UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Statistical Inference and Ensemble Machine Learning for Dependent Data

Permalink
https://escholarship.org/uc/item/6895805q

Author
Davies, Molly Margaret

Publication Date
2015

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Statistical Inference and Ensemble Machine Learning for Dependent Data

by

Molly Margaret Davies

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biostatistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Mark J. van der Laan, Chair
Professor Alan E. Hubbard
Professor Nina Maggi Kelly

Summer 2015

Statistical Inference and Ensemble Machine Learning for Dependent Data

Copyright 2015
by
Molly Margaret Davies

Abstract

Statistical Inference and Ensemble Machine Learning for Dependent Data

by

Molly Margaret Davies

Doctor of Philosophy in Biostatistics

University of California, Berkeley

Professor Mark J. van der Laan, Chair

The focus of this dissertation is on extending targeted learning to settings with complex unknown dependence structure, with an emphasis on applications in environmental science and environmental health.

The bulk of the work in targeted learning and semiparametric inference in general has been with respect to data generated by independent units. Truly independent, randomized experiments in the environmental sciences and environmental health are rare, and data indexed by time and/or space is quite common. These scientific disciplines need flexible algorithms for model selection and model combining that can accommodate things like physical process models and Bayesian hierarchical approaches. They also need inference that honestly and realistically handles limited knowledge about dependence in the data. The goal of the research program reflected in this dissertation is to formalize results and build tools to address these needs.

Chapter 1 provides a brief introduction to the context and spirit of the work contained in this dissertation.

Chapter 2 focuses on Super Learner for spatial prediction. Spatial prediction is an important problem in many scientific disciplines, and plays an especially important role in the environmental sciences. We review the optimality properties of Super Learner in general and discuss the assumptions required in order for them to hold when using Super Learner for spatial prediction. We present results of a simulation study confirming Super Learner works well in practice under a variety of sample sizes, sampling designs, and data-generating functions. We also apply Super Learner to a real world, benchmark dataset for spatial prediction methods. Appendix A contains a theorem extending an existing oracle inequality to the case of fixed design regression.

Chapter 3 describes a new approach to standard error estimation called Sieve Plateau (SP) variance estimation, an approach that allows us to learn from sequences of influence function based variance estimators, even when the true dependence structure is poorly understood. SP variance estimation can be prohibitively computationally expensive if not implemented with care. Appendix D contains completely general, highly optimized, heavily commented code as a reference for future users.

Chapter 4 uses targeted learning techniques to examine the relationship between ventilation rate and illness absence in a California school district observed over a period of two years. There is much that is unknown about the relationship between ventilation rates and human health outcomes, and there is a particular need to know more with respect to the school environment. It would be helpful for policy makers and indoor environment scientists to have estimates of average classroom illness absence rates when the average ventilation rate in the recent past failed to achieve a variety of different thresholds. The aim of this work is to provide these estimates. These data are challenging to work with, as they constitute a clustered, discontinuous time series with unknown dependence structure at both the classroom and school level. We use Super Learner to estimate the relevant parts of the likelihood; targeted maximum likelihood to estimate the target parameters; and SP variance estimation to obtain standard error estimates.

To my husband Casey, my parents Sharon and Steve, and to Ellie, for all her nudges under the elbow.

Contents

Contents **ii**

List of Figures **iv**

List of Tables **vi**

1 Introduction **1**

2 Optimal Spatial Prediction using Ensemble Machine Learning **4**
2.1 Introduction 4
2.2 Problem Formulation 5
2.3 The Super Learner Algorithm 6
2.4 Cross-validation and Spatial Data 8
2.5 Simulation Study 9
2.6 Practical Example: Predicting Lake Acidity 18
2.7 Discussion and Future Directions 19

3 Sieve Plateau Variance Estimators: a New Approach to Confidence Interval Estimation for Dependent Data **22**
3.1 Introduction 22
3.2 Target Parameter 24
3.3 Sieve Plateau Estimators 25
3.4 Supporting Theory 28
3.5 Simulation Study: Variance of the Sample Mean of a Time Series . . . 30
3.6 Practical Data Analysis: Average Treatment Effect for a Time Series . 34
3.7 Discussion and Future Directions 43

4 Small Increases in Classroom Ventilation Rates May Substantially Reduce Illness Absence: Evidence From a Prospective Study in California Elementary Schools **46**
4.1 Introduction 46
4.2 Observed Data and Target Parameter 48
4.3 Estimating the Target Parameter Using TMLE 53
4.4 Dependence in the Data 56

4.5 Estimating Standard Errors When Dependence is Poorly Understood . 58
4.6 Results 61
4.7 Discussion 62

Bibliography **66**

A Spatial Prediction - Oracle Inequality for Independent, Nonidentical Experiments and Quadratic Loss **73**

B Spatial Prediction - Tables **78**

C Sieve Plateau Variance Estimation - Approximating the Variance of Variance Estimators **81**

D SP Variance Estimation - Code for Computing the Variance of Variance estimator **83**

E SP Variance Estimation - Proof of Theorem 2 **95**

F Ventilation and Illness Absence - Table **97**

List of Figures

2.1 The six spatial processes used in the simulation study. All surfaces were simulated once on the domain $[0,1]^2$. Process values for all surfaces were scaled to $[-4,4] \subset \mathbb{R}$. 10
2.2 (a) A map of Super Learner's pH predictions, and (b) a plot of Super Learner's predictions as a function of the observed data. Super Learner mildly attenuated the pH values at either end of the range, but otherwise provided a fairly close fit to the data. 20

3.1 Boxplot of overall standardized bias for a subset of estimators. Boxplots are ordered vertically (top is best, bottom is worst) according to average normalized MSE. 35
3.2 Boxplot of standardized bias by sample size for a subset of estimators. Boxplots are ordered vertically (top is best, bottom is worst) according average normalized MSE. 36
3.3 Diagnostic plots of subsample size for SS estimator with b estimated data-adaptively. 37
3.4 Descriptive plots of (a) 7-day average VRs (L/s/p), (b) daily illness absence counts, and (c) a scatterplot of daily illness absence as a function of prior 7-day average VR. 43
3.5 Visualizations of SP variance estimation approaches when ordering by (a) L_1fit, (b) complexity, and (c) the number of nonzero D_{ij} pairs included in the estimator. (d) shows the estimated densities of each PAV curve. 44

4.1 Density plots of $\overline{V}(t)$ for various subcategories. 49
4.2 Bar plots of the proportion of daily classroom illness absence counts, $Y(t)$. 50
4.3 Mean seven day moving average VR, $\overline{V}(t)$, and daily illness absence count $Y(t)$ by various categories. 52
4.4 Barplot of daily classroom enrollment. 52
4.5 63

4.6 $\psi_n(v)$ and estimated confidence intervals for each VR threshold v. Estimated 0.95 confidence intervals assuming indepedent observations are substantively smaller than those based on SP variance estimation. All nine SP-based confidence intervals tended to be in close agreement with one another. In Figure (d), $\psi_n(v)$ and naive estimates for each VR threshold v, along with the largest (most conservative) estimated confidence interval. . 63
4.7 Densities of PAV curve values for each v and each sieve ordering. All three orderings tended to have very similar modes, although the complexity ordering exhibited more pronounced multimodality. 64

List of Tables

2.1 A list of R packages used to build the Super Learner library for spatial prediction. 11
2.2 Kernels implemented in the simulation library. $\langle \mathbf{x}, \mathbf{x}' \rangle$ is an inner product. 12
2.3 Average FVUs (standard deviations in parentheses) from the simulation study for each algorithm class. FVUs were calculated from predictions made on all unsampled points at each iteration. Algorithms are ordered according to overall performance. 16
2.4 Average FVU (standard deviation in parentheses) by spatial process. . . . 17

3.1 Simulation results. Normalized MSE with respect to the true variance, $\left(\sigma_n^2 - \sigma_0^2\right)^2 / \sigma_0^2$. Normalized bias with respect to the true variance is in parentheses. 33
3.2 Coverage probabilities 35
3.3 ATE variance estimation. Results ignoring dependence, and SP estimators, ordering by number of non-zero elements in the estimator; L_1 fit; and complexity. All SP estimators are of the 'kitchen sink' variety, utilizing $12,956$ unique dependence lag vectors. 42

4.1 Threshold values used to define $A_v(t)$, and the proportion of classroom days where $A_v(t) = 1$ for each of these values. Note that 7.1 L/s/p is the current standard for newly constructed schools in most building codes. 50

B.1 Simulation results for full library. For each algorithm, average Fraction of Variance Unexplained, (Avg FVU, standard deviation in parentheses) is the FVU averaged over all spatial processes, sample sizes, sampling designs, and noise condidtions. At each iteration, MSEs were calculated using all unsamped locations. Note that of the eight Kriging algorithms, only two were used to predict all spatial processes. 79
B.2 Lake acidity results for full library. **S** denotes the variable subset each algorithm was given. Risks were estimated via cross-validation (CV) or on the full dataset (Full). . . . 80

F.1 Estimated mean IA counts when $\overline{V}(t)$ failed to reach v L/s/p, and associated 0.95 CIs. 97

Acknowledgments

I owe an enormous debt of gratitude to my adviser Mark van der Laan - for his patience, open mindedness, incredible dedication to developing his students and empowering them to do good, ethical work; and for teaching me that having a bad memory can be an amazing asset if it means you will always look at things with fresh eyes.

I am also very grateful to Mark J. Mendell. My time as his research assistant at Lawrence Berkeley National Laboratory was foundational for me as a statistician and enormously inspiring. I could not have hoped for a better supervisor. The illness absence and ventilation rate data used throughout this dissertation were collected during my time in his lab. He has generously allowed me to use it for my own purposes.

I am especially grateful for my math teachers at Monterey Peninsula College, who were all, without exception, the best teachers I have ever had. I cannot thank them enough for what they did for me. I owe a special thanks to Don Philley in particular, a profoundly gifted, humane educator who knew exactly what to with a nervous, curious little math spark: add a dash of humor, some amazing physics examples, world class board work, and away we go!

There are a number of parts of this dissertation that simply wouldn't have happened without the help of Nathan Kurz, who has offered tremendous technical support and patient instruction in addition to his steadfast friendship. He helped me crawl out of the primordial ooze of thinking I knew how to program, skillfully guided me through the dangerous intermediate stages of thinking that now I *really* knew how to program, and has safely delivered me to a place of healthy respect for all that I don't know. Hardware matters!

I am also thankful to Alan Hubbard, for being so helpful throughout my time at Berkeley, and to Maggie Kelly, Mike Jerrett, the ESPM spatial seminar folks and all my biostatistics colleagues, for providing a wonderful sense of intellectual kinship.

Finally, none of this would have been possible without the faith and support of my parents Stephen and Sharon and my husband Casey. My gratitude to them for helping me achieve this dream knows no bounds. We are done!

Chapter 1

Introduction

The focus of this dissertation is on extending targeted learning to settings with complex unknown dependence structure, with an emphasis on applications in environmental science and environmental health.

