higher-order interactions among the inputs. This can be achieved by varying all the input factors over their respective ranges and computing the sensitivity indices. This procedure is known as the *global sensitivity analysis (GSA)*.

Since it is computationally prohibitive to directly run a simulator for GSA, one can utilize the emulated outputs which are computationally efficient to obtain and closely approximate the simulator outputs; see Saltelli et al. (2008); Pianosi et al. (2016); Razavi et al. (2021). In our work, we study the global sensitivity of RHESSys inputs by computing the variance-based sensitivity measures based on the predicted outputs given by the emulator.

The procedure for conducting the sensitivity analysis is as follows:

- Draw quasi-random samples for each input using Sobol's method (Sobol, 1967);
- Obtain the predictions for the sampled inputs by running the emulator;
- Compute the first-order, second-order, and total-order sensitivity indices.

4.2.1 Quasi-random Sampling

For the quasi-random sampling used in the first step of our GSA, we use Sobol's low-discrepancy sequence, which is one type of quasi-random sequences (Sobol (1967)).[1] Low-discrepancy sequences are able to fill the space more rapidly with an even spread of the samples without creating clusters or un-sampled regions. In contrast, classical pseudo-random sampling can result in clusters and empty regions since each point is sampled independently. Sobol's sequences can be obtained easily and efficiently using standard computer software. In this work, we use the R package *sensobol* (Puy et al. (2021)) for generating the quasi-random samples.

4.2.2 Variance-based Sensitivity Measures

Variance-based sensitivity analysis utilizes variance to describe the uncertainty of the model output given the input. However, the existing variance-based sensitivity measures are proposed for models with scalar outcomes; see Saltelli et al. (2008, 2010); Puy et al. (2021); Sobol (1967). This cannot be directly applied to the time series outputs unless summary statistics are extracted. For example, one can obtain the quantiles of a time series and compute how the inputs contribute to the variation to the quantiles. In our work, we compute the standard sensitivity indices for each time step. Namely, the contribution of each soil input to the streamflow at the same time step is measured by the computed indices. We note that all sensitivity indices described below are completed with the R package (Puy et al. (2021)). In the following part, we provide a detailed derivation of the sensitivity measures.

First, we write the total variance of the model output at t_j as $V(\mathbf{y}^j)$ and decompose

[1] Other sampling strategies are also possible, e.g., random sampling, Latin hypercube sampling (Stein (1987); McKay et al. (2000); Saltelli et al. (2008)). Comparisons of these sampling methods can be found in Puy et al. (2021); Kucherenko et al. (2015).

it following Sobol's functional decomposition (Sobol (1993)):

$$V(\mathbf{y}^j) = \sum_{q=1}^{d} V_q^j + \sum_{q=1}^{d} \sum_{h>q}^{d} V_{qh}^j + \cdots + V_{12\ldots d}^j, \tag{4.12}$$

where V_q^j is the first-order effect of the q^{th} model input at t_j, V_{qh}^j is the second-order interaction between the q^{th} and h^{th} model inputs, and $V_{12\ldots d}^j$ is the interaction of all the inputs.

In Equation (4.12), the first-order effect V_q is defined by

$$V_q^j = \mathbb{V}_{X_q}[\mathbb{E}(\mathbf{y}^j|X_q)],$$

which measures the mean effect of the q^{th} input X_q to the variation in the output when the remaining inputs are fixed.

The second-order effect is defined as

$$V_{qh}^j = \mathbb{V}_{X_q,X_h}[\mathbb{E}(\mathbf{y}^j|X_q,X_h)] - \mathbb{V}_{X_q}[\mathbb{E}(\mathbf{y}^j|X_q)] - \mathbb{V}_{X_h}[\mathbb{E}(\mathbf{y}^j|X_h)],$$

which quantifies the joint effect of the pair of inputs X_q and X_h on $\mathbf{y}^j$. It can be seen that the last two terms in the second-order effect are indeed V_q^j and V_h^j. Note that all the expectations are taken over the explanatory variables that are not conditioned upon.

The variance-based measures use ratios to quantify the contribution of each term to the total variance. Based on the decomposition provided by Equation (4.12), we further define the first-order, second-order, and total-order indices at each time step. More specifically, we divide Equation (4.12) by $V(\mathbf{y}^j)$ on both sides. As a result, the

first-order sensitivity index S_q^j for $t_j \in \mathbf{t}$ is given by

$$S_q^j = \frac{V_q^j}{V(\mathbf{y}^j)} = \frac{\mathbb{V}_{X_q}[\mathbb{E}(\mathbf{y}^j|X_q)]}{V(\mathbf{y}^j)}. \tag{4.13}$$

The second-order sensitivity index S_{qh}^j for $t_j \in \mathbf{t}$ is given by

$$S_{qh}^j = \frac{V_{qh}^j}{V(\mathbf{y}^j)} = \frac{\mathbb{V}_{X_q,X_h}[\mathbb{E}(\mathbf{y}^j|X_q,X_h)] - V_q - V_h}{V(\mathbf{y}^j)}. \tag{4.14}$$

Furthermore, after division by the total variance, Equation (4.12) becomes

$$1 = \sum_{q=1}^{d} S_q^j + \sum_{q=1}^{d}\sum_{h>q}^{d} S_{qh}^j + \cdots + S_{12\ldots d}^j. \tag{4.15}$$

When $\sum_{q=1}^{d} S_q^j = 1$, the model is additive at t_j since the total variation in $\mathbf{y}^j$ can be fully explained by the first-order terms.

To quantify the contribution of an input to the output variance at all the orders, Homma and Saltelli (1996) proposed the total-order index. Extending this to the time series output, the index at time t_j is given by

$$T_q^j = 1 - \frac{\mathbb{V}_{X_{-q}}\left[\mathbb{E}\left(\mathbf{y}^j \mid X_{-q}\right)\right]}{V(\mathbf{y}^j)}, \tag{4.16}$$

which is the sum of the first-order effect and all high-order effects related to input X_q. X_{-q} represents all inputs except for the q^{th} input. Note that the expectation is taken over the variables that are not conditioned upon. The sum of total-order effects over the inputs are greater than or equal to one, i.e., $\sum_{q=1}^{d} T_q^j \geq 1$.

4.3 RHESSys Simulation Data

This section describes the RHESSys simulation runs that we use to build and assess our proposed emulator for two sites of interest, Sagehen Creek and Rattlesnake Canyon, which exhibit different climate characteristics. Sagehen Creek is located in the Sierra Nevada in Northern California. It is dominated by winter precipitation, with much of this falling as snow, and is almost blanketed in snow during the winter months (Meyers et al. (2010)). Rattlesnake Canyon is located in Southern California, also dominated by a winter precipitation pattern; it is usually dry throughout the year except for a few months where precipitation falls as rain.

The Tague Team Lab ran RHESSys to simulate the streamflows at the two sites and provided inputs and outputs for each site's training and testing sets for emulator development. In each run, the simulator takes six soil property inputs and time series forcing factors like carbon dioxide and generates a corresponding time series output, the streamflow at the site. The soil property inputs are summarized in Table 4.1. Note that the soil attributes are scalars. Figure 4.2 and 4.3 show the pairwise scatter plots of soil inputs at the two sites, respectively, which capture the pairwise correlation patterns between the soil inputs. The numerical correlation values are also shown in the figures. For each site, we obtain $n = 500$ simulation runs from RHESSys. When running RHESSys for the Rattlesnake Canyon site, the soil inputs are randomly sampled from their respective ranges, except for m for which 200 runs are randomly obtained from $[0, 1]$ and the other 300 from $[1, 20]$. When running the simulations for the Sagehen Creek site, the soil inputs are sampled based on a Sobol design (Sobol (1967)).

We show sample simulated time series streamflow outputs in Figure 4.4. For Sagehen Creek, the daily time series covers from 10/01/2008 to 09/30/2018 (10 years). For the Rattlesnake Canyon, the daily time series covers from 10/01/2003 to 09/30/2013 (10

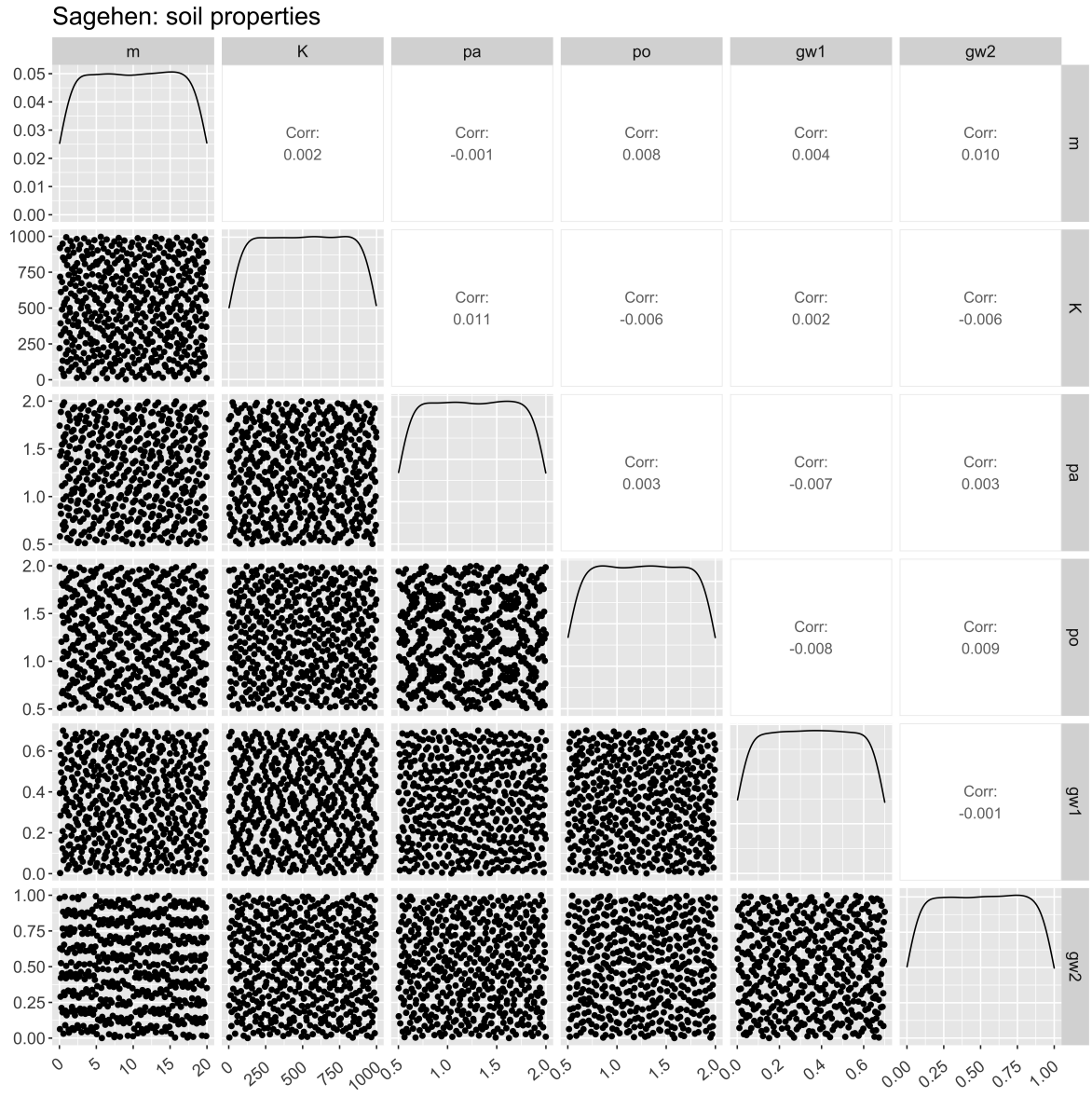

Figure 4.2: Pair-wise scatter plot of soil inputs used for running RHESSys for Sagehen Creek.

years). It can be seen that the streamflows present different patterns at the two sites. For instance, the flow level at Sagehen is overall higher than that at Rattlesnake since the Rattlesnake Canyon has a drier climate as compared to the Sagehen Creek. We also observe greater low streamflows at Sagehen Creek, partially due to the seasonal snowpacks that store winter precipitation and release meltwater into the early summer/dry season.