Targeted learning is concerned with semiparametric estimators and corresponding statistical inference for potentially complex parameters of interest [Rose and van der Laan, 2011b]. It incorporates ensemble machine learning methods and differs from other approaches in advantageous ways. The methods are tailored to perform optimally for the target parameter of interest. This minimizes the need to fit unnecessary nuisance parameters and targets the bias-variance trade-off toward the goal of optimal estimation of the parameter of interest.

Targeted learning has two general purpose components. The first is an ensemble machine learning algorithm, Super Learner, which works by combing predictions from a diverse set of competing algorithms using cross-validation. This allows scientists to incorporate multiple competing hypotheses about how the data are generated, thus eliminating the need to choose a single algorithm a priori. Theory guarantees Super Learner will perform asymptotically at least as well as the best algorithm in the competing set. The second component is Targeted Maximum Likelihood (TML) estimation, a procedure for estimating parameters in semiparametric models. TML estimators are efficient, unbiased, loss-based defined substitution estimators that work by updating initial estimates in a bias-reduction step targeted toward the parameter of interest instead of the overall density. Targeted learning has been used in numerous contexts, including randomized controlled trials and observational studies, direct and indirect effect analyses, and case-control studies with complex censoring mechanisms.

The bulk of the work in targeted learning and semiparametric inference in general has been with respect to data generated by independent units. There have also been some recent, important extensions to TML estimation for dependent networks and other data structures not based on independent units, enabling TML estimation in disciplines where the most fundamental questions involve causal relationships between elements of highly interrelated systems [van der Laan, 2014]. However, these methods require one to know the underlying dependence structure of one's data. In the environmental sciences and environmental health, this is often not the case.

Furthermore, even when scientists do have such knowledge, it is very likely incomplete. In addition, these disciplines are in the midst of a grand data revolution, driven by technologies such as imaging spectroscopy and highly time-resolved sensor networks. As such, they are moving toward experiments that simply measure everything instead of randomly sampling from a target population.

Both the environmental sciences and environmental health have strong traditions of mathematical and parametric structural equation modeling. Thus many scientists in these disciplines are intuitively familiar with some of the core concepts of causal inference and targeted learning, such as counterfactuals and conditional independence. However, methodological development in these areas has traditionally focused on ways to describe complete systems. This has meant that much of the work is more descriptive in nature and does not necessarily generate actionable information. With the advent of remotely sensed imagery and large sensor networks, there has been an increased focus on prediction, occasionally using more flexible machine learning methods, but the scientific aim is most often fundamentally the same: to describe the state of the world as accurately and completely as possible.

There are questions of critical importance in these disciplines that cannot be addressed rigorously through descriptive approaches alone, however. These scientists need flexible algorithms for model selection and model combining that can accommodate things like physical process models and Bayesian hierarchical approaches. They also need inference that honestly and realistically handles limited knowledge about dependence in the data. The goal of the research program reflected in this dissertation is to formalize results and build tools to address these needs.

Chapter 2 focuses on Super Learner for spatial prediction. Spatial prediction is an important problem in many scientific disciplines, and plays an especially important role in the environmental sciences. Chapter 2 reviews the optimality properties of Super Learner in general and discusses the assumptions required in order for these properties to hold when using Super Learner for spatial prediction. It also presents results of a simulation study confirming Super Learner works well in practice under a variety of sample sizes, sampling designs, and data-generating functions. Chapter 2 demonstrates an application of Super Learner to a real world, benchmark dataset for spatial prediction methods. A theorem extending an existing oracle inequality to the case of fixed design regression is contained in the appendix.

Chapter 3 describes a new approach to standard error estimation called Sieve Plateau (SP) variance estimation. Suppose we have a data set of n observations where the extent of dependence between them is poorly understood. We assume we have an estimator that is $\sqrt{n}$-consistent for a particular estimand, and the dependence structure is weak enough so that the standardized estimator is asymptotically normally distributed. Our goal is to estimate the asymptotic variance of the standardized estimator so that we can construct a Wald-type confidence interval for the estimand. This chapter presents an approach that allows us to learn this asymptotic variance from a sequence of influence function-based candidate variance estimators. The focus is on time dependence, but the proposed method generalizes to data with arbitrary

dependence structure. Chapter 3 shows this approach is theoretically consistent under appropriate conditions. It also contains an evaluation of its practical performance with a simulation study, which shows the method compares favorably with various existing subsampling and bootstrap approaches. A real-world data analysis is also included, which estimates an average treatment effect (and a confidence interval) of ventilation rate on illness absence for a classroom observed over time. SP variance estimation can be prohibitively computationally expensive if not
implemented with care. Appendix D contains ompletely general, highly optimized code as a reference for future users. Under relatively modest sample sizes of $n \leq 2000$, this code will run on a standard laptop in a matter of seconds to minutes. The code is heavily commented, and provides some guidance as to how to modify and/or extend it to accommodate significantly larger sample sizes.

Chapter 4 uses targeted learning techniques to examine the relationship between ventilation rate and illness absence more fully. There is much that is unknown about the relationship between ventilation rates and human health outcomes, and there is a particular need to know more with respect to the school environment. It would be helpful to learn, for a small set of hypothetical thresholds, what average illness absence would be if classrooms failed to attain that particular threshold. The goal of this study is to provide policy makers with that information. To do this, we use data collected over a period of two years from 59 classrooms in a single California school district. These data constitute a clustered, discontinuous time series with unknown dependence structure at both the classroom and school level. Without SP variance estimation, it would be very difficult to obtain valid inference in this context.

Throughout this dissertation, a consistent effort is made to distinguish between true causal dependence and that which is merely similar by virtue of being close in space and/or time. This is not a distinction that is made in a large majority of work involving spatially and/or temporally indexed observations, where model-based inference is the norm. However, if semiparametric methodological development is to progress in these subject matter areas, we need to educate our scientific collaborators of the importance in our work of distinguishing between properties of the underlying data generating process and the models that have been traditionally used to represent that process. This distinction may seem overly technical to some, but it can be a good first step toward viewing one's data as a natural experiment, and help stimulate our collaborators to think more expansively and creatively about parameters they'd like to estimate. As statisticians, we have everything to gain from making this effort. The research questions in these disciplines are urgent; the potential estimation problems are beautifully complex and challenging; and there already exist rich inventories of data whose scientific potential has yet to be fully tapped.

Chapter 2

Optimal Spatial Prediction using Ensemble Machine Learning

2.1 Introduction

Optimal prediction of a spatially indexed variable is a crucial task in many scientific disciplines. For example, environmental health applications concerning air pollution often involve predicting the spatial distribution of pollutants of interest, and many agricultural studies rely heavily on interpolated maps of various soil properties.

Numerous algorithmic approaches to spatial prediction have been proposed (see Cressie [1993] and Schabenberger and Gotway [2005] for reviews), but selecting the best approach for a given data set remains a difficult statistical problem. One particularly challenging aspect of spatial prediction is that location is often used as a surrogate for large sets of unmeasured spatially indexed covariates. In such instances, effective prediction algorithms capable of capturing local variation must make strong, mostly untestable assumptions about the underlying spatial structure of the sampled surface and can be prone to overfitting. Ensemble predictors that combine the output of multiple predictors can be a useful approach in these contexts, allowing one to consider multiple aggressive predictors. There have been some recent examples of the use of ensemble approaches in the spatial and spatiotemporal literature. For example, Zaier et al. [2010] used ensembles of artificial neural networks to estimate the ice thickness of lakes and Chen and Wang [2009] used stacked generalization to combine support vector machines classifying land-cover types in hyperspectral imagery. Ensembling techniques have also been used to make spatially indexed risk maps. For example, Rossi et al. [2010] used logistic regression to combine a library of four base learners trained on a subset of the observed data to obtain landslide susceptibility forecasts for the central Umbrian region of Italy. Kleiber et al. [2011] have developed a Bayesian model averaging technique for obtaining locally calibrated probabilistic precipitation forecasts by combining output from multiple deterministic models.

The Super Learner prediction algorithm is an ensemble approach that combines a user-supplied library of heterogeneous candidate learners in such a way as to minimize ν-fold cross-validated risk [Polley and van der Laan, 2010]. It is a generalization of the stacking algorithm first introduced by Wolpert [1992] within the context of

neural networks and later adapted by Breiman [1996] to the context of variable subset regression. LeBlanc and Tibshirani [1996] discuss stacking and its relationship to the model-mix algorithm of Stone [1974] and the predictive sample-reuse method of Geisser [1975]. The library on which Super Learner trains can include parametric and nonparametric models as well as mathematical models and other ensemble learners. These learners are then combined in an optimal way in the sense that the Super Learner predictor will perform asymptotically as well as or better than any single prediction algorithm in the library under consideration. Super Learner has been used successfully in nonspatial prediction (see for example Polley et al. [2011a]). This chapter reviews its optimality properties and discusses the assumptions necessary for these optimality properties to hold within the context of spatial prediction. The results of a simulation study are also presented, demonstrating that Super Learner works well in practice under a variety of spatial sampling schemes and data-generating distributions. In addition, Super Learner is applied to a real world dataset, predicting water acidity for a set of 112 lakes in the Southeastern United States. Super Learner is shown to be a practical, data-driven, theoretically supported way to build an optimal spatial prediction algorithm from a large, heterogeneous set of predictors, protecting against both model misspecification and over-fitting. A novel oracle inequality within the context of fixed design regression is contained in Appendix A.

2.2 Problem Formulation

Consider a random spatial process indexed by location over a fixed, continuous, d-dimensional domain, $\left\{Y(s) : s \in D \subset \mathbb{R}^d\right\}$. For a particular set of distinct sampling points $\{S_1, ..., S_n\} \subset D$, We observe $\{(\mathbf{S}_i, Y_i^*) : i = 1, \ldots, n\}$, where $Y^* = Y(\mathbf{S}_i) + \epsilon_i$ and ϵ_i represents measurement error for the i^{th} observation. For all i, we assume $\mathbb{E}[Y_i^* | \mathbf{S}_i = \mathbf{s}] = Y(\mathbf{s})$. Our objective is to predict $Y(\mathbf{s}')$ for unobserved locations $\mathbf{s}' \subset D$. Thus, our parameter of interest is the spatial process itself. We do not make any assumptions about the functional form of the spatial process. We do, however, assume that one of the following is true: for all i, either

(1) $(\mathbf{S}_i, Y_i^*)$ are independently and identically distributed (i.i.d.), or

(2) $(\mathbf{S}_i, Y_i^*)$ are independent but not identically distributed, or

(3) Y_i^* are independent given $\mathbf{S}_1, \ldots, \mathbf{S}_n$; and $\mathbb{E}[Y_i^* | \mathbf{S}_1, \ldots, \mathbf{S}_n] = \mathbb{E}[Y_i^* | \mathbf{S}_i] = Y(\mathbf{S}_i)$. This corresponds to a fixed design.

Each of these sets of assumptions implies that any measurement error is mean zero conditional on $\mathbf{S}_i$, or in the case of fixed design, conditional on $\mathbf{S}_1, \ldots, \mathbf{S}_n$. It is important to note that $\mathbf{S}$ could consist of both location and some additional covariates $\mathbf{W}$, i.e. $\mathbf{S} = (\mathbf{X}, \mathbf{W})$, where $\mathbf{X}$ refers to location. In such cases, it may be that measurement error is mean zero conditional on location and covariates, but not on location alone.

While these are reasonable assumptions for many spatial prediction problems, they are nontrivial and may not always be appropriate. For instance, instrumentation and calibration error within sensor networks can result in spatially structured measurement error that is not mean zero given $\mathbf{S}_1, \ldots, \mathbf{S}_n$. There has been an effort on the part of researchers to develop ways to adapt the cross-validation procedure so as to minimize the effects of this kind of measurement error when choosing parameters such as bandwidth in local linear regression or smoothing parameters for splines. Interested readers should consult Opsomer et al. [2001] and Francisco-Fernandez and Opsomer [2005] for overviews.

2.3 The Super Learner Algorithm

Suppose we have observed n copies of the random variable O with true data-generating distribution $P_0 \in \mathcal{M}$, where the statistical model $\mathcal{M}$ contains all possible data generating distributions for O. The empirical distribution for our sample is denoted P_n. Define a parameter $\Psi : \mathcal{M} \to \mathbf{\Psi} \equiv \{\Psi[P] : P \in \mathcal{M}\}$ in terms of a risk function R as follows: $\Psi[P] = \text{argmin}_{\psi \in \mathbf{\Psi}} R(\psi, P)$. In this paper, we will limit our discussion to so-called linear risk functions, where $R(\psi, P) = PL(\psi) = \int L(\psi)(o) dP(o)$ for some loss function L. For a discussion of nonlinear risk functions, see van der Laan and Dudoit [2003]. We write our parameter of interest as $\psi_0 = \Psi[P_0] = \text{argmin}_{\psi} R(\psi, P_0)$, a function of the true data generating distribution P_0. For many spatial prediction applications, the Mean-Squared Error (MSE) is an appropriate choice for the risk function R, but this needn't necessarily be the case.

Define a library of J base learners of the parameter of interest ψ_0, denoted

$$\{\widehat{\Psi}_j : P_n \to \widehat{\Psi}_j[P_n]\}_{j=1}^J.$$

We make no restrictions on the functional form of the base learners. For example, within the context of spatial prediction, a library could consist of various Kriging and smoothing splines algorithms, Bayesian hierarchical models, mathematical models, machine learning algorithms, and other ensemble algorithms. We make a minimal assumption about the size of the library: it must be at most polynomial in sample size. Given this library of base learners, we consider a family of combining algorithms $\{\widehat{\Psi}_\alpha = f(\{\widehat{\Psi}_j : j\}, \alpha) : \alpha\}$ indexed by a Euclidean vector α for some function f. One possible choice of combining family is the family of linear combinations, $\widehat{\Psi}_\alpha = \sum_{j=1}^J \alpha(j) \widehat{\Psi}_j$. If it is known that $\psi_0 \in [0, 1]$, one might instead consider the logistic family, $\log[\widehat{\Psi}_\alpha/(1 - \widehat{\Psi}_\alpha)] = \sum_{j=1}^J \alpha(j) \log[\widehat{\Psi}_\alpha/(1 - \widehat{\Psi}_\alpha)]$. In either of these families, one can also constrain the values α can take. In this paper, we constrain ourselves to convex combinations, i.e. for all j, $\alpha(j) \geq 0$ and $\sum_j \alpha(j) = 1$.

Let $\{\mathbf{B}_n\}$ be a collection of length n binary vectors that define a random partition of the observed data into a training set $\{O_i : B_n(i) = 0\}$ and a validation set $\{O_i : B_n(i) = 1\}$. The empirical probability distributions for the training and validation sets are denoted $P^0_{n,\mathbf{B}_n}$ and $P^1_{n,\mathbf{B}_n}$, respectively. The estimated risk of a particular estimator

$\widehat{\Psi} : P_n \rightarrow \widehat{\Psi}[P_n]$ obtained via cross-validation is defined as

$$\begin{aligned}\mathbb{E}_{\mathbf{B}_n}\left[R\left(\widehat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right], P^1_{n,\mathbf{B}_n}\right)\right] &= \mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n} L\left(\widehat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right]\right)\right] \\ &= \mathbb{E}_{\mathbf{B}_n}\left[\int L\left(\widehat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right], y\right) dP^1_{n,\mathbf{B}_n}(y)\right].\end{aligned}$$

Given a particular class of candidate estimators indexed by α, the cross-validation selector selects the candidate which minimizes the cross-validated risk under the empirical distribution P_n,

$$\alpha_n \equiv \underset{\alpha}{\operatorname{argmin}} \left\{\mathbb{E}_{\mathbf{B}_n}\left[R\left(\widehat{\Psi}_\alpha\left[P^0_{n,\mathbf{B}_n}\right], P^1_{n,\mathbf{B}_n}\right)\right]\right\}.$$

The Super Learner estimate of ψ_0 is denoted $\widehat{\Psi}_{\alpha_n}[P_n]$.

2.3.1 Key Theoretical Results

Super Learner's aggressive use of cross-validation is informed by a series of theoretical results originally presented in van der Laan and Dudoit [2003] and expanded upon in van der Vaart et al. [2006]. We provide a summary of these results below. For details and proofs, the reader is referred to these papers.

First, we define a benchmark procedure called the oracle selector, which selects the candidate estimator that minimizes the cross-validated risk under the true data generating distribution P_0. We denote the oracle selector for estimators based on cross-validation training sets of size $n(1-p)$, where p is the proportion of observations in the validation set, as

$$\widetilde{\alpha}_n \equiv \underset{\alpha}{\operatorname{argmin}} \left\{\mathbb{E}_{\mathbf{B}_n}\left[R\left(\widehat{\Psi}_\alpha\left[P^0_{n,\mathbf{B}_n}\right], P_0\right)\right]\right\}.$$

van der Laan and Dudoit [2003] present an oracle inequality for the cross-validation selector α_n in the case of random design regression. Let $L(\cdot)$ be a uniformly bounded loss function with

$$M_1 \equiv \sup_{\psi, O} |L(\psi)[O] - L(\psi_0)[O]| < \infty.$$

Let $d_n(\psi, \psi_0) = P_0\left[L(\psi) - L(\psi_0)\right]$ be a loss-function based risk dissimilarity between an arbitrary predictor ψ and the parameter of interest ψ_0, where the risk dissimilarity $d_n(\cdot)$ is quadratic in the difference between ψ and ψ_0, i.e.

$$P_0[L(\psi) - L(\psi_0)]^2 \leq M_2 P_0[L(\psi) - L(\psi_0)].$$

Suppose the cross-validation selector α_n defined above is a minimizer over a grid of K_n different α-indexed candidate estimators. Then for any real-valued $\delta > 0$,

$$\begin{aligned}\mathbb{E}\left[d_n\left(\widehat{\Psi}_{\alpha_n}\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right] \leq (1+2\delta)\, \mathbb{E}\left[\min_\alpha \mathbb{E}_{\mathbf{B}_n} d_n\left(\widehat{\Psi}_\alpha[P^0_{n,\mathbf{B}_n}], \psi_0\right)\right] \\ + C(M_1, M_2, \delta)\frac{\log K_n}{n},\end{aligned} \tag{2.1}$$

where $C(\cdot)$ is a constant defined in van der Vaart et al. [2006] (see also Appendix A for a definition within the context of fixed regression). Thus if the proportion of observations in the validation set, p, goes to zero as $n \to \infty$, and

$$\frac{1}{n \log n} \mathbb{E}\left[\min_{\alpha} \mathbb{E}_{\mathbf{B}_n} d_n\left(\widehat{\Psi}_\alpha\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right] \xrightarrow{n\to\infty} 0,$$

it follows that $\widehat{\Psi}_{\alpha_n}$, the estimator selected by the cross-validation selector, is asymptotically equivalent to the estimator selected by the oracle, $\widehat{\Psi}_{\widetilde{\alpha}_n}$, when applied to training samples of size $n(1-p)$, in the sense that

$$\frac{\mathbb{E}_{\mathbf{B}_n}\left[d\left(\widehat{\Psi}_{\alpha_n}\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right]}{\mathbb{E}_{\mathbf{B}_n}\left[d\left(\widehat{\Psi}_{\widetilde{\alpha}_n}\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right]} \xrightarrow{n\to\infty} 1.$$

The oracle inequality as presented in equation (2.1) shows us that if none of the base learners in the library are a correctly specified parametric model and therefore do not converge at a parametric rate, the cross-validation selector performs as well in terms of expected risk dissimilarity from the truth as the oracle selector, up to a typically second order term bounded by $(\log K_n)/n$. If one of the base learners *is* a correctly specified parametric model and thus achieves a parametric rate of convergence, the cross-validation selector converges (with respect to expected risk dissimilarity) at an almost parametric rate of $(\log K_n)/n$.

For the special case where $Y^* = Y$ and the dimension of $\mathbf{S}$ is two, the cross-validation selector performs asymptotically as well as the oracle selector up until a constant factor of $(\log K_n)/n$. When $Y^* = Y$ and the dimension of $\mathbf{S}$, d, is greater than two, the rates of convergence of the base learners will be $n^{-1/d}$. This is slower than $n^{-1/2}$, the rate for a correctly specified parametric model, so the asymptotic equivalence of the cross-validation selector with the oracle selector applies.

The original work of van der Laan and Dudoit [2003] used a random regression formulation. Spatial prediction problems where we have assumed either (2) or (3) in section 2.2 above require a fixed design regression formulation. A proof of the oracle inequality for the fixed design regression case is contained in Appendix A.

The key message is that Super Learner is a data-driven, theoretically supported way to build the best possible prediction algorithm from a large, heterogeneous set of predictors. It will perform asymptotically as well as or better than the best candidate prediction algorithm under consideration. Expanding the search space to include all convex combinations of the candidates can be an important advantage in spatial prediction problems, where location is often used as a surrogate for unmeasured spatially indexed covariates. Super Learner allows one to consider sufficiently complex, flexible functions while providing protection against overfitting.