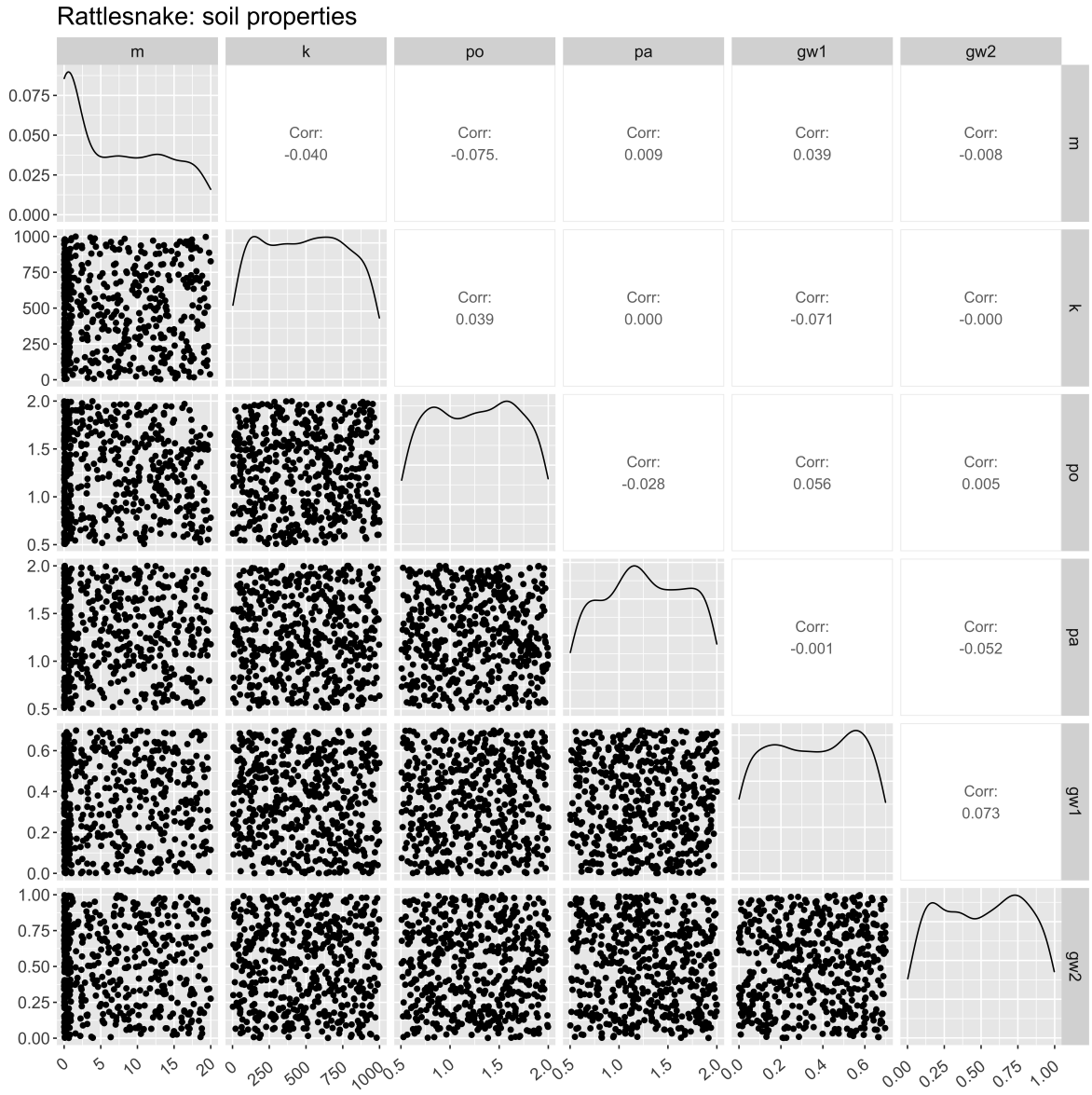

Figure 4.3: Pair-wise scatter plot of soil inputs used for running RHESSys for Rattlesnake Canyon.

For both sites, we can see that there exist clear seasonal patterns.

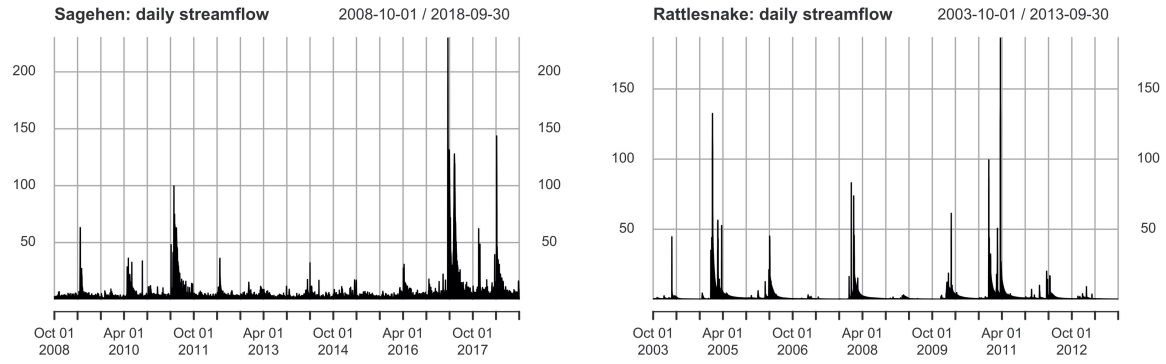

Figure 4.4: Sample streamflows simulated by RHESSys for Sagehen Creek (left) and Rattlesnake Canyon (right).

4.4 Emulation Results and Analysis

We apply our proposed emulator, which leverages the Gaussian Process with seasonality, in order to emulate/approximate RHESSys simulation results at each of Sagehen Creek and Rattlesnake Canyon. Using test RHESSys runs, we assess the accuracy of our emulators' predictions of RHESSys outputs and conduct sensitivity analysis to study the relative importance of each of the soil inputs.

4.4.1 Data Pre-Processing and Train-Test Split

We apply data pre-processing to the data of each site. First, we pre-process the input values. As shown in Figure 4.2 and 4.3, the soil inputs have different ranges. For instance, the input K has a range of $[0, 1000]$ while $gw1$ has a range of $[0, 0.7]$. As such, we scale each input into the $[0, 1]$ range. Next, we apply a site-specific transformation to the stacked simulation outputs for each of the sites, in order to improve the Gaussian Process approximation. Specifically, we estimate (and apply) a Box-Cox transformation (Box and Cox (1964)) to the stacked streamflow outputs, separately for each of the two

sites. The Box-Cox transformation is formalized as follows:

$$\tilde{y}_i^j(\lambda) = \begin{cases} \dfrac{(y_i^j)^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln y_i^j & \text{if } \lambda = 0. \end{cases} \tag{4.17}$$

For each location, the same λ value is applied to y_i^j for every simulator run i and each time instance t_j. The λ value is determined by maximizing the Gaussian likelihood.

Figure 4.5 shows the normal Quantile-Quantile plot of stacked Sagehen Creek streamflow outputs before and after performing the Box-Cox transformation. It can be seen that after the transformation, the streamflow output data has better normality.

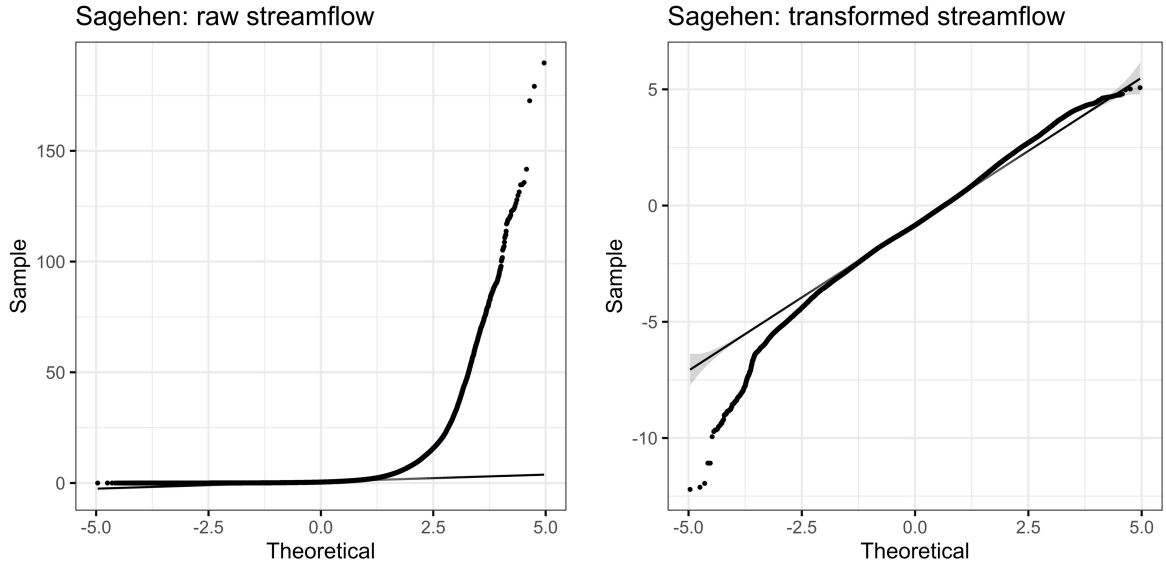

Figure 4.5: Normal Quantile-Quantile plot of stacked streamflow outputs of Sagehen Creek, for the raw data (left) and for the Box-Cox transformed version (right).

For each watershed, the RHESSys simulator runs are randomly partitioned into 80% training and 20% testing. We construct the emulator by estimating parameters using the training set and evaluate the emulator accuracy using the testing set of RHESSys runs for each site.

4.4.2 Performance Metrics

We evaluate the accuracy of emulation approximations using two metrics. Specifically, given any new test design $\mathbf{x}_i^*$ for $i \in \{1, \ldots, n^*\}$, we compute

- Mean absolute error (MAE):

$$MAE_i = \frac{1}{l}\|\hat{\mathbf{y}}_i^* - \mathbf{y}_i^*\|_1,$$

  where $\|\cdot\|$ is the vector norm, and l is the length of the time series output. MAE is the average of the absolute distance between the simulator output and the emulator prediction.