2.4 Cross-validation and Spatial Data

The theoretical results outlined above depend on the training and validation sets being independent. When this is not the case, there are generally no developed theoretical

guarantees of the asymptotic performance of any cross-validation procedure [Arlot and Celisse, 2010]. Bernstein's inequality, which van der Laan and Dudoit [2003] use in developing their proof of the oracle inequality, has been extended to accommodate certain weak dependence structures, so it may be that there are ways to justify certain optimality properties of ν-fold cross-validation in these cases. There have also been some extensions to potentially useful fundamental theorems that accommodate other specific dependence structures. Lumley [2005] proved an empirical process limit theorem for sparsely correlated data which can be extended to the multidimensional case. Jiang [2009] provided probability bounds for uniform deviations in data with certain kinds of exponentially decaying one-dimensional dependence, although it is unclear how to extend these results to multidimensional dependency structures where sampling may be irregular. Neither of these extensions is immediately applicable to the general spatial case, where sampling may or may not be regular and the extent of spatial correlation cannot necessarily be assumed to be sparse. There has been some attention in the spatial literature to the use of cross-validation within the context of Kriging and selecting the best estimates for the parameters in a covariance function, most of it urging cautious and exploratory use [Cressie, 1993, Davis, 1987]. Todini [2001] has investigated methods to provide accurate estimates of model-based Kriging error when the covariance structure has been selected via leave-one-out cross-validation, although this remains an open problem.

Recall from section 2.2 above that our parameter of interest is the spatial process $Y(\mathbf{s})$ and we have assumed $\mathbb{E}[Y^*|\mathbf{S} = \mathbf{s}] = Y(\mathbf{s})$. Even if $Y(\mathbf{s})$ is a spatially dependent stochastic process such as a Gaussian random field, the true parameter of interest in most cases is not the full stochastic process, but rather the particular realization from which we have sampled. Conditioning on this realization removes all randomness associated with the stochastic process, and any remaining randomness comes from the sampling design and measurement error. So long as the data conform to one of the statistical models outlined above in section 2.2, the optimality properties outlined above will apply.

2.5 Simulation Study

The Super Learner prediction algorithm was applied to six data sets with known data generating distributions simulated on a grid of $128 \times 128 = 16,384$ points in $[0, 1]^2 \subset \mathbb{R}^2$. Each spatial process was simulated once, hence samples of stochastic processes were taken from a common realization. All simulated processes were scaled to $[-4, 4]$ before sampling.

The function $f_1(\cdot)$ is a mean zero stationary Gaussian random field (GRF) with Matérn covariance function [Matérn, 1986]

$$C(h, \theta) = \sigma^2 \left[\frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{h}{\phi} \right)^{\nu} K_\nu \left(\frac{h}{\phi} \right) \right] + \tau^2,$$
$$\theta = \left(\sigma^2 = 5,\ \phi = 0.5,\ \nu = 0.5,\ \tau^2 = 0 \right),$$

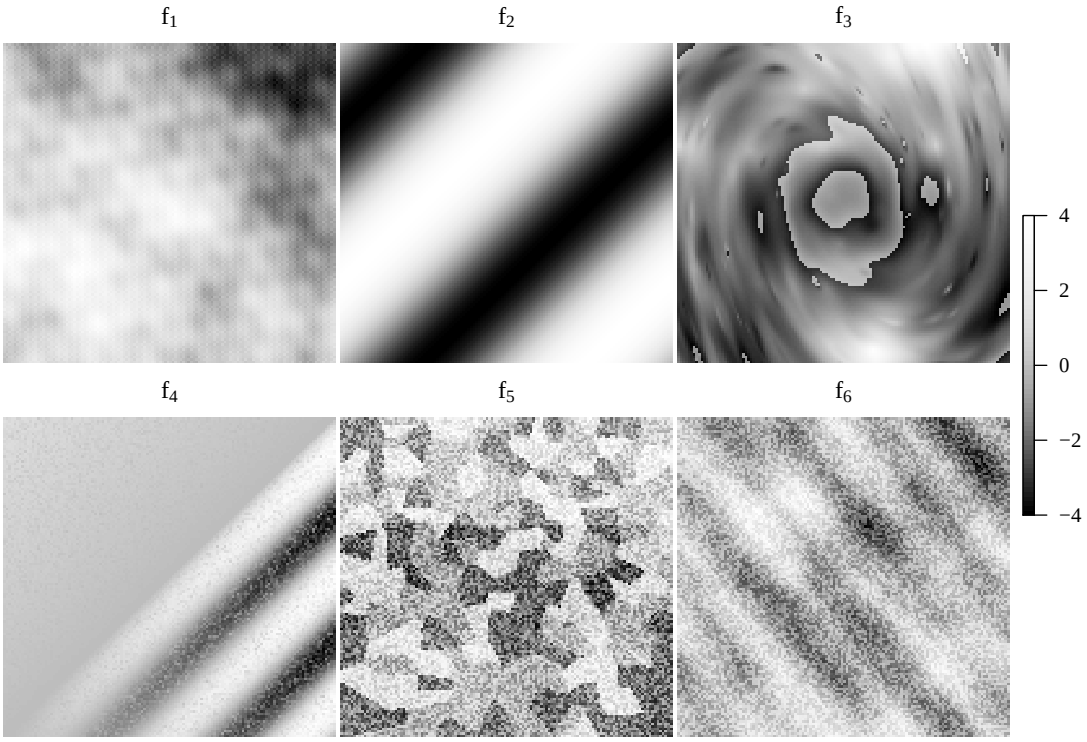


Figure 2.1: The six spatial processes used in the simulation study. All surfaces were simulated once on the domain $[0,1]^2$. Process values for all surfaces were scaled to $[-4,4] \subset \mathbb{R}$.

where h is a distance magnitude between two spatial locations, σ^2 is a scaling parameter, $\phi > 0$ is a range parameter influencing the spatial extent of the covariance function and τ^2 is a parameter capturing micro-scale variation and/or measurement error. $K_\nu(\cdot)$ is a modified Bessel function of the third order and $\nu > 0$ parametrizes the smoothness of the spatial covariation. Learners were given spatial location as covariates.

The function $f_2(\cdot)$ is a smooth sinusoidal surface used as a test function in both Huang and Chen [2007] and Gu [2002], $f_2(\mathbf{s}) = 1 + 3\sin(2\pi[s_1 - s_2] - \pi)$. Learners were given spatial location as covariates.

The function $f_3(\cdot)$ is a weighted nonlinear function of a spatiotemporal cyclone GRF and an exponential decay function of distances to a set of randomly chosen points in $[-0.5, 1.5]^2 \subset \mathbb{R}^2$. In addition to spatial location, learners were given the distance to the nearest point as a covariate.

The function $f_4(\cdot)$ is defined by the piecewise function

$$f_4(\mathbf{s}, w) = \big\{|s_1 - s_2| + w\big\} I(s_1 < s_2) + \big\{3s_1 \sin\big(5\pi[s_1 - s_2]\big) + w\big\} I(s_1 \geq s_2),$$

where w is Beta distributed with non-centrality parameter 3 and shape parameters 4 and 1.5. Learners were given spatial location and w as covariates.

Algorithm class	R library	Reference(s)
DSA	DSA	Neugebauer and Bullard [2010]
GAM	GAM	Hastie [2011]
GP	kernlab	Karatzoglou, Smola, Hornik, and Zeileis [2004]
GBM	GBM	Ridgeway [2010]
GLMnet	glmnet	Friedman, Hastie, and Tibshirani [2010]
KNNreg	FNN	Li [2012]
Kriging	geoR	Diggle and Jr. [2007], Ribeiro and Diggle [2001]
Polymars	polspline	Kooperberg [2010]
Random Forest	randomForest	Liaw and Wiener [2002]
SVM	kernlab	Karatzoglou, Smola, Hornik, and Zeileis [2004]
TPS	fields	Furrer, Nychka, and Sain [2011]

Table 2.1: A list of R packages used to build the Super Learner library for spatial prediction.

The function $f_5(\cdot)$ is a sum of several surfaces on $[0,1] \subset \mathbb{R}^2$; a nonlinear function of a random partition of $[0,1]^2$; a piecewise smooth function; and $w_2 \sim uniform(-1,1)$. Learners were given spatial location, partition membership (w_1) and w_2 as covariates.

The function $f_6(\cdot)$ is a weighted sum of a spatiotemporal GRF with five time-points, a distance decay function of a random set of points in $[0,1]^2$, and a beta-distributed random variable with non-centrality parameter 0 and shape parameters both equal to 0.5. Learners were given spatial location, the five GRFs and the beta-distributed random variable as covariates.

2.5.1 Spatial Prediction Library

The library provided to Super Learner consisted of either 83 (number of covariates = 2) or 85 (number of covariates > 2) base learners from 13 general classes of prediction algorithms. A brief description of each, and list the parameter values used in the libraries is provided below. All algorithms were implemented in R [R Development Core Team, 2012]. The names of the R packages used are listed in table 2.1.

Deletion/Substitution/Addition (**DSA**) performs data-adaptive polynomial regression using ν-fold cross-validation and the L_2 loss [Sinisi and van der Laan, 2004]. Both the number of folds in the algorithm's internal cross-validation and the maximum number of terms allowed in the model (excluding the intercept) were fixed to five. The maximum order of interactions was $k \in \{3,4\}$, and the maximum sum of powers of any single term in the model was $p \in \{5,10\}$.

Generalized Additive Models (**GAM**) assume the data are generated by a model of the form $\mathbb{E}[Y|X_1,\ldots,X_p] = \alpha + \sum_{i=1}^{p} \phi_i(X_i)$, where Y is the outcome, $(X_1,\ldots,X_p)$ are covariates and each $\phi_i(\cdot)$ is a smooth nonparametric function [Hastie, 1991]. In this simulation study, the $\phi(\cdot)$ are cubic smoothing spline functions parametrized by desired equivalent number of degrees of freedom, $df \in \{2,3,4,5,6\}$. To achieve a uniformly bounded loss function, predicted values were truncated to the range of the sampled data, plus or minus one.

Kernel	Function $k(\mathbf{x}, \mathbf{x}')$	Parameter values
Bessel	$\frac{J_{\nu+1}(\sigma\|\|\mathbf{x}-\mathbf{x}'\|\|)}{(\|\|\mathbf{x}-\mathbf{x}'\|\|)^{-d(\nu+1)}}$	$J_{\nu+1}$ is a Bessel function of 1^{st} kind, $(\sigma, \nu, d) \in \{1\} \times \{0.5, 1, 2\} \times \{2\}$
Radial Basis Function (RBF)	$\exp(-\sigma\|\mathbf{x}-\mathbf{x}'\|^2)$	Inverse kernel width σ estimated from data.
linear	$\langle \mathbf{x}, \mathbf{x}' \rangle$	None
polynomial	$(\alpha\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d$	$(\sigma, \alpha, d) \in \{1, 3\} \times \{0.001, 0.1, 1\} \times \{1\}$
hyberbolic tangent	$\tanh(\alpha\langle \mathbf{x}, \mathbf{x}' \rangle + c)$	$(\alpha, c) \in \{0.005, 0.002, 0.01\} \times \{0.25, 1\}$

Table 2.2: Kernels implemented in the simulation library. $\langle \mathbf{x}, \mathbf{x}' \rangle$ is an inner product.

Gaussian Processes (**GP**) assume the observed data are normally distributed with a covariance structure that can be represented as a kernel matrix [Williams, 1999]. Various implementations of the Bessel, Gaussian radial basis, linear and polynomial kernels were used. See table 2.2 for details about the kernel functions and parameter values. Predicted values were truncated to the range of the observed data, plus or minus one, to achieve a uniformly bounded loss function.