- Nash–Sutcliffe model efficiency coefficient (NSE):

$$NSE_i = 1 - \frac{\|\hat{\mathbf{y}}_i^* - \mathbf{y}_i^*\|_2^2}{\|\mathbf{y}_i^* - \bar{\mathbf{y}}_i^*\|_2^2}, \tag{4.18}$$

  where $\bar{\mathbf{y}}_i^*$ is the mean of the simulator output over time: $\bar{\mathbf{y}}_i^* = \frac{1}{l}\sum_{j=1}^{l} y_i^j$. The second term is the ratio of the squared error to the variance of the simulator output. A positive NSE value indicates that the emulator prediction can more accurately capture the simulator output as compared to the mean of the output.

4.4.3 Streamflow Emulation and Sensitivity Analysis

In this section, we present the results of emulating the streamflow simulated by RHESSys at Sagehen creek and Rattlesnake canyon, using our proposed emulation method. By utilizing the emulation outputs, which are efficient to obtain, we further conduct variance-based global sensitivity analysis to study how the soil conditions affect the streamflow.

Sagehen Creek: Emulation Results

The estimated parameters for the Sagehen streamflow emulator are summarized in Equation (4.19) and Table 4.4. Specifically, mean-related parameters ω_k, U_k, V_k for $k \in \{1, \ldots, 4\}$ and β_q for $q \in \{1, \ldots, 6\}$ are shown in the following equation:

$$\begin{aligned}\hat{\mu}_i^j = &-1.41 + 0.28x_{i1} - 0.13x_{i2} - 0.15x_{i3} + 0.01x_{i4} + 1.76x_{i5} - 0.58x_{i6} \\ &- 0.36sin(\frac{2\pi}{365}t_j) - 0.65cos(\frac{2\pi}{365}t_j) + 0.54sin(\frac{4\pi}{365}t_j) - 0.12cos(\frac{4\pi}{365}t_j) \\ &- 0.16sin(\frac{6\pi}{365}t_j) - 0.05cos(\frac{6\pi}{365}t_j) - 0.02sin(\frac{8\pi}{365}t_j) - 0.14cos(\frac{8\pi}{365}t_j),\end{aligned} \tag{4.19}$$

for $t_j \in \{1, 2, \ldots, 3653\}$ and $i \in \{1, \ldots, 500\}$.

Table 4.4: Covariance-related parameter estimates of Sagehen streamflow emulator

Site	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ρ	v	τ
Sagehen	0.37	10	3.60	3.25	0.99	1.19	0.94	0.01	0.02

The seasonal patterns present in the streamflow outputs are captured by the periodic trigonometric functions in Equation (4.19). Figure 4.6 compares our estimated periodic components of Equation (4.19) and the seasonal component of a randomly selected test streamflow output. The red curve shows the sum of the trigonometric components in Equation (4.19), and the black curve shows the seasonal component of the streamflow extracted via additive decomposition. It can be seen that the two match closely.

We evaluate the prediction performance of the Sagehen streamflow emulator with MAE and NSE. Figure 4.7 show the boxplots of MAE and NSE. It can be seen that our constructed emulator is able to accurately predict the simulator output in most cases, as indicated by the small MSE and the large NSE.

There are two test cases showing negative NSE values and large MAE values. For analysis, we compare the emulated and simulated streamflow for one of such test cases, as shown in Figure 4.8 (bottom). We see that the simulated streamflow is flat between

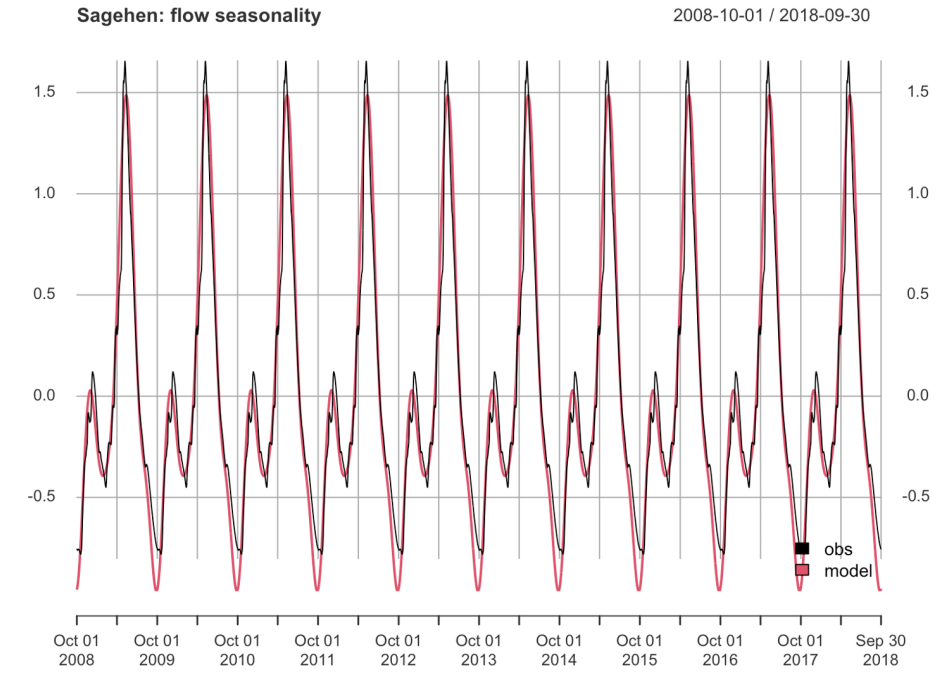

Figure 4.6: Our estimated seasonality term **s** (red) and the seasonal component in a randomly selected Sagehen streamflow simulation output.

October 1, 2008, and August 1, 2016, and the fluctuation occurs only during a short period of time towards the end of the time series. The simulated streamflow of the other test case with a negative NSE also exhibits a similar pattern. In fact, such a pattern is only present in these two test samples and not in the training set. Due to the absence of training data with similar characteristics, the constructed emulator is unable to explain such output patterns.

In order to further understand the prediction performance of the emulator, we examine a few more test cases that have positive but small NSE values. These cases exhibit a high fluctuating pattern, an example of which is shown in Figure 4.8 (top). We construct the empirical cumulative distribution function for each soil input based on the training set to determine what soil inputs may contribute to such flat or highly fluctuating streamflow outputs. We then find the quantile of every soil input for the test cases that display

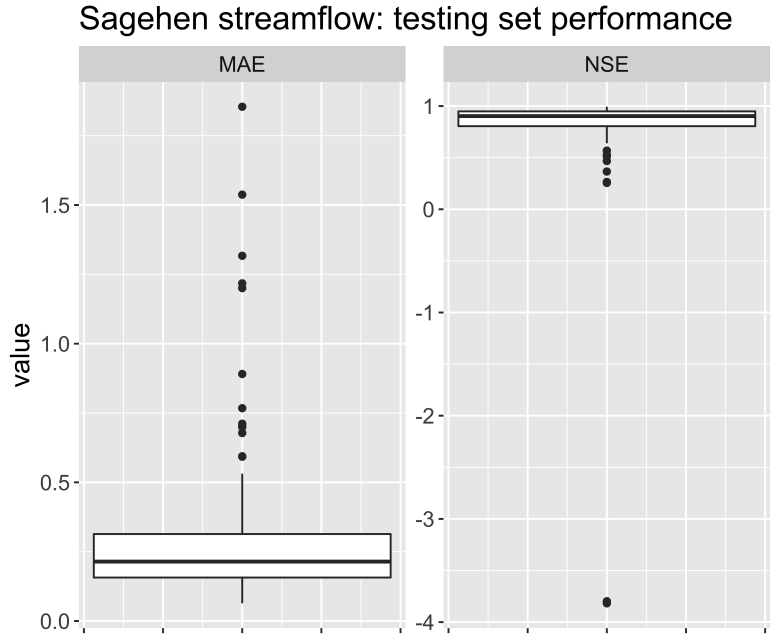

Figure 4.7: Boxplots of MAE (left) and NSE (right) from evaluating the Sagehen streamflow emulator on 100 unseen test simulation runs.

the aforementioned patterns. We discover that in all these test cases, the $gw1$ value is of low-quantile in the empirical function. Table 4.5 shows the un-normalized soil input values for these cases. It can be seen that all the $gw1$ values are very small, considering that this input ranges from 0 to 0.7. Therefore, a small $gw1$ can result in flat or highly fluctuating streamflows, challenging for the emulator to predict.

Table 4.5: Soil inputs in the raw scale (no normalization) for Sagehen streamflows that exhibit low-variation or high-fluctuation patterns

Site	m	K	pa	po	$gw1$	$gw2$	Pattern
Sagehen $\mathbf{x}^*_{371}$	0.10	717.66	1.43	0.90	**0.00**	0.63	highly fluctuating
Sagehen $\mathbf{x}^*_{461}$	16.70	776.04	0.74	1.19	**0.01**	0.48	highly fluctuating
Sagehen $\mathbf{x}^*_{261}$	12.61	265.05	1.88	0.59	**0.01**	0.07	little variation
Sagehen $\mathbf{x}^*_{79}$	18.07	844.25	1.06	1.43	**0.02**	0.11	little variation

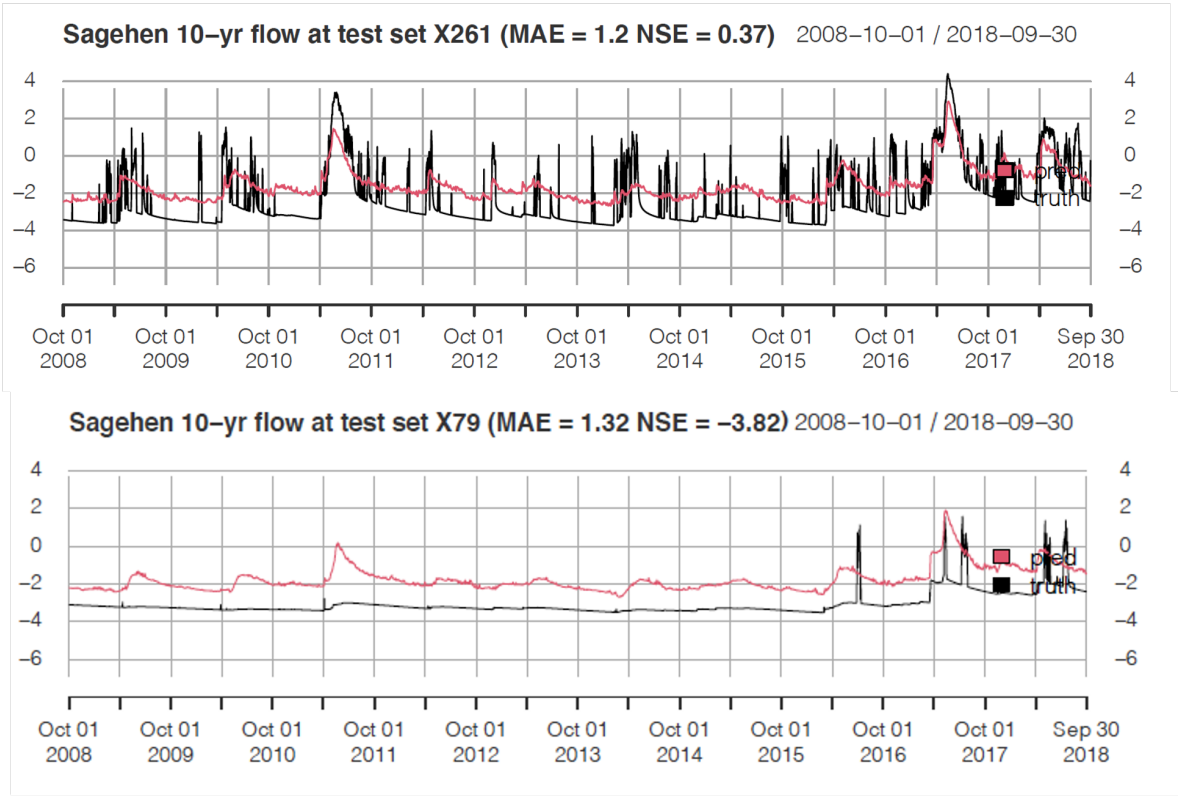

Figure 4.8: Challenging streamflow patterns in test simulation cases (black) and the corresponding emulator predictions (red). The top figure shows the highly fluctuating pattern, which the emulator cannot capture properly. The bottom figure shows nearly flat streamflow, for which the emulator prediction is off approximately by a constant.

Sagehen Creek: Sensitivity Analysis

In order to more systematically understand how soil inputs impact the streamflow, we conduct sensitivity analysis by utilizing the emulator's predictions and computing the variance-based measures as described in Section 4.2.

First, we assess the first-order effects of the soil inputs. Figure 4.9 shows Saltelli's first-order indices computed for each soil input throughout the ten-year time frame and the associated 95% confidence intervals. It can be seen that the mean effect of $gw1$ is the largest as Saltelli's first-order indices are close to 1 and considerably higher than those of the other soil inputs. In other words, the first-order effect of $gw1$ can explain away

the most variance of the streamflow. This finding reinforces our previous discussion that identifies $gw1$ as an important soil input at Sagehen creek. As for the other soil inputs, we see that m gives the second largest first-order effect and $gw2$ has a small first-order effect, while the remaining inputs have a near-zero first-order effect.

Next, we conduct a total-order sensitivity analysis. Figure 4.10 shows the total-order contribution due to each soil input. It can be seen that the indices of $gw1$ and m are the highest among all the soil inputs throughout the ten-year period. This indicates that streamflow variation can be mostly accounted for by $gw1$ and m through their first-order effects as well as all the higher-order effects. This again confirms that $gw1$ and m are key soil inputs contributing to the streamflow variations. In addition, we see that the total-order indices of $gw2$ are also large. On the other hand, we can identify K, pa, and po as non-influential soil factors to the uncertainty in streamflow, since the total-order measures are very close to zero and the confidence intervals are very tight throughout the entire time frame.

Finally, we compute Sobol's second-order indices for interactions between every pair of the soil inputs, which are shown in Figure 4.11. It can be seen that for the important soil inputs $gw1$, m, and $gw2$, the second-order effects are not very obvious as the indices are close to zero most of the time. In addition, the second-order indices for K, pa, and po are zero and the corresponding confidence intervals are narrow.

Overall, the uncertainty in Sagehen streamflow can be mostly explained by the first-order and higher-order effects due to $gw1$, m, and $gw2$, whereas the second-order effects do not contribute significantly to explaining the variations.

Rattlesnake Canyon: Emulation Results

We provide the estimated parameters for the Rattlesnake streamflow emulator in Equation (4.20) and Table 4.6. Specifically, the mean-related parameters ω_k, U_k, V_k for

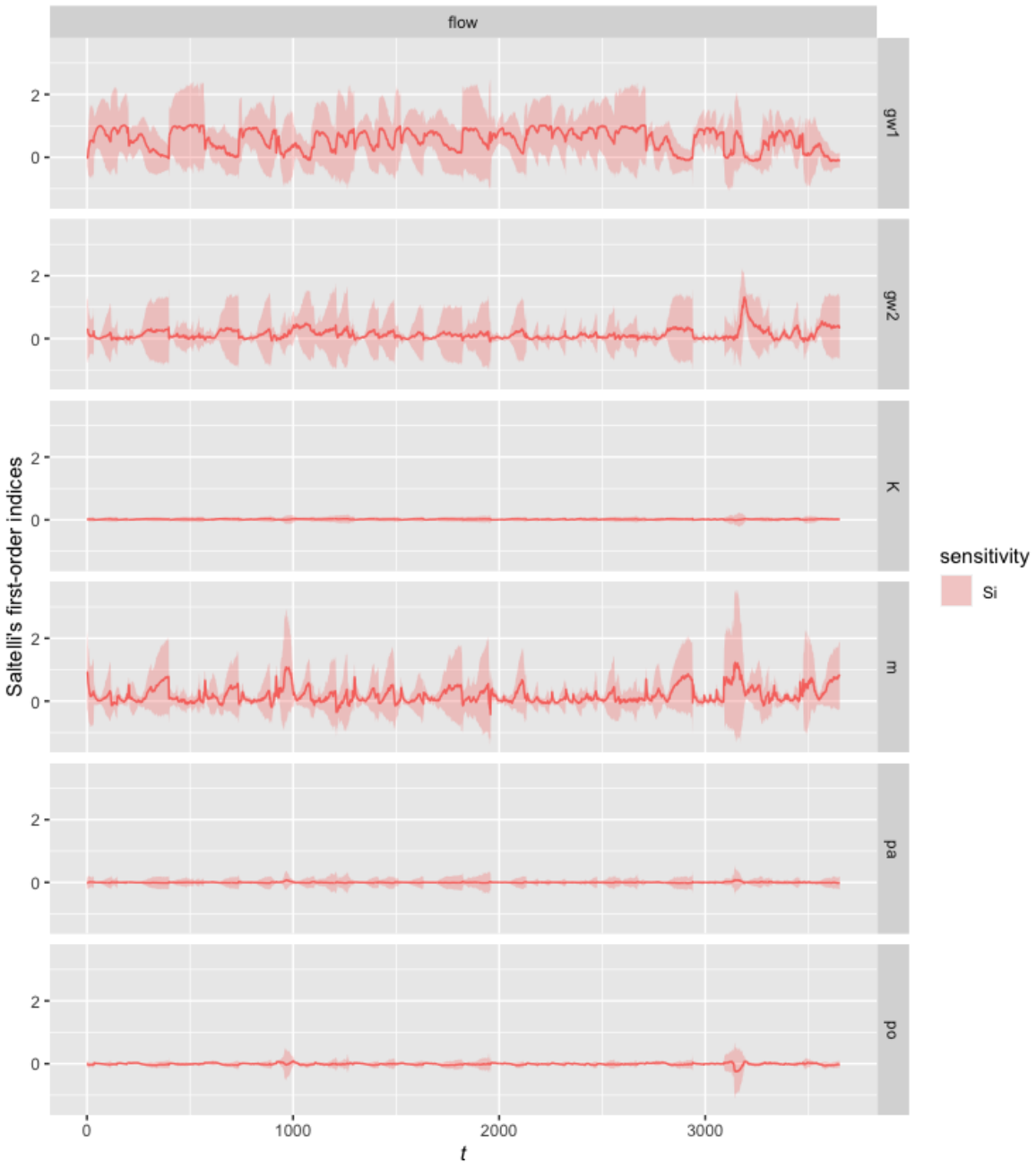

Figure 4.9: Saltelli's first-order sensitivity index and 95% confidence interval computed for the soil inputs ($gw1$, $gw2$, K, m, pa, po) based on the emulated Sagehen streamflow, for the ten-year period. Solid lines indicate the sensitivity measure and the shaded areas indicate the 95% confidence interval.

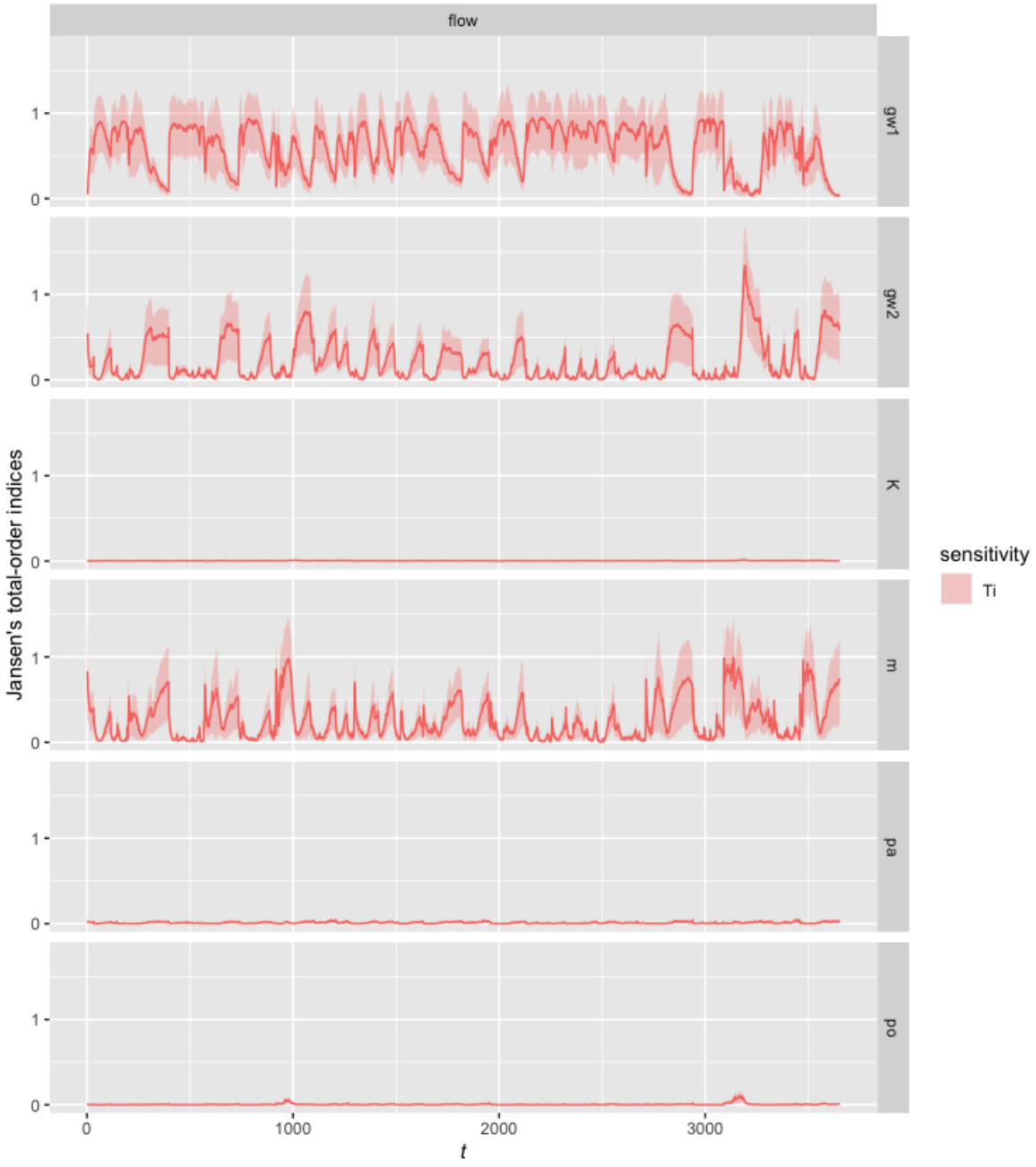

Figure 4.10: Jansen's total-order sensitivity index and 95% confidence interval computed for the soil inputs ($gw1$, $gw2$, K, m, pa, po) based on the emulated Sagehen streamflow. Solid lines indicate the sensitivity measure and the shaded areas indicate the 95% confidence interval.