Generalized Boosted Modeling (**GBM**) combines regression trees, which model the relationship between an outcome and predictors by recursive binary splits, and boosting, an adaptive method for combining many weak predictors into a single prediction ensemble [Friedman, 2001]. The GBM predictor can be thought of as an additive regression model fitted in a forward stage-wise fashion, where each term in the model is a simple tree. We used the following parameter values: number of trees = 10,000; shrinkage parameter $\lambda = 0.001$; bag fraction (subsampling rate) = 0.5; minimum number of observations in the terminal nodes of each tree = 10; interaction depth $d \in \{1, 2, 3, 4, 5, 6\}$, where an interaction depth of d implies a model with up to d-way interactions.

GLMnet is a GLM fitted via penalized maximum likelihood with elastic-net mixing parameter $\alpha \in \{1/4, 1/2, 3/4\}$ [Friedman et al., 2010].

K-Nearest Neighbor Regression (**KNNreg**) assumes the unobserved spatial process at a prediction point $\mathbf{s}'$ can be well-approximated by an average of the observed spatial process values at the k nearest sampled locations to $\mathbf{s}'$, $k \in \{1, 5, 10, 20\}$. When $k = 1$ and $\mathbf{S}$ are spatial locations only, this is essentially equivalent to Thiessen Polygons.

Kriging is perhaps the most commonly used spatial prediction approach. A general formulation of the spatial model assumed by Kriging can be written as $Y(s) = \mu(s) + \delta(s),\ \delta(s) \sim N(0, C(\theta))$. The first term represents the large-scale mean trend, assumed to be deterministic and continuous. The second term is a Gaussian random function with mean zero and positive semi-definite covariance function $C(\theta)$ satisfying a stationarity assumption. The Kriging predictor is given as a linear combination of the observed data, $\widehat{\Psi}(\mathbf{s}') = \sum_{i=1}^{n} w_i(\mathbf{s}') Y^*(\mathbf{s}_i)$. The weights $\{w_i\}_{i=1}^n$ are chosen so that $\mathrm{Var}\left[\widehat{\Psi}(\mathbf{s}') - Y(\mathbf{s}')\right]$ is minimized, subject to the constraint that the predictions

are unbiased. Thus, given a parametric covariance function with known parameters θ and a known mean structure, a Kriging predictor computes the best linear unbiased predictor of $Y(\mathbf{s}')$. For the Kriging base learners, the parametric covariance function was assumed to be spherical,

$$C(h,\theta) = \tau^2 + \sigma^2 \left[1 - \frac{2}{\pi}\left(\sin^{-1}\left(\frac{h}{\phi}\right) + \frac{h}{\phi}\sqrt{1-\left(\frac{h}{\phi}\right)^2}\right)\right]\mathbf{I}\left(h < \phi\right).$$

The nugget τ^2, scale σ^2, and range ϕ were estimated using Restricted Maximum Likelihood (for details about REML, see for example Gelfand et al. [2010], chapter 4, pp 48-49). The trend was assumed to be one of the following: Constant (traditional Ordinary Kriging, OK); a first order polynomial of the locations (traditional Universal Kriging, UK); a weighted linear combination of non-location covariates only (if any); a weighted linear combination of both locations and non-location covariates (if any). All libraries contained the first and second Kriging algorithms. Libraries for simulated processes with additional covariates contained the third and fourth algorithms as well.

Multivariate adaptive polynomial spline regression (**Polymars**) is an adaptive regression procedure using piecewise linear splines to model the spatial process, and is parametrized by the maximum size $m = \min\{6n^{1/3}, n/4, 100\}$, where n is sample size [Stone et al., 1997].

The **Random Forest** algorithm proposed by Breiman [2001] is an ensemble approach that averages together the predictions of many regression trees constructed by drawing B bootstrap samples and for each sample, growing an unpruned regression tree where at each node, the best split among a subset of q randomly selected covariates is chosen. In our implementation, B was set to 1000, the minimum size of the terminal nodes was 5, and the number of randomly sampled variables at each split was $\lfloor\sqrt{p}\rfloor$, where p was the number of covariates.

The library contained a number of Support Vector Machines (**SVM**), each implementing one of two types of regression (epsilon regression, $\epsilon = 0.1$; or nu regression, $\nu = 0.2$), and one of five kernels: Bessel, Gaussian radial basis, linear, polynomial, and hyperbolic tangent. The kernels are described in table 2.2. Predicted values were truncated to plus or minus one the range of the observed data to ensure a bounded loss, and the cost of constraints violation was fixed at 1.

Thin-plate splines (**TPS**) is another common approach to spatial prediction. The observed data are presumed to be generated by a deterministic process $Y(\mathbf{s}) = g(\mathbf{s})$, where $g(\cdot)$ is an m times differentiable deterministic function with $m > d/2$ and $\dim(\mathbf{s}) = d$. The estimator of $g(\cdot)$ is the minimizer of a penalized sum of squares,

$$\hat{g} = \underset{g\in\mathcal{G}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(Y_i - g\left(\mathbf{s}_i\right)\right)^2 + \lambda J_m(g), \tag{2.2}$$

with d-dimensional roughness penalty

$$J_m(g) = \int_{\mathbb{R}^d} \sum_{\{(v_1,\ldots,v_d)\}} \binom{m}{v_1,\ldots,v_d} \left(\frac{\partial^m g(\mathbf{s})}{\partial \mathbf{s}_1^{v_1} \ldots \partial \mathbf{s}_d^{v_d}} \right)^2 d\mathbf{s},$$

where the sum in (2.5.1) is taken over all nonnegative integers $(v_1, \ldots, v_d)$ such that $\sum_{i=1}^d v_i = m$ [Green and Silverman, 1994]. The tuning parameter $\lambda \in [0, \infty)$ in (2.2) controls the permitted degree of roughness for $\widehat{g}$. As λ tends to zero, the predicted surface approaches one that exactly interpolates the observed data. Larger values of λ allow the roughness penalty term to dominate, and as λ approaches infinity, $\widehat{g}$ tends toward a multivariate least squares estimator. In our library, the smoothing parameter was either fixed to $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.1\}$ or estimated data-adaptively using Generalized Cross-validation (GCV) (see Craven and Wahba [1979] for a description of the GCV procedure). Predicted values were truncated to plus or minus one of the range of the observed data to ensure a bounded loss.

The library also contained a main terms Generalized Linear Model (**GLM**) and a simple empirical mean function.

2.5.2 Simulation Procedure

The simulation study examined the effect of sample size ($n \in \{64, 100, 529\}$), signal-to-noise ratio (SNR), and sampling scheme. SNR was defined as the ratio of the sample variance of the spatial process and the variance of additive zero-mean normally distributed noise representing measurement error. Processes were simulated with either no added noise or with noise added to achieve a SNR of 4. Three sampling schemes were examined: simple random sampling (SRS), random regular sampling (RRS), and stratified sampling (SS). Random regular samples were regularly spaced subsets of the $16,384$ point grid with the initial point selected at random. Stratified random samples were taken by first dividing the domain $[0, 1]^2$ into n equal-area bins and then randomly selecting a single point from each bin.

The following procedure was repeated 100 times for each combination of spatial process, sample size, SNR level, and sampling design, giving a total of 10,800 simulations:

1. Sample n locations and any associated covariates and process values from the grid of $16,384$ points in $[0, 1]^2 \subset \mathbb{R}^2$ according to one of the three sampling designs described above.

2. For those simulations with SNR $= 4$, draw n i.i.d. samples of the random variable $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ and add them to the n sampled process values $\{Y_1, \ldots, Y_n\}$, where σ_ε^2 has been calculated to achieve an SNR of 4.

3. Pass the sampled values to Super Learner, along with a library of base learners on which to train. The number of folds ν used in the cross-validation procedure depended on n: if $n = 64$, then $\nu = 64$; if $n = 100$, then $\nu = 20$; if $n = 529$, then

$\nu = 10$. Super learner uses cross-validation and the L_2 loss function to estimate the risk of each candidate predictor and returns an estimate of the optimal convex combination of the predictions made by all base learners according to their cross-validated risk.

4. For each base learner in the library and for the trained Super Learner, predict the spatial process under consideration at all unsampled points. Calculate mean squared errors (MSEs) and then divide these by the variance of the spatial process. This measure of performance is referred to as the Fraction of Variance Unexplained (FVU). This makes it reasonable to compare prediction performances across different spatial processes.

2.5.3 Simulation Results

Table B.1 in Appendix B lists the average performance for each individual base learner in the library, and Table 2.3 summarizes prediction performance for each algorithm class in the library and for Super Learner itself. Super learner was clearly the best predictor overall when comparing across broad classes, with an average FVU of 0.24 (SD = 0.22). The next best performing algorithmic class was thin-plate splines using GCV to choose the roughness penalty, with an average FVU of 0.42 (SD = 0.36). Universal Kriging (FVU = 0.44), random forest (FVU = 0.35), and Ordinary Kriging (FVU = 0.45) all performed similarly, which was slightly less well than TPS (GCV). Super Learner was also the best performer across noise conditions, sampling designs, and sample sizes, with performance improving markedly as sample size increased.

Table 2.4 breaks algorithmic class performance down by simulated surface. f_1 was a mean-zero GRF, something we would expect both Kriging and thin-plate splines algorithms to predict well. TPS (GCV) and Super Learner were the best performers, with nearly identical average FVUs of 0.11 (SD = 0.06). The other TPS algorithms and Universal Kriging faired slightly less well, with an average FVU of 0.15. Ordinary Kriging had an average FVU of 0.26, which was actually greater than the average FVUs for Random Forest (0.16), K-nearest neighbors regression (0.19), GBM (0.22), GAM (0.24), and DSA (0.25).

The function f_2 was a simple sinusoidal surface, another functional form where we would expect thin-plate splines to excel, provided the samples properly captured the periodicity of the process. TPS (GCV) had the best overall performance, with an average FVU of 0.07 (SD = 0.09). Super Learner performed only slightly less well, with an average FVU of 0.09 (SD = 0.11). The other TPS algorithms (0.23), Ordinary Kriging (0.24) and Universal Kriging (0.25) performed substantially less well on average.

The function f_3 was a relatively complex function involving a "cyclone" Gaussian random field and a distance decay function of randomly selected points. Once again, the average performances of TPS (GCV) and Super Learner were nearly identical (FVU = 0.30, SD = 0.11).