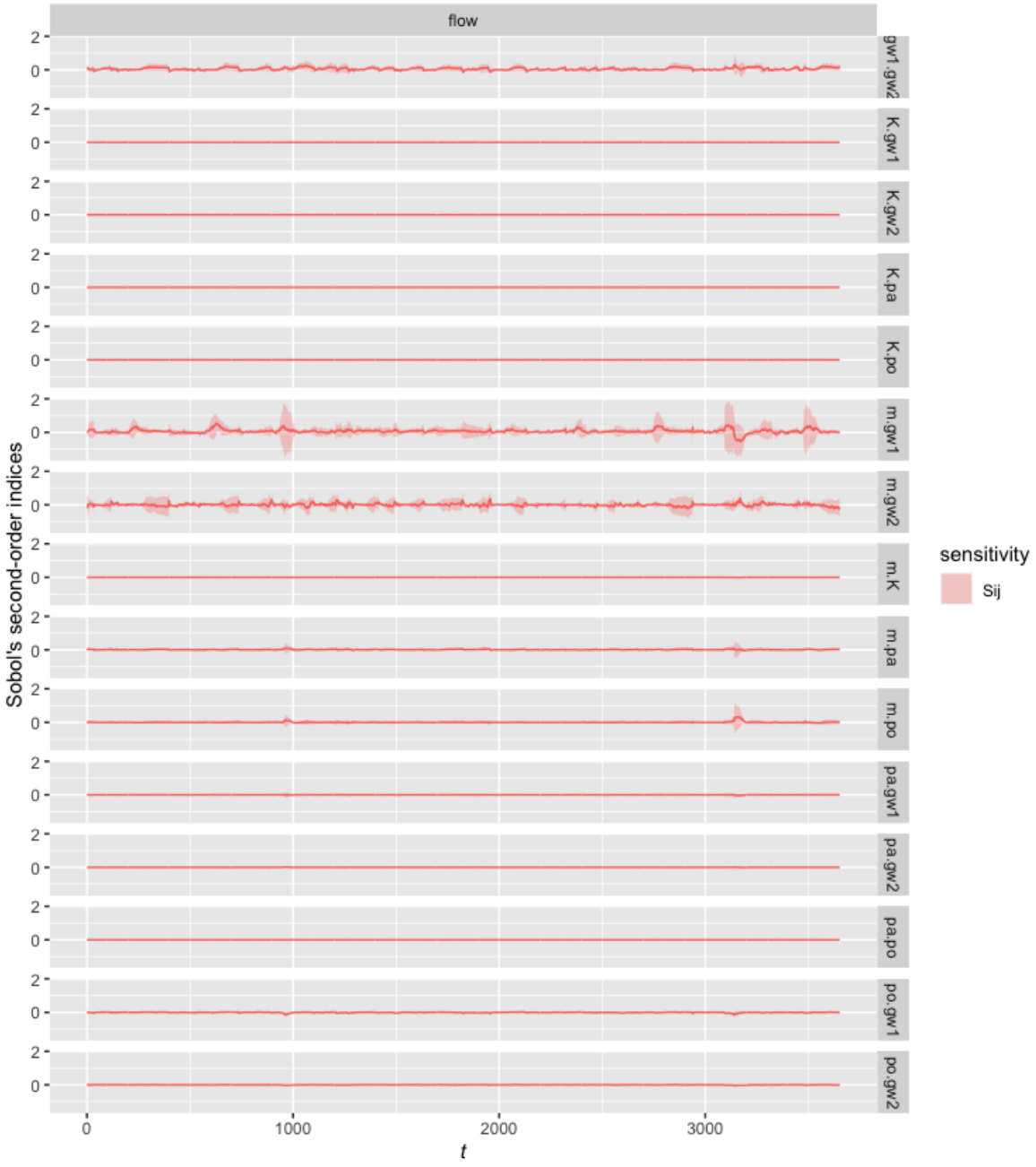

Figure 4.11: Sobol's second-order sensitivity index and 95% confidence interval computed for soil inputs ($gw1$, $gw2$, K, m, pa, po) based on emulated Sagehen streamflow emulator. Solid lines indicate the sensitivity measure and the shaded areas represent the 95% confidence interval.

$k \in \{1, \ldots, 4\}$ and β_q for $q \in \{1, \ldots, 6\}$ are shown in the following equation:

$$\begin{aligned}\hat{\mu}_i^j = &-2.16 - 0.97x_{i1} - 0.59x_{i2} - 0.41x_{i3} + 0.45x_{i4} + 1.12x_{i5} - 1.63x_{i6} \\ &+ 1.39sin(\frac{2\pi}{365}t_j) - 2.12cos(\frac{2\pi}{365}t_j) + 0.04sin(\frac{4\pi}{365}t_j) - 0.06cos(\frac{4\pi}{365}t_j) \\ &+ 0.03sin(\frac{6\pi}{365}t_j) - 0.06cos(\frac{6\pi}{365}t_j) + 0.12sin(\frac{8\pi}{365}t_j) - 0.06cos(\frac{8\pi}{365}t_j)\end{aligned} \tag{4.20}$$

for $t_j \in \{1, 2, \ldots, 3653\}$ and $i \in \{1, \ldots, 505\}$.

Table 4.6: Covariance-related parameter estimates of Rattlesnake streamflow emulator

Site	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ρ	v	τ
Rattlesnake	0.13	1.89	4.30	2.88	2.04	1.00	0.97	0.20	0.03

The sum of the trigonometric functions in Equation 4.20 captures the seasonal pattern in the Rattlesnake streamflow, which is extracted by additive decomposition. Figure 4.12 compares our estimated seasonal term (red) with the seasonal component of a randomly selected Rattlesnake streamflow (black). It can be seen that the two match closely, except for the fast fluctuations around the peaks. Such fast fluctuations can be captured by including trigonometric functions with higher frequencies, which, however, increases the complexity of the model.

We evaluate the prediction performance of the Rattlesnake streamflow emulator with MAE and NSE. Figure 4.13 shows the boxplots of the MAE and NSE values over the test cases. It can be seen that our emulator is accurate since the MAE values are small and the NSE values are close to one for most of the test cases. We observe seven cases where the NSE values are negative and the MAE values are large. This is because the streamflows for these seven sets of soil inputs exhibit only slight variation or are almost flat throughout the entire time period; an example is shown in Figure 4.14. In order to find the most relevant soil input contributing to such patterns, we construct the empirical cumulative distribution function and compute the quantiles of each soil input. We find

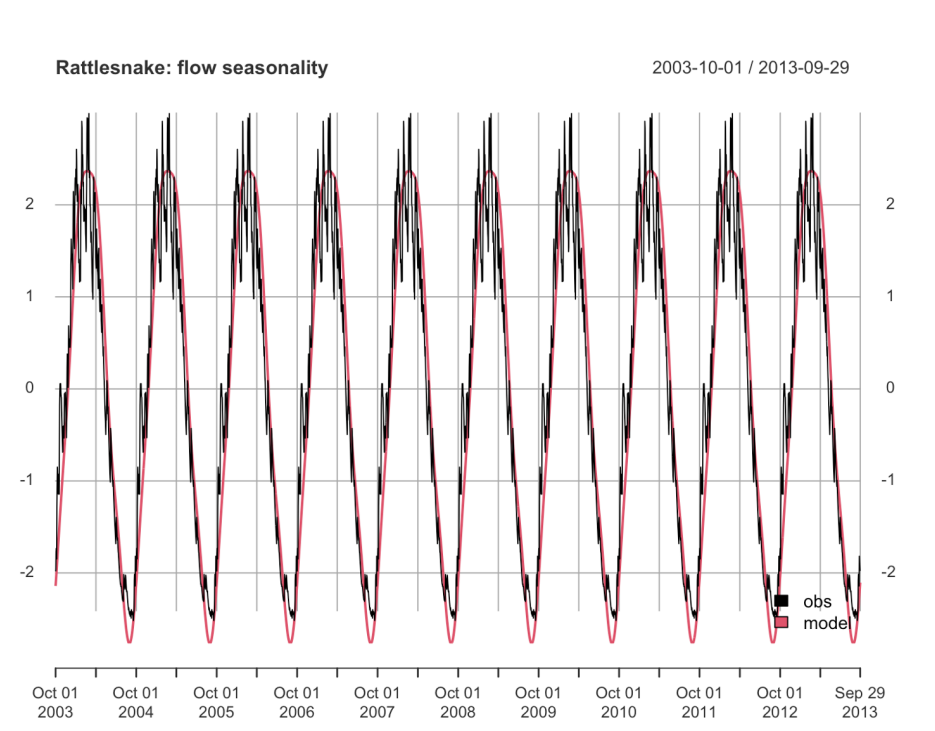

Figure 4.12: Our estimated seasonality term **s** (red) and the seasonal component in a randomly selected Rattlesnake streamflow simulation output.

that these cases all have small K values. The un-normalized soil inputs corresponding to the three test cases with the most negative NSE values are listed in Table 4.7. It can be seen that they all have very large K values.

Table 4.7: Soil inputs in the raw scale (no normalization) for Rattlesnake streamflows that exhibit low-variation patterns

Site	m	K	pa	po	$gw1$	$gw2$	Pattern
Rattlesnake $\mathbf{x}^*_{169}$	7.04	**21.38**	1.66	1.27	0.23	0.63	little variation
Rattlesnake $\mathbf{x}^*_{270}$	115.17	**9.86**	1.20	1.07	0.62	0.93	little variation
Rattlesnake $\mathbf{x}^*_{190}$	6.31	**15.99**	1.58	1.36	0.19	0.39	little variation

Rattlesnake Canyon: Sensitivity Analysis

In this part, we conduct sensitivity analysis to understand how the uncertainty in the Rattlesnake streamflows is attributed to the soil factors.

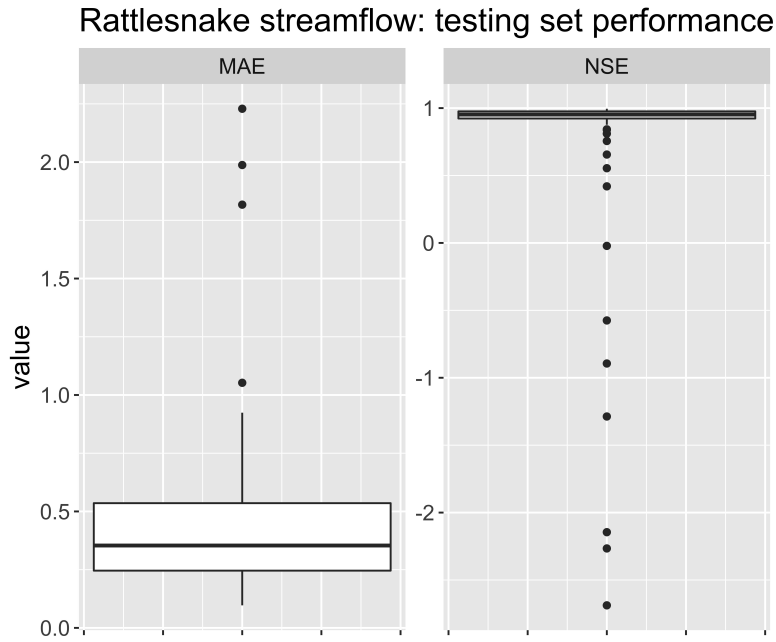

Figure 4.13: Boxplots of MAE (left) and NSE (right) from evaluating the Rattlesnake streamflow emulator on 100 unseen test simulation runs.

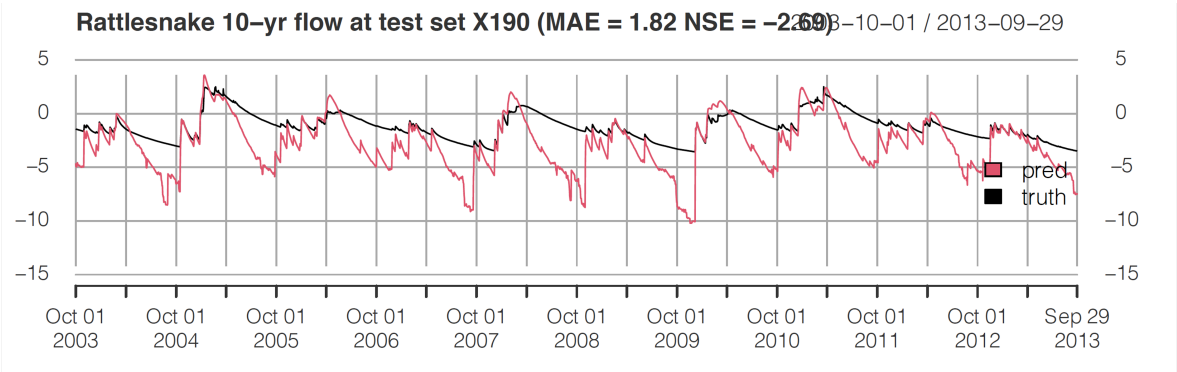

Figure 4.14: Challenging streamflow patterns in test simulation cases (black) and the corresponding emulator predictions (red). The streamflow exhibits only slight variations throughout the entire time period.

Figure 4.15 shows Saltelli's first-order indices computed for each soil input throughout the ten-year period and the corresponding 95% confidence intervals. Similar to Sagehen streamflow, we find that $gw1$ and m have the largest first-order effects. The remaining soil inputs do not exhibit a significant first-order influence on the Rattlesnake streamflow.

Next, we show the total contribution including the first-order and high-order effects

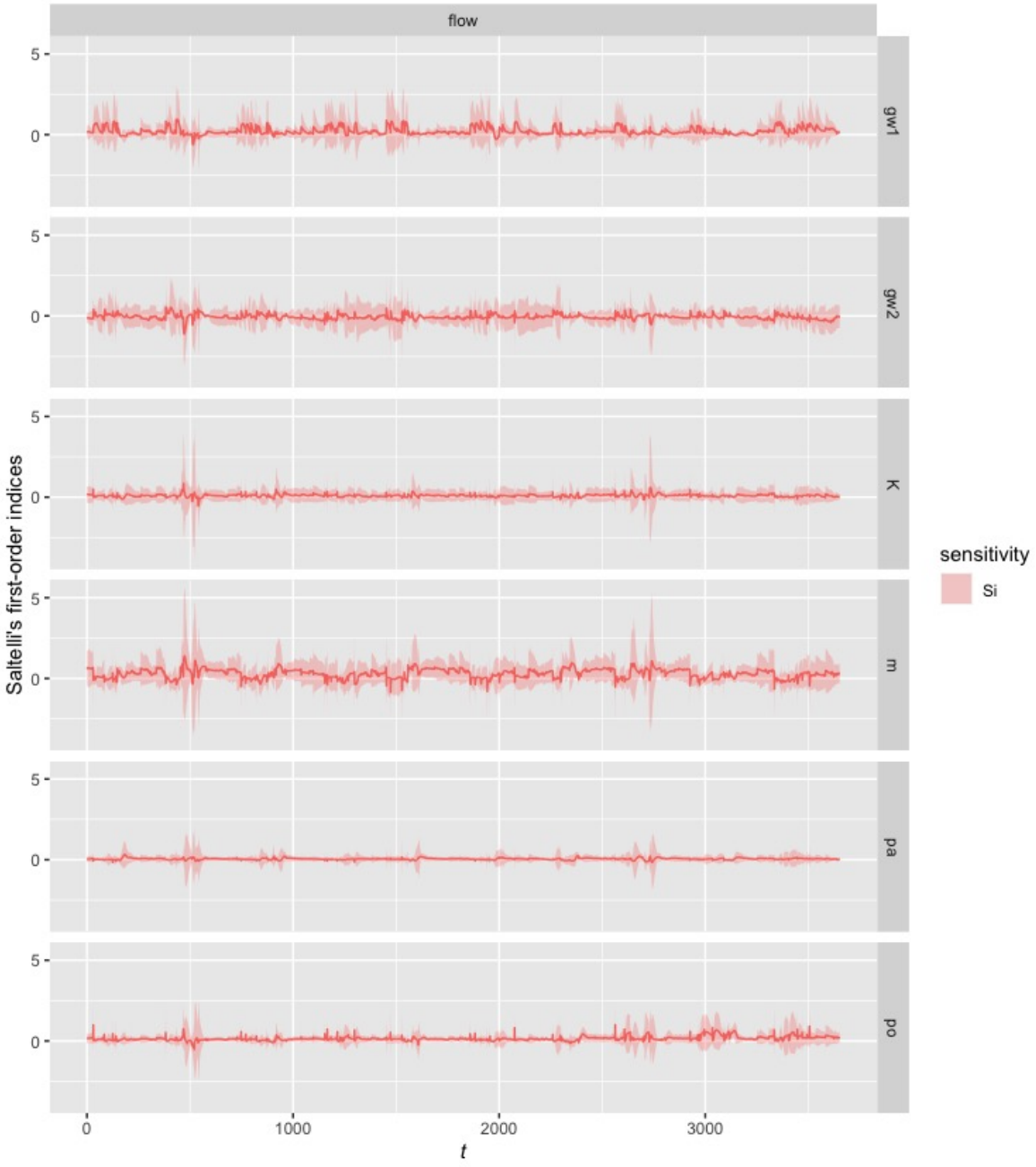

Figure 4.15: Saltelli's first-order sensitivity index and 95% confidence interval computed for soil inputs ($gw1$, $gw2$, K, m, pa, po) based on the emulated Rattlesnake streamflow. Solid lines indicate the sensitivity measure and the shaded areas indicate the 95% confidence interval.

in Figure 4.16 for each soil input. We see that Jansen's total-order indices are the largest for m, followed by $gw1$ and $gw2$, and the indices are smaller for K, po, and pa but still

have non-zero values. Since the respective mean effects of $gw2$, K, pa, and po are small but their total-order indices are larger, the higher-order interactions involving these soil inputs contribute to uncertainty in the streamflow.

Lastly, Figure 4.17 shows the Sobol's second-order indices for all pairwise interactions of the soil inputs. Since m has the largest total-order effect and the second-largest first-order effect, we see non-zero variations in all the pairwise indices involving m. In particular, the second-order indices for the interaction between m and $gw2$ are the largest, followed by the one between m and $gw1$, among all the pairwise combinations of soil inputs. The second-order indices are numerically zero and have tight confidence intervals for interactions involving pa and po.

To summarize, pa and po are non-influential soil inputs for explaining the uncertainty in the Rattlesnake streamflow. The variation in the outcome can be attributed to the first-order effects of $gw1$ and m, the second-order effects of the interactions between m and $gw2$ and between m and $gw1$, and higher-order interactions.

4.5 Summary

In this chapter, we used Gaussian Process-based emulators to approximate a hydro-ecological simulator RHESSys. This study aims to understand the relationship between the soil properties and the streamflow at two watersheds, Sagehen Creek and Rattlesnake Canyon. We also provided global sensitivity analysis for identifying influential and non-influential soil properties to explain the variation in streamflow at both watersheds.

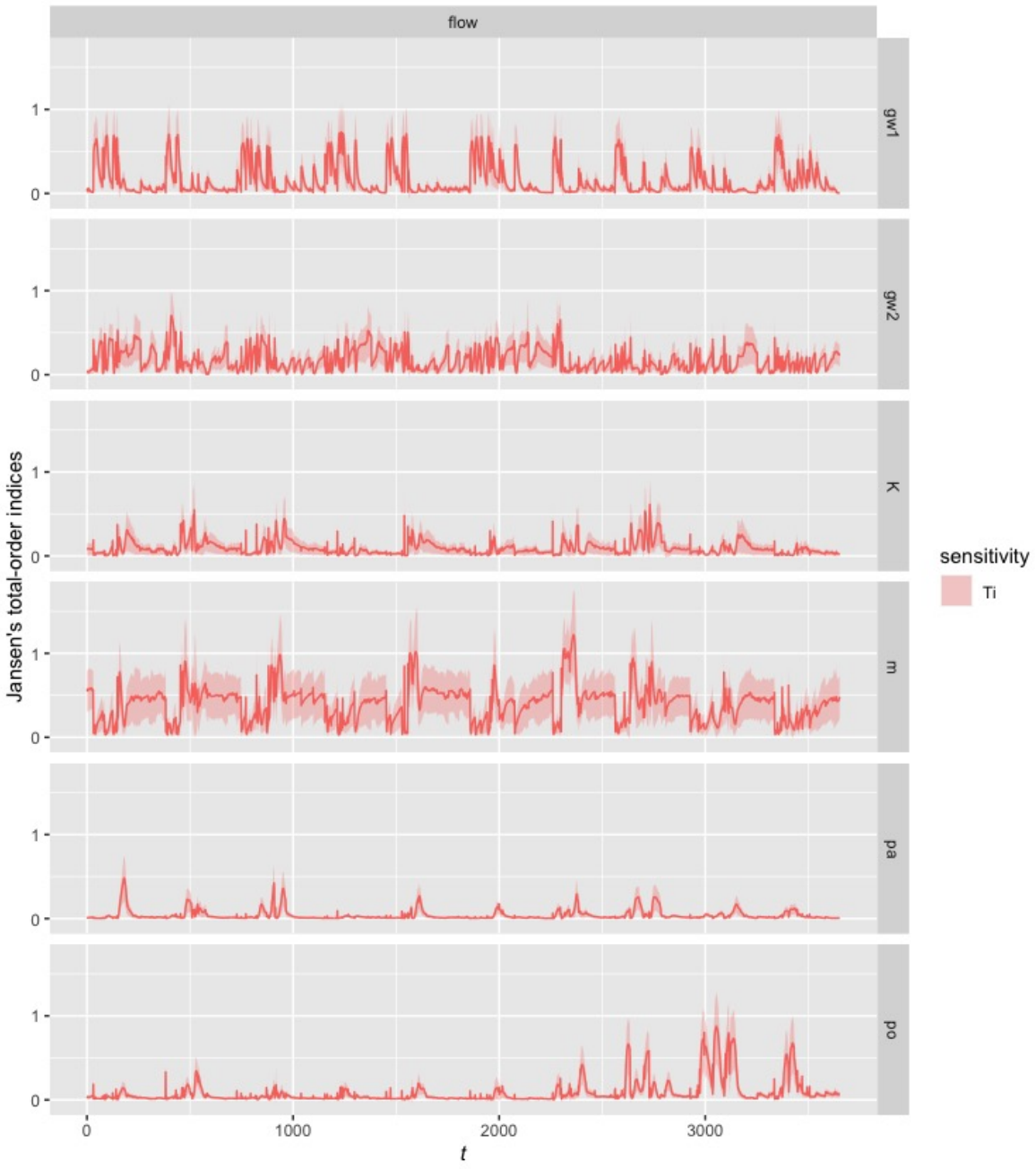

Figure 4.16: Jansen's total-order sensitivity index and 95% confidence interval computed for soil inputs ($gw1$, $gw2$, K, m, pa, po) based on the emulated Rattlesnake streamflow. Solid lines indicate the sensitivity measure and the shaded areas indicate the 95% confidence interval.