		Sample Size			SNR		Sampling Design		
Algorithm Class	Overall	64	100	529	None	4	SRS	RRS	SS
Super Learner	0.24 (0.22)	0.40 (0.26)	0.25 (0.15)	0.07 (0.06)	0.22 (0.22)	0.27 (0.22)	0.26 (0.23)	0.25 (0.24)	0.22 (0.18)
TPS (GCV)	0.42 (0.36)	0.58 (0.39)	0.44 (0.35)	0.24 (0.25)	0.40 (0.37)	0.45 (0.35)	0.46 (0.38)	0.41 (0.37)	0.40 (0.34)
Krige (UK)	0.44 (0.30)	0.59 (0.28)	0.51 (0.27)	0.21 (0.20)	0.42 (0.31)	0.46 (0.29)	0.42 (0.30)	0.53 (0.31)	0.36 (0.28)
Random Forest	0.45 (0.26)	0.56 (0.24)	0.49 (0.25)	0.29 (0.21)	0.44 (0.26)	0.46 (0.26)	0.48 (0.27)	0.42 (0.24)	0.45 (0.26)
Krige (OK)	0.45 (0.32)	0.62 (0.29)	0.53 (0.28)	0.21 (0.20)	0.43 (0.32)	0.47 (0.31)	0.41 (0.29)	0.59 (0.33)	0.36 (0.28)
KNNreg	0.50 (0.34)	0.67 (0.34)	0.56 (0.33)	0.27 (0.21)	0.47 (0.35)	0.53 (0.33)	0.53 (0.35)	0.48 (0.33)	0.49 (0.34)
TPS	0.53 (0.37)	0.64 (0.40)	0.56 (0.37)	0.37 (0.30)	0.49 (0.38)	0.56 (0.37)	0.58 (0.40)	0.49 (0.35)	0.52 (0.37)
GBM	0.54 (0.30)	0.69 (0.26)	0.57 (0.25)	0.36 (0.29)	0.53 (0.30)	0.55 (0.30)	0.55 (0.30)	0.54 (0.30)	0.54 (0.30)
DSA	0.61 (0.28)	0.68 (0.31)	0.62 (0.26)	0.54 (0.23)	0.60 (0.26)	0.63 (0.29)	0.64 (0.31)	0.60 (0.26)	0.60 (0.26)
GAM	0.65 (0.30)	0.70 (0.31)	0.65 (0.30)	0.60 (0.29)	0.64 (0.30)	0.66 (0.31)	0.68 (0.32)	0.63 (0.29)	0.64 (0.30)
GLMnet	0.69 (0.25)	0.71 (0.24)	0.69 (0.25)	0.67 (0.24)	0.69 (0.25)	0.69 (0.25)	0.70 (0.25)	0.69 (0.24)	0.69 (0.24)
GLM	0.69 (0.25)	0.71 (0.25)	0.69 (0.25)	0.67 (0.24)	0.69 (0.25)	0.69 (0.25)	0.70 (0.25)	0.68 (0.24)	0.69 (0.24)
Polymars	0.73 (0.36)	0.84 (0.40)	0.78 (0.33)	0.56 (0.29)	0.71 (0.34)	0.74 (0.38)	0.76 (0.40)	0.70 (0.34)	0.71 (0.34)
SVM	0.76 (0.30)	0.83 (0.30)	0.80 (0.31)	0.66 (0.27)	0.76 (0.30)	0.77 (0.30)	0.78 (0.31)	0.76 (0.30)	0.76 (0.30)
GP	0.77 (0.67)	0.89 (0.68)	0.80 (0.60)	0.61 (0.69)	0.74 (0.62)	0.80 (0.71)	0.80 (0.67)	0.76 (0.68)	0.76 (0.66)
Mean	1.01 (0.01)	1.01 (0.02)	1.01 (0.01)	1.00 (0.00)	1.01 (0.01)	1.01 (0.01)	1.01 (0.02)	1.00 (0.01)	1.00 (0.01)

Table 2.3: Average FVUs (standard deviations in parentheses) from the simulation study for each algorithm class. FVUs were calculated from predictions made on all unsampled points at each iteration. Algorithms are ordered according to overall performance.

	Average FVU					
Algorithm Class	f_1	f_2	f_3	f_4	f_5	f_6
Super Learner	0.11 (0.06)	0.09 (0.11)	0.30 (0.11)	0.43 (0.36)	0.22 (0.14)	0.31 (0.19)
TPS (GCV)	0.11 (0.06)	0.07 (0.09)	0.30 (0.11)	0.42 (0.36)	0.91 (0.17)	0.72 (0.23)
Krige (UK)	0.15 (0.11)	0.25 (0.33)	0.37 (0.20)	0.46 (0.32)	0.68 (0.23)	0.47 (0.28)
Random Forest	0.16 (0.06)	0.31 (0.18)	0.41 (0.12)	0.89 (0.15)	0.47 (0.14)	0.46 (0.09)
Krige (OK)	0.26 (0.31)	0.24 (0.33)	0.39 (0.24)	0.45 (0.32)	0.70 (0.23)	0.47 (0.28)
KNNreg	0.19 (0.10)	0.29 (0.26)	0.44 (0.16)	0.92 (0.29)	0.47 (0.34)	0.70 (0.19)
TPS	0.15 (0.07)	0.23 (0.24)	0.38 (0.14)	0.60 (0.35)	1.01 (0.23)	0.78 (0.19)
GBM	0.22 (0.07)	0.65 (0.36)	0.49 (0.13)	0.97 (0.08)	0.47 (0.24)	0.46 (0.08)
DSA	0.25 (0.05)	0.72 (0.25)	0.53 (0.08)	1.03 (0.15)	0.68 (0.11)	0.48 (0.08)
GAM	0.24 (0.02)	1.05 (0.08)	0.49 (0.04)	1.02 (0.09)	0.62 (0.08)	0.49 (0.12)
GLMnet	0.37 (0.01)	1.01 (0.02)	0.67 (0.03)	0.99 (0.03)	0.67 (0.03)	0.44 (0.03)
GLM	0.37 (0.01)	1.02 (0.03)	0.67 (0.02)	0.98 (0.03)	0.67 (0.03)	0.44 (0.03)
Polymars	0.28 (0.10)	0.94 (0.30)	0.60 (0.19)	1.11 (0.25)	0.78 (0.20)	0.64 (0.34)
SVM	0.49 (0.28)	0.87 (0.27)	0.71 (0.20)	1.05 (0.15)	0.80 (0.19)	0.66 (0.33)
GP	0.28 (0.10)	0.64 (0.42)	0.57 (0.20)	1.31 (0.61)	1.01 (0.63)	0.81 (1.02)
Mean	1.00 (0.01)	1.01 (0.01)	1.01 (0.01)	1.00 (0.01)	1.01 (0.01)	1.01 (0.01)

Table 2.4: Average FVU (standard deviation in parentheses) by spatial process.

The function f_4 was a smooth, heterogeneous process. TPS (GCV) (average FVU = 0.42), Super Learner (0.43), Ordinary Kriging (0.45), and Universal Kriging (0.46) all performed similarly.

The function f_5 was a clustered, rough surface we would expect to be well-suited to K nearest neighbors, GBM, and Random Forest. In fact, all three of these algorithmic classes had nearly identical performances, with an average FVU of 0.47. Super Learner, however, had an average FVU of 0.22 (SD = 0.14), which was dramatically better than any of the other algorithmic classes. The Ordinary (average FVU = 0.70) and Universal (0.68) Kriging algorithms had similar average performances to GAM (0.62), GLM (0.67), GLMnet (0.67), and DSA (0.68). Not surprisingly, TPS (GCV) and TPS with fixed λ did poorly, with average FVUs of 0.91 and 1.01, respectively.

f_6 was a somewhat rough surface constructed from a Gaussian random field and point-source distance decay functions. As expected, Kriging with trend $w_1, \ldots, w_6$ had the best performance on average, with an FVU of 0.25 (SD = 0.14), closely followed by Kriging with trend $\mathbf{s}, w_1, \ldots, w_6$ (average FVU = 0.26, SD = 0.15). Super Learner had the next best average performance, with an average FVU of 0.31 (SD = 0.19). GLM, GLMnet, GBM, Random Forest, the Ordinary and Universal Kriging algorithms, and DSA all performed similarly slightly less well, with average FVUs from 0.44 to 0.48. The TPS (GCV) and TPS with fixed λ were at a disadvantage given the roughness of the surface, with average FVUs of 0.72 and 0.78, respectively.

These simulation results clearly illustrate some of the chief advantages of Super Learner as a spatial predictor. For surfaces that were perfectly suited for one or more base learners in the library, Super Learner either performed almost as well as the best

base learner, or it outperformed its library. For more complex, rougher surfaces, Super Learner performed significantly better than any single base learner in the library. It had the best overall performance even at the smallest sample size, and appeared to be relatively insensitive to sampling strategy.

2.6 Practical Example: Predicting Lake Acidity

We applied Super Learner to a lake acidity data set previously analyzed by Gu [2002] and Huang and Chen [2007]. Increases in water acidity are known to have a deleterious effect on lake ecology. Having an accurate estimate of the spatial distribution of lake acidity is an essential first step toward crafting effective regulatory interventions to control it. The data were sampled by the U.S. Environmental Protection Agency during the Fall of 1984 in the Blue Ridge region of the Southeastern United States [Ellers et al., 1988], and consist of longitudes and latitudes (in degrees), calcium ion concentrations (in milligrams per liter), and pH values. The EPA used a systematic stratified sampling design which is treated as fixed here. Because only one sample per lake was collected, we assume some measurement error that is independent of lake pH, calcium ion concentration, and spatial location. The data are freely available in the R package gss [Gu, 2012]. The same nearly equal area projection was used as in Gu [2002] and Huang and Chen [2007],

$$x_1 = \cos((\pi x_{lat})/180)\sin(\pi(x_{lon} - \overline{x}_{lon})/180)$$
$$x_2 = \sin(\pi(x_{lat} - \overline{x}_{lat})/180),$$

where $\overline{x}_{lat}$ and $\overline{x}_{lon}$ are the midpoints of the latitude and longitude ranges, respectively.

Let $\mathbf{x}_i = (x_{i,1}, x_{i,2})$ denote the i^{th} sampling location; w_i denote the calcium ion concentration observed at the i^{th} sampling location; and Y_i^* be the pH value observed at the i^{th} sampling location. We assume that $\mathbf{E}[Y_i^*|\mathbf{S}_i = \mathbf{s}] = Y(\mathbf{s})$, where $\mathbf{S}_i = (\mathbf{x}_i, w_i)$. Our objective is to learn the lake pH spatial process from the data.

The library used to predict lake acidity was similar in composition to the simulation library described in subsection 2.5.1, with some important differences. The number of parameterizations for some of the algorithm classes in the library was reduced. We used one DSA learner, which used 10-fold cross-validation and considered polynomials of up to five terms ($m = 5$), each term being at most a two-way interaction ($k = 2$) with a maximum sum of powers $p = 3$. We used a reduced number of parameterizations of GAM, GBM, TPS, GP, and SVM learners, as well. We also included screening algorithms that allowed us to train learners on specific subsets of covariates: $\mathbf{x}$, w, $\log w$, $(\mathbf{x}, w)$, and $(\mathbf{x}, \log w)$. We considered the L_2 loss function, and the predictions from all base learners were truncated to the observed pH range in order to ensure a uniformly bounded loss.

Table B.2 in Appendix B provides a detailed list of the library and shows performance results for each base learner as well as Super Learner itself. Figure 2.2 provides graphical representations of Super Learner's pH predictions. Many of the algorithms in the library performed slightly better when given $\log w$ as opposed to

w, but for those algorithms like GBM and Random Forest that were not attempting to fit some kind of polynomial trend, logging the calcium ion concentration made little difference in performance. This is not surprising, given the relationship between the activities of Ca^2+ and H^+. As expected, most algorithms had cross-validated risk estimates that were worse than their empirical risk estimates calculated from predictions made after training on the full data set. The Kriging algorithms, for instance, were all exact interpolators when trained on the full data, and thus had estimated empirical MSEs of 0, whereas their MSEs estimated via cross-validation ranged from 0.07 (FVU = 0.46) to 0.11 (FVU = 0.72). The Gaussian processes with RBF kernel had the most pronounced differences between the two risk estimates. For example, GP (RBF) trained on the covariates $(\mathbf{x}, w)$ had an empirical MSE of 0.01 (FVU = 0.08) and a cross-validated MSE of 0.22 (FVU = 1.46).

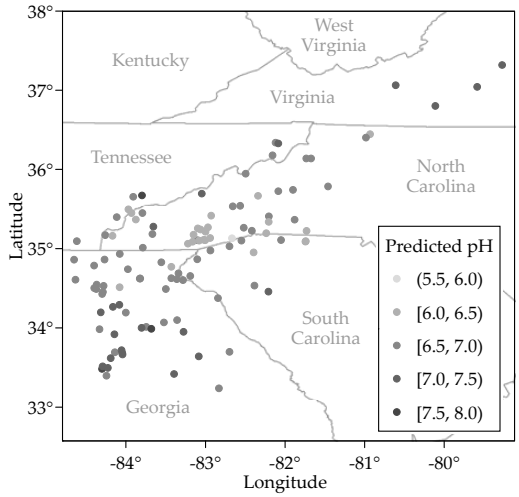
The Super Learner algorithm gave non-zero weights to the predictions of eight base learners from five different algorithm classes: GBM, KNNreg, Kriging, Random Forest, and SVM (polynomial kernel). While the largest weight went to an exactly interpolating algorithm (Kriging with trend term $\log w$, $\beta = 0.58$), Super Learner pH predictions are a slightly smoothed version of the observed data, with attenuated predictions for the highest and lowest observations. One should use caution in general when interpreting the weights returned by the Super Learner algorithm. It can be tempting to use them to try to draw conclusions about the underlying data generating process. However, these weights are not necessarily stable. We would expect them to differ across different samples, or across different initial random seeds that determine the folds in the cross-validation procedure.

2.7 Discussion and Future Directions

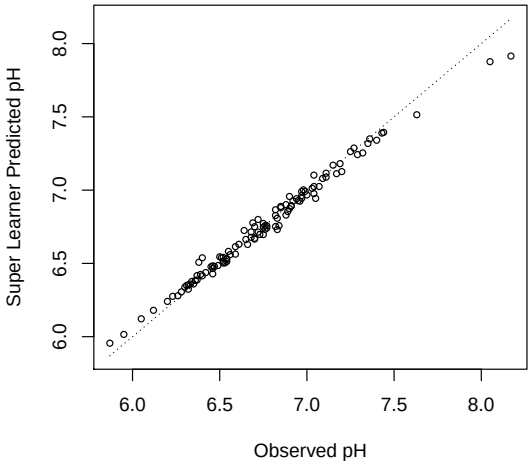
This chapter demonstrated the use of an ensemble learner for spatial prediction that uses cross-validation to optimally combine the predictions from multiple, heterogeneous base learners. We have reviewed important theoretical results giving performance bounds that imply Super Learner will perform asymptotically at least as well as the best candidate in the library. We discussed the assumptions required for these optimality properties hold in the context of spatial data. These assumptions are reasonable for many measurement error scenarios and commonly implemented spatial sampling designs, including various forms of stratified and random regular sampling.

Our simulations and practical data analysis used comparatively modest sample sizes. With the increased availability of massive environmental sensor networks and extremely large remotely sensed imagery, scalability of a spatial prediction method is often an important practical concern. The cross-validation step of Super Learner is a so-called 'embarrassingly parallel'[1] problem, and there are implementations of Super Learner that parallelize this step (see for instance the function CV.SuperLearner in the SuperLearner R package). There are also two versions of the Super Learner

[1] An embarrassingly parallel problem is one for which little or no effort is required to separate the problem into a number of parallel tasks.



(a)



(b)

Figure 2.2: (a) A map of Super Learner's pH predictions, and (b) a plot of Super Learner's predictions as a function of the observed data. Super Learner mildly attenuated the pH values at either end of the range, but otherwise provided a fairly close fit to the data.

algorithm under active development that are intended specifically to cope with very large data sets: one has been modified to work with streaming data [sam]; and another, h2oEnsemble (http://www.stat.berkeley.edu/∼ledell/software.html), is a stand-alone R package that makes use of a Java-based machine learning platform.

Dependent sampling designs, where sampling at one point changes the probability of sampling at another point, have not been addressed here. This is an important area for future research. We also limited our scope to the case where measurement error is at least conditionally mean-zero. Spatially structured measurement error that is not conditionally mean zero can be a problem in some spatial prediction applications, and there have been a number of attempts to alter the cross-validation procedure to accommodate it [Francisco-Fernandez and Opsomer, 2005, Carmack et al., 2009]. These proposed techniques generally require one to estimate the error correlation structure from the data or to know it a priori. How well these algorithms perform if the correlation extent is substantially underestimated is unknown. Ideally, it would be best to have a stronger theoretical understanding of how the degree of dependence between training and validation sets affects cross-validated risk estimates both asymptotically and in finite samples. This is another important future area for research.

Chapter 3

Sieve Plateau Variance Estimators: a New Approach to Confidence Interval Estimation for Dependent Data

3.1 Introduction

For the sake of defining the challenge addressed in this paper, first suppose we observe realizations of n independent and identically distributed (i.i.d.) random variables. Consider a particular estimator ψ_n of a specified target parameter ψ_0. Statistical inference can now proceed in a variety of ways. Suppose we can prove the estimator is asymptotically linear with a specified influence function, i.e. $\psi_n - \psi_0$ can be written as an empirical mean of the influence function applied to the observation, plus a second order term assumed to converge to zero in probability at a rate faster than $\sqrt{n}$. In that case, it is known that the $\sqrt{n}$-standardized estimator converges to a normal distribution with asymptotic variance σ^2 equal to the variance of the influence function. An estimator of σ^2, which we denote σ_n^2, provides an asymptotic 0.95-confidence interval $\psi_n \pm 1.96\sqrt{\sigma_n^2/n}$. One way to estimate σ^2 is to estimate the influence function and set σ_n^2 equal to the sample variance of the estimated influence function values. We could also use other approaches such as the nonparametric bootstrap or subsampling.

In this paper, we are concerned with obtaining valid statistical inference when the data are known to be dependent, but the precise nature of that dependence is unknown. Specifically, we are interested in a method that will work well for estimators of relatively complex parameters one finds in semiparametric causal inference applications. We assume throughout that the estimator behaves in first order like an empirical mean of dependent random variables. We refer to such an estimator as (generalized) asymptotically linear and these random variables as (generalized) influence functions. In addition, we assume the dependence between influence functions is sufficiently weak so that the $\sqrt{n}$-standardized estimator converges to a normal distribution with mean zero and variance σ_0^2. We focus on time dependence, as it is well-represented in the literature. However,the methods we propose are generally applicable. Dependence could be spatiotemporal, for example, or over a poorly understood network. We discuss such extensions throughout. We limit ourselves to the case of positive covariances, but again,

the method has a natural extension to general covariance structures.

Numerous blocked bootstrap and subsampling approaches have been developed to accommodate unknown time dependence, and there are comprehensive book-length treatments of both (see for example Lahiri [2013] for blocked bootstrap approaches and Politis et al. [1999] for subsampling). These approaches involve estimating a tuning parameter b. In the context of blocked bootstraps, b corresponds to the size of the contiguous blocks resampled with replacement, or to the geometric mean of the block size in the case of stationary block bootstrapping. Blocked bootstraps have been shown to perform well when the optimal b is known or can be effectively estimated from the data. However, these estimators are sensitive to the choice of b, which is frequently difficult to estimate. The bootstrap approach also relies on some important regularity conditions not always met by all asymptotically linear, normally distributed estimators. For example, if the influence function depends on the true data generating distribution through densities, there is a literature warning against the application of a nonparametric bootstrap, and refinements and regularizations will be needed. See Mammen [1992] for a comprehensive discussion and examples.

In subsampling, b corresponds to the size of the contiguous subsample. One of subsampling's most attractive features is that it requires very few assumptions: the size of the subsample must be such that as sample size $n \to \infty$, $b \to \infty$ and $b/n \to 0$; and the standardized estimator must converge to some limit distribution. One need not know the rate of convergence nor the specific limit distribution. However, finite sample performance can be heavily dependent on the choice of b, which must be large enough to capture dependence, yet small enough to adequately approximate the underlying target distribution. Finding an optimal b for any given estimator is a nontrivial undertaking [Politis and Romano, 1993], and for more complex estimators, the sample sizes required in order to adequately estimate ψ_0 in each subsample can be impractically large.

We present a method of learning from sequences of ordered, sparse covariance structures on influence functions, where dependence decreases monotonically with distance. We assume there exists an (unknown) distance threshold $\boldsymbol{\tau}_{0,t}$ for each time point t such that any observation farther than $\boldsymbol{\tau}_{0,t}$ away from observation t is independent from observation t. Our proposed procedure seeks to select a variance estimate close to what we would have obtained had we known the true distance thresholds $(\boldsymbol{\tau}_{0,t} : t = 1, ..., n)$. Assume for a moment this dependence structure is constant across time, i.e. $\boldsymbol{\tau}_{0,t}$ is equal to some positive integer $\boldsymbol{\tau}_0$ for all t. Theory tells us a variance estimator ignoring this dependence will result in a biased estimate, and the magnitude of this bias will decrease as the number of nonzero covariances used in the variance estimate increases, until all true nonzero covariances are incorporated. Estimates assuming nonzero covariances beyond this will be unbiased, but will become more variable. This simple insight provides the rationale for our proposed approach. Intuitively, Sieve Plateau (SP) variance estimation searches for a 'plateau' in a sequence of variance estimates that assume increasing numbers of nonzero covariances. While our approach requires stronger assumptions than subsampling, its performance does not depend heavily on additional tuning parameters. It can also be used with complex

estimators that require substantial sample sizes for proper estimation, and in settings where contiguity of the dependence structure is incompletely understood.