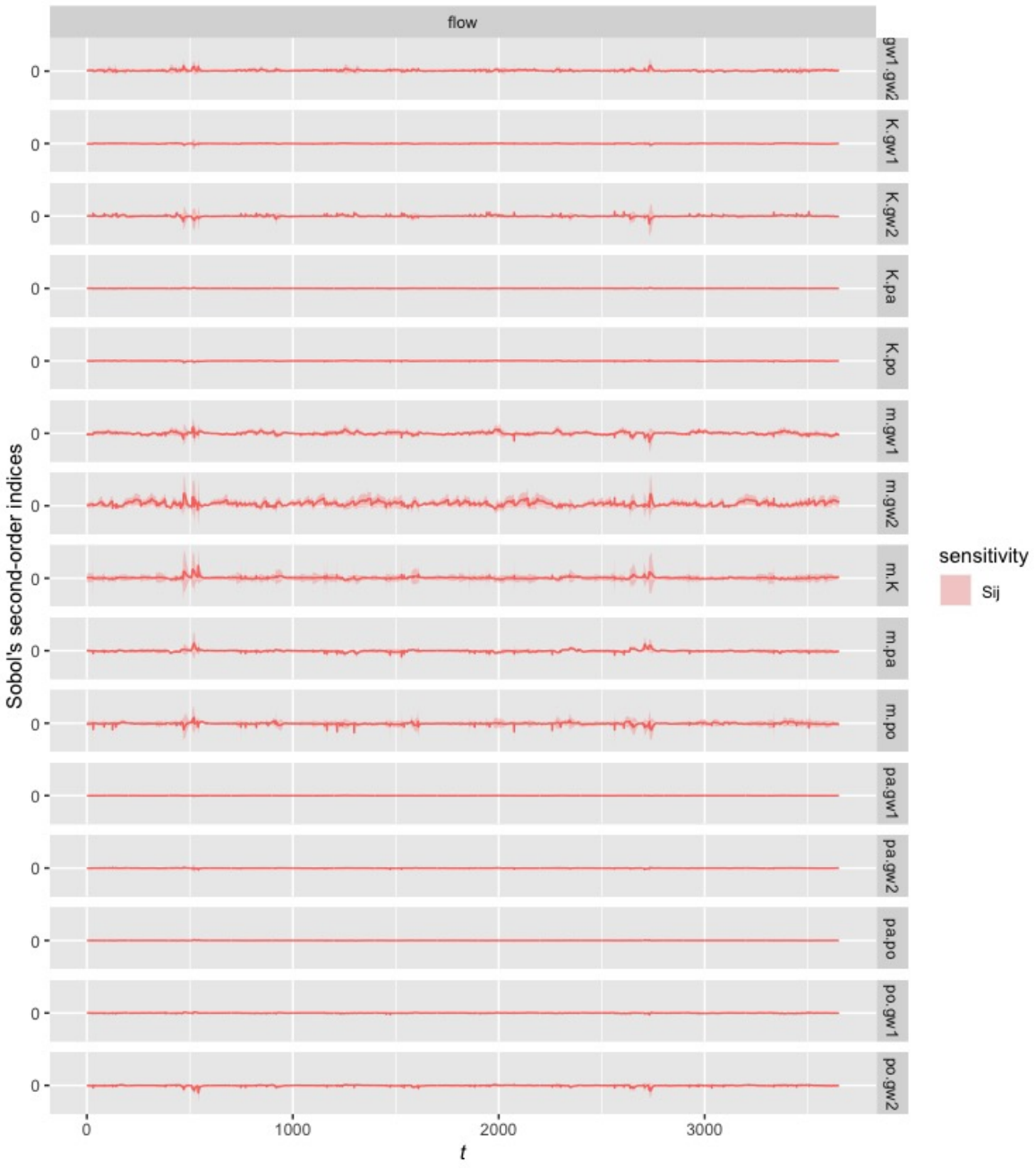

Figure 4.17: Sobol's second-order sensitivity index and 95% confidence interval computed for soil inputs ($gw1$, $gw2$, K, m, pa, po) based on the emulated Rattlesnake streamflow. Solid lines indicate the sensitivity measure and the shaded areas indicate the 95% confidence interval.

Chapter 5

Conclusions and Future Studies

5.1 Block-wise Robust Selection

In chapter 2, we proposed block-wise robust selection (BRS) for selecting the regularization parameter in the Gaussian graphical model. BRS enables adaptive regularization for different blocks of entries in the inverse covariance matrix. BRS first uses binary segmentation to group the variables based on their sample standard deviations. Then, the inverse covariance matrix and the penalty parameter matrix are divided into blocks according to the obtained variable grouping. In order to find the regularization parameters for the different blocks, BRS first determines a baseline block based on the block-wise robust Wasserstein profile function and computes the regularization parameter for this baseline block. The block-wise penalty matrix is then constructed based on this baseline parameter and a weighing scheme capturing data dispersion. By doing this, BRS avoids data normalization and only has one hyper-parameter with a range of $[0, 1]$, making it easy to tune. Furthermore, BRS is computationally efficient and specifically, the computational cost of our proposed BRS tuning method is significantly lower than k-fold cross-validation.

We conducted extensive simulation studies to validate the performance of our proposed BRS method. In these simulation experiments, we considered three settings, with

the ground-truth graphs being one power-law network and one Erdős-Rényi network. The graph recovery performance of our method was evaluated with two metrics, F_1 score and Matthews correlation coefficient (MCC). The results demonstrated that our proposed BRS approach could achieve higher MCC and F_1 scores across different scenarios when compared to two variants of the robust selection algorithm.

5.2 Paleo-Climate Reconstruction Using Block-wise Robust Selection

In Chapter 3, we further utilized our proposed BRS tuning method for two environmental applications related to paleo-climate field reconstruction. The HadCRUT4 datasets provide 100-ensemble members of historical surface temperature, containing approximately 60% of missing values. We proposed reconstructing such paleo-climate by treating it as a missing value imputation task. The temperature fields in the pre-instrumental need to be estimated based on information provided in more recent years. To this end, we modeled the spatial grid over the globe by a Gaussian graphical model and discovered the conditional correlations among them by estimating the model parameters and obtaining the adjacency graph. In this step, the Gaussian graphical model is tuned by BRS. Then, the past temperature fields are estimated based on a modified expectation-maximization algorithm that embeds the estimated Gaussian graphical model. Our experimental results demonstrated that the reconstruction based on the BRS graph is similar to the reconstruction based on a graph selected by environmental scientists, while saving achieving significant computational efficiency.

5.3 Emulation of RHESSys

In chapter 4, we emulated a mechanistic simulator, the Regional Hydro-Ecologic Simulation System (RHESSys), and provided a global sensitivity analysis (GSA) of it. This study aims to explore the relationship between the soil properties (RHESSys inputs) and streamflow (RHESSys output) at two watersheds with different climate conditions. Sagehen Creek, located in the Sierra Nevada in Northern California, is dominated by winter precipitation (primarily as snow); Rattlesnake Canyon, located in Southern California, is also dominated by winter precipitation (rain) but is usually dry for the majority of each year. The Tague Team Lab provides the training and testing data for studying and emulating the two watersheds.

First, we developed Gaussian Process (GP)-based emulators for RHESSys at the two watersheds. Both emulators were trained using 80% of the provided RHESSys simulator runs, with the remaining used for emulator assessment. To develop the emulator, we assume the emulation time series (which interpolates RHESSys time series at the training inputs) are generated by a Gaussian Process, where the mean is a linear combination of the soil properties and trigonometric functions, and the process has a separable covariance function. When evaluated at a finite set of time points and any finite set of soil input vectors, we modeled the covariance matrix of the process as a Kronecker product of temporal and spatial covariance matrices. Here, the spatial domain is the multi-dimensional space of soil parameter input values.

The parameter estimation we presented for the emulator was maximum likelihood, providing reasonable computational efficiency for our goals. We then computed mean absolute errors (MAE) and Nash–Sutcliffe model efficiency coefficients (NSE) for evaluating test set predictions using the remaining 20% of RHESSys simulator results, to assess how well our emulators perform in each watershed. The Sagehen Creek and Rattlesnake

Canyon emulators showed close-to-zero MAE and close-to-one NSE boxplots, which indicated that the test set predictions were "close" to the RHESSys simulated outputs based on the number of training runs used. Further assessment is ongoing when training the emulator using differing numbers of RHESSys training runs. We observed a few cases where the predicted streamflow deviates substantially from the RHESSys output for each emulator. We found that $gw1$ was close to zero in those cases at Sagehen Creek and K was large in those cases at Rattlesnake Canyon. We did not expect the emulator to perform well under inputs far from those in the training set, motivating continued research into efficient/effective design of training set inputs under various emulator assessment criteria.

Second, we provided variance-based measures to quantify the sensitivity of streamflow to each soil property at both watersheds. We used Sobol's low-discrepancy sequences (quasi-sampling) for conducting the analysis. Since our emulators provide reasonable predictions/approximations to key properties of RHESSys streamflow based on the training/testing sets used, we used the emulator predictions to compute sensitivity measures. Table 5.1 summarizes the identified influential soil properties for each watershed, that may help inform input designs for future RHESSys studies, and foci for evaluating RHESSys emulators.

Table 5.1: Influential soil properties identified by each sensitivity measure for Sagehen Creek and Rattlesnake Canyon

Sensitivity Measure	Sagehen Creek	Rattlesnake Canyon
First-order	$gw1$, m, $gw2$	$gw1$, m
Total-order	$gw1$, m, $gw2$	m, $gw1$, $gw2$, K, pa, po
Second-order	$m * gw1$, $m * gw2$, $gw1 * gw2$	m-related pairwise interactions

Bibliography

M. Ahmed, P. J. Krusic, F. Charpentier Ljungqvist, E. Zorita, et al. Continental-scale temperature variability during the past two millennia. *Nature Geoscience*, 6(5):339–346, 2013.