The remainder of this paper is organized as follows. In section 2, we define the formal estimation problem. In section 3, we introduce SP variance estimation and the intuitive rationale behind the approach. In section 4, we provide a more formal justification for why our proposed method works and state conditions upon which our method relies. Section 5 describes several specific implementations of our proposed approach within the context of estimating the variance of a sample mean of a time series. We also present results of an extensive simulation study, which demonstrate our approach works well in practice and consistently outperforms subsampling and blocked bootstrapping in a context where they are known to perform well. In section 6, we present a real data analysis, estimating the Average Treatment Effect (ATE) of ventilation rate on illness absence in an elementary school classroom. We also discuss why subsampling and blocked bootstraps are ill-suited to this particular estimation problem. We conclude with a discussion.

Notation conventions. The distance between two points x and y is denoted $d(x, y)$. Parameters with subscript 0 are features of the true data probability distribution. Subscript n indicates an estimator or some quantity that is a function of the empirical distribution. If f is a function of the observed data and P a possible probability distribution of the data, then Pf is the expectation of f taken with respect to P.

3.2 Target Parameter

Let $O = (O_1, \ldots, O_n)$ be a data set consisting of n time-ordered observations on a certain random process. Let P_0^n be the probability distribution of O. Let ψ_0 be a real valued feature of P_0^n, i.e. $\psi_0 = \Psi(P_0^n)$ for some mapping Ψ from the statistical model for P_0^n into the real line. We are given an estimator ψ_n based on O. We assume dependence structure in O is sufficiently weak so that ψ_n satisfies an expansion

$$\psi_n - \psi_0 = \frac{1}{n} \sum_{i=1}^{n} D_i(O; \theta_0) + r_n,$$

where r_n is a second order term assumed to converge to zero in probability at a rate faster than $\sqrt{n}$. The functions $D_i(O; \theta_0)$ are called influence functions. They have expectation zero under P_0^n and depend on the unknown P_0^n through some parameter θ_0. For clarity, we often notate them as $D_i(\theta_0)$, but it is important to remember that they are functions of O. Assume there is a function f that assigns a measure $S = f(O)$ to each unit observation O, and define $s_i = f(O_i)$, $i = 1, \ldots, n$. Assume for all s_i there exists a bounded $\boldsymbol{\tau}_{0,i}$ such that $d(s_i, s_j) > \boldsymbol{\tau}_{0,i}$ implies the covariance $P_0^n D_{0,i} D_{0,j} = 0$. Define $\Omega_{\boldsymbol{\tau}_{0,i}} = \Omega_{0,i}$ to be the set of j such that $d(s_i, s_j) \leq \boldsymbol{\tau}_{0,i}$. Define

$$\sigma_{0n}^2 \equiv \text{VAR} \left\{ \frac{1}{\sqrt{n}} \sum_{i=1}^{n} D_i(\theta_0) \right\} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \Omega_{0,i}} P_0^n \left\{ D_i(\theta_0) D_j(\theta_0) \right\}. \tag{3.1}$$

In many cases one might also assume $\sigma_{0n}^2 \to \sigma_0^2$ in probability for some fixed σ_0^2, but this is not necessary. We assume that as $n \to \infty$, $\mathbb{E}\big(r_n\sqrt{n}\big)^2 \to 0$ and $\sigma_{0n}^{-1}Z(n) \equiv \sigma_{0n}^{-1}/\sqrt{n}\sum_{i=1}^n D_i(\theta_0) \Rightarrow_d N(0,1)$. These assumptions imply that as $n \to \infty$, the standardized estimator converges weakly to a mean zero normal distribution with variance one, i.e.

$$\sigma_{0n}^{-1}\sqrt{n}(\psi_n - \psi_0) = \sigma_{0n}^{-1}Z(n) + o_P(1) \Rightarrow_d N(0,1).$$

If σ_{0n}^2 converges to a fixed σ_0^2, then $Z(n) \Rightarrow_d N(0,\sigma_0^2)$ and $\sqrt{n}(\psi_n - \psi_0) \Rightarrow_d N(0,\sigma_0^2)$. Thus $1/\sqrt{n}$ denotes the rate at which ψ_n converges to ψ_0. This paper is concerned with estimating σ_{0n}^2 so we can construct an asymptotic 0.95-confidence interval $\psi_n \pm 1.96\sqrt{\sigma_n^2/n}$, where σ_n^2 is an estimator of σ_{0n}^2.

Consider estimators of the form

$$\sigma_n^2(\boldsymbol{\tau}) \equiv \frac{1}{n}\sum_{i=1}^n\sum_{j=1}^n \delta_{\boldsymbol{\tau}}(i,j)D_i(\theta_n)D_j(\theta_n), \tag{3.2}$$

where $\boldsymbol{\tau}$ in a parameter space T is an n-dimensional vector defining

$$\delta_{\boldsymbol{\tau}}(i,j) = I\{d(s_i,s_j) < \boldsymbol{\tau}_i\}.$$

Our conditions require the size (as measured by entropy) of the parameter space T remains controlled as $n \to \infty$. An extreme example would be $\boldsymbol{\tau}$ such that $\boldsymbol{\tau}_i = \lambda$ for all $i = 1,\ldots,n$ and a constant $\lambda > 0$. More generally, one might parametrize $\boldsymbol{\tau}_i = \boldsymbol{\tau}_i(\lambda)$ for a fixed dimensional parameter λ, or one might assume T is a union of a finite number of such parametric subsets. An influence function (IF) based oracle estimator of (3.1) is defined as (3.2) using $\delta_{\boldsymbol{\tau}} = \delta_{\boldsymbol{\tau}_0}$.

It will be convenient for us to define the notion of an arbitrary $\boldsymbol{\tau}$ being 'at least as large as' $\boldsymbol{\tau}_0$. Let $\Omega_{\boldsymbol{\tau}_i} = \{j : d(s_i,s_j) \le \boldsymbol{\tau}_i\}$ be the set defined by $\boldsymbol{\tau}_i$. Define $T_0 \subset T$ as the set of $\boldsymbol{\tau}$ where $\delta_{\boldsymbol{\tau}}(i,j) \ge \delta_{\boldsymbol{\tau}_0}(i,j)$ for all i,j. Thus, T_0 contains $\boldsymbol{\tau}_0$ and all other $\boldsymbol{\tau}$ such that for all i, any element in $\Omega_{\boldsymbol{\tau}_0,i}$ is also in $\Omega_{\boldsymbol{\tau}_i}$. When we say $\boldsymbol{\tau}$ 'contains' $\boldsymbol{\tau}_0$, we mean $\boldsymbol{\tau}$ is an element of T_0. In classical time series where dependence decays over time, it also makes sense to say $\boldsymbol{\tau}$ is at least as large as $\boldsymbol{\tau}_0$ when $\boldsymbol{\tau} \in T_0$.

3.3 Sieve Plateau Estimators

Suppose we have a collection of vectors $(\boldsymbol{\tau}_{n,k} : k)$, a proportion of which are in T_0. Consider the associated collection of variance estimators $(\sigma_n^2(\boldsymbol{\tau}_{n,k}) : k)$. Under conditions similar to those required for asymptotic normality of ψ_n, estimators based on $\boldsymbol{\tau}_{n,k} \in T_0$ will be unbiased (see Theorem 2). Suppose we could order these variance estimators so that they start out making too many independence assumptions and end up making too few. A smoothed version of this ordered sequence would give a curve with a plateau in the tail. In particular, since we are assuming all true covariances between influence functions are positive, we would expect this curve to be monotone increasing. We propose finding the plateau of this smoothed curve and using this knowledge to estimate σ_{0n}^2. The general approach is as follows.

Construct a sieve of variance estimators.

1. Formulate a set of vectors $\{\boldsymbol{\tau}_{n,k} : k\}$ in T that covers a range of covariance structures starting with independence of all influence functions or some other appropriate lower bound and ending with covariance structures $\boldsymbol{\tau} \in T_0$. These vectors can reflect true knowledge about the dependence in one's data. For instance, in the case of time dependence, if one knows the dependence lag between all influence functions is constant, a simple sequence of vectors could assume increasing constant dependence lags, starting with 0 and ending with an upper bound $\boldsymbol{\tau}'$. Alternatively, one might believe the dependence structure is either constant or fluctuates smoothly over time according to seasonal trends. Note it is not necessary for $\boldsymbol{\tau}_0$ to be an element of $\{\boldsymbol{\tau}_{n,k} : k\}$. We describe more complex approaches to generating $\boldsymbol{\tau}$ vectors in section 5.

2. For each $\boldsymbol{\tau}_{n,k}$, compute $\sigma_n^2(\boldsymbol{\tau}_{n,k})$ as defined in equation (3.2).

3. Order the variance estimates so that they become increasingly unbiased for σ_0^2. A valid ordering need not be perfect. For example, we could define a matrix $M_{\boldsymbol{\tau}_{n,k}}$, whose (i,j)-th element is $D_i(\theta_n)D_j(\theta_n)$ if $\delta_{\boldsymbol{\tau}_{n,k}}(i,j) = 1$ and zero otherwise. One could order the variance estimates by L_1 fit (average of absolute values) between its corresponding $M_{\boldsymbol{\tau}_{n,k}}$ and the matrix that assumes independence, starting with the smallest estimated L_1 fit and ending with the largest. We use this ordering in our simulation study. One could also order according to another complexity criterion. In our practical data analysis, for example, we compare L_1 fit ordering with ordering by a rough approximation of the variance of each estimator $\sigma_n^2(\boldsymbol{\tau}_{n,k})$ in the sequence (see equation (3.3)). One could also order by the number of nonzero pairs directly. This works reasonably well in time series, but could be more problematic in settings with higher dimensional dependence. We implemented this ordering in our practical data analysis, as well.

Find the plateau where variance estimators are unbiased.

4. Estimate a monotone increasing curve from the ordered sequence using weighted isotonic regression. We use Pooled Adjacent Violators (PAV) for this purpose (Turner [2013]; Robertson et al. [1988]), weighting each variance estimate according to the inverse of an approximation of its variance (see equation (3.3)). Here we rely on the fact that covariances are known to be positive. Without this assumption, one would have to use other nonparametric regression methods.

5. Find the plateau where the variance estimates are no longer biased. To do this, we estimate the mode of the PAV smoothed curve by first estimating its density using a Gaussian kernel with bandwidth selected according to Silverman [1986] (pg 48, eqn. 3.31). Then we take the maximum of that estimated density. We have found this approach works well under a wide range of sieves and true dependence structures. We can use this estimated mode as a variance estimate in its own

right, or we can use it to locate the beginning of the plateau and then take as our estimate either the value of the PAV curve at this location or the value of the closest sample variance estimate to that location. We refer to estimators using the mode as 'mode-based'; those using the value of the PAV curve as 'step-based'; and those using the closest sample variance estimate in the sequence as 'sample-based'. In principle, because the mode-based estimator takes advantage of additional averaging, we would expect it to be less variable than the sample-based estimator. Our simulation results and practical data analysis appear to confirm this intuition. We have observed that the difference between step-based and mode-based estimators is negligible under the three orderings mentioned above.

In practice, the choice of upper bound $\boldsymbol{\tau}'$ is critical