I. Azzini, T. Mara, and R. Rosati. Monte Carlo estimators of first-and total-orders Sobol's indices. *arXiv preprint arXiv:2006.08232*, 2020.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

S. Boughorbel, F. Jarray, and M. El-Anbari. Optimal classifier for imbalanced data using Matthews correlation coefficient metric. *PloS One*, 12(6):e0177678, 2017.

G. E. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

P. Brohan, J. Kennedy, I. Harris, S. Tett, and P. Jones. Uncertainty estimates in regional and global observed temperature changes: A new data set from 1850. *Journal of Geophysical Research: Atmospheres*, 111(D12), 2006.

D. Chicco and G. Jurman. The advantages of the Matthews correlation coefficient (MCC) over F_1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1): 1–13, 2020.

P. Cisneros-Velarde, A. Petersen, and S.-Y. Oh. Distributionally robust formulation and model selection for the graphical lasso. In *International Conference on Artificial Intelligence and Statistics*, pages 756–765. PMLR, 2020.

P. Consortium et al. A global multiproxy database for temperature reconstructions of the Common Era. *Scientific Data*, 4, 2017.

S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3):640–651, 2010.

S. Conti, J. P. Gosling, J. E. Oakley, and A. O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

K. Cowtan and R. G. Way. Coverage bias in the HadCRUT4 temperature series and its impact on recent temperature trends. *Quarterly Journal of the Royal Meteorological Society*, 140(683):1935–1944, 2014.

R. Cukier, C. Fortuin, K. E. Shuler, A. Petschek, and J. H. Schaibly. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I Theory. *The Journal of Chemical Physics*, 59(8):3873–3878, 1973.

D. Edwards. *Introduction to graphical modelling*. Springer Science & Business Media, 2012.

J. Fan, Y. Feng, and Y. Wu. Network exploration via the adaptive lasso and SCAD penalties. *The Annals of Applied Statistics*, 3(2):521, 2009.

J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

D. W. Gladish, D. E. Pagendam, L. J. Peeters, P. M. Kuhnert, and J. Vaze. Emulation engines: Choice and quantification of uncertainty for complex hydrological models. *Journal of Agricultural, Biological and Environmental Statistics*, 23(1):39–62, 2018.

S. E. Godsey, J. W. Kirchner, and C. L. Tague. Effects of changes in winter snowpacks on summer low flows: Case studies in the Sierra Nevada, California, USA. *Hydrological Processes*, 28(19):5048–5064, 2014.

J. Gomez-Navarro, J. Werner, S. Wagner, J. Luterbacher, and E. Zorita. Establishing the skill of climate field reconstruction techniques for precipitation with pseudoproxy experiments. *Climate Dynamics*, 45(5-6):1395–1413, 2015.

M. Gu, J. O. Berger, et al. Parallel partial Gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, 10(3):1317–1347, 2016.

D. Guillot, B. Rajaratnam, and J. Emile-Geay. Statistical paleoclimate reconstructions via Markov random fields. *The Annals of Applied Statistics*, 9(1):324–352, 2015.

E. J. Hanan, C. Tague, and J. P. Schimel. Nitrogen cycling and export in California chaparral: The role of climate in shaping ecosystem responses to fire. *Ecological Monographs*, 87(1):76–90, 2017.

T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. Ravikumar, and R. A. Poldrack. BIG & QUIC: Sparse inverse covariance estimation for a million variables. In *Advances in Neural Information Processing Systems*, volume 26, pages 3165–3173, 2013.

M. J. Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117(1-2):35–43, 1999.

P. Jones, D. Lister, T. Osborn, C. Harpham, M. Salmon, and C. Morice. Hemispheric and large-scale land-surface air temperature variations: An extensive revision and an update to 2010. *Journal of Geophysical Research: Atmospheres*, 117(D5), 2012.

J. J. Kennedy, N. Rayner, R. Smith, D. Parker, and M. Saunby. Reassessing biases and other uncertainties in sea surface temperature observations measured in situ since 1850: 1. Measurement and sampling uncertainties. *Journal of Geophysical Research: Atmospheres*, 116(D14), 2011.

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

M. C. Kennedy, D. McKenzie, C. Tague, and A. L. Dugger. Balancing uncertainty and complexity to incorporate fire spread in an eco-hydrological model. *International Journal of Wildland Fire*, 26(8):706–718, 2017.

K. Khare, S.-Y. Oh, and B. Rajaratnam. A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(4):803–825, 2015.

R. Killick and I. Eckley. changepoint: An R package for change point analysis. *Journal of Statistical Software*, 58(3):1–19, 2014.

R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of change points with a linear computational cost. *Journal of the American Statistical Association*, 107(500): 1590–1598, 2012.

F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger. Rainfall-runoff modelling using long short-term memory (LSTM) networks. *Hydrology and Earth System Sciences*, 22(11):6005–6022, 2018.

S. Kucherenko, D. Albrecht, and A. Saltelli. Exploring multi-dimensional spaces: A comparison of Latin hypercube and quasi Monte Carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015.

S. L. Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

F. Liu, M. West, et al. A dynamic modelling strategy for Bayesian computer model emulation. *Bayesian Analysis*, 4(2):393–411, 2009.

H. Liu, K. Roeder, and L. Wasserman. Stability approach to regularization selection (StARS) for high dimensional graphical models. *Advances in Neural Information Processing Systems*, 24(2):1432, 2010.

M. Mann, S. Rutherford, E. Wahl, and C. Ammann. Robustness of proxy-based climate field reconstruction methods. *Journal of Geophysical Research: Atmospheres*, 112 (D12), 2007.

R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *The Journal of Machine Learning Research*, 13(1):781–794, 2012.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

E. M. Meyers, B. Dobrowski, and C. L. Tague. Climate change impacts on flood frequency, intensity, and timing may affect trout species in Sagehen Creek, California. *Transactions of the American Fisheries Society*, 139(6):1657–1664, 2010.

H. Mohammadi, P. Challenor, and M. Goodfellow. Emulating dynamic non-linear simulators using Gaussian processes. *Computational Statistics & Data Analysis*, 139:178–196, 2019.

C. P. Morice, J. J. Kennedy, N. A. Rayner, and P. D. Jones. Quantifying uncertainties in global and regional temperature change using an ensemble of observational estimates: The HadCRUT4 data set. *Journal of Geophysical Research: Atmospheres*, 117(D8), 2012.

K. P. Murphy. *Machine learning: A probabilistic perspective*. MIT press, 2012.

R. Neukom, N. Steiger, J. J. Gómez-Navarro, J. Wang, and J. P. Werner. No evidence for globally coherent warm and cold periods over the preindustrial Common Era. *Nature*, 571(7766):550–554, 2019.

R. Olson, K. L. Ruckert, W. Chang, K. Keller, M. Haran, and S.-I. An. Stilt: Easy emulation of time series AR(1) computer model output in multidimensional parameter space. *R Journal*, 10(2), 2018.

N. E. Owen and L. Liuzzo. Impact of land use on water resources via a Gaussian process emulator with dimension reduction. *Journal of Hydroinformatics*, 21(3):411–426, 2019.

O. K. Oyebamiji, D. J. Wilkinson, B. Li, P. G. Jayathilake, P. Zuliani, and T. P. Curtis. Bayesian emulation and calibration of an individual-based model of microbial communities. *Journal of Computational Science*, 30:194–208, 2019.

A. O'Hagan. Bayesian analysis of computer code outputs: A tutorial. *Reliability Engineering & System Safety*, 91(10-11):1290–1300, 2006.

J. Peng, P. Wang, N. Zhou, and J. Zhu. Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association*, 104(486):735–746, 2009.

F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016.

A. Puy, S. L. Piano, A. Saltelli, and S. A. Levin. sensobol: An R package to compute variance-based sensitivity indices. *arXiv preprint arXiv:2101.10103*, 2021.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

M. M. Rajabi and H. Ketabchi. Uncertainty-based simulation-optimization using Gaussian process emulation: application to coastal groundwater management. *Journal of Hydrology*, 555:518–534, 2017.

C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

S. Razavi, A. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, E. Plischke, S. L. Piano, T. Iwanaga, W. Becker, et al. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954, 2021.

M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

A. J. Rothman, P. J. Bickel, E. Levina, and J. Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

J. Rougier. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, 17(4):827–843, 2008.

A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity analysis in practice: A guide to assessing scientific models*, volume 1. Wiley Online Library, 2004.

A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: The primer*. John Wiley & Sons, 2008.

A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010.

S. Scher. Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, 45(22): 12–616, 2018.

T. Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14(5):853–871, 2001.

A. J. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, pages 507–512, 1974.

J. Smerdon, A. Kaplan, E. Zorita, J. González-Rouco, and M. Evans. Spatial performance of four climate field reconstruction methods targeting the Common Era. *Geophysical Research Letters*, 38(11), 2011.

I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

I. M. Sobol. Sensitivity analysis for non-linear mathematical models. *Mathematical Modelling and Computational Experiment*, 1:407–414, 1993.

I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1-3):271–280, 2001.

M. Stein. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

C. Tague and L. Band. RHESSys: Regional Hydro-Ecologic Simulation System—An object-oriented approach to spatially distributed modeling of carbon, water, and nutrient cycling. *Earth Interactions*, 8(19):1–42, 2004.

C. Tague and G. E. Grant. A geological framework for interpreting the low-flow regimes of Cascade streams, Willamette River Basin, Oregon. *Water Resources Research*, 40 (4), 2004.

C. Tague and H. Peng. The sensitivity of forest water use to the timing of precipitation and snowmelt recharge in the California Sierra: Implications for a warming climate. *Journal of Geophysical Research: Biogeosciences*, 118(2):875–887, 2013.

C. Tague, C. McMichael, A. Hope, J. Choate, and R. Clark. Application of the RHESSys model to a California semiarid shrubland watershed. *Journal of the American Water Resources Association*, 40(3):575–589, 2004.

M. H. Taylor, M. Losch, M. Wenzel, and J. Schröter. On the sensitivity of field reconstruction and prediction using empirical orthogonal functions derived from gappy data. *Journal of Climate*, 26(22):9194–9205, 2013.

C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

A. Vaccaro, J. Emile-Geay, D. Guillot, R. Verna, C. Morice, J. Kennedy, and B. Rajaratnam. Climate field completion via Markov random fields: Application to the HadCRUT4.6 temperature dataset. *Journal of Climate*, 34(10):4169–4188, 2021.

J. Yang, A. Jakeman, G. Fang, and X. Chen. Uncertainty analysis of a semi-distributed hydrologic model based on a Gaussian process emulator. *Environmental Modelling & Software*, 101:289–300, 2018.

M. Yuan. Efficient computation of l_1 regularized estimates in Gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17(4):809–826, 2008.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

N. R. Zhang and D. O. Siegmund. A modified Bayes information criterion with applications to the analysis of comparative genomic hybridization data. *Biometrics*, 63(1): 22–32, 2007.

B. Zierl, H. Bugmann, and C. L. Tague. Water and carbon fluxes of European ecosystems: An evaluation of the ecohydrological model RHESSys. *Hydrological Processes: An International Journal*, 21(24):3328–3339, 2007.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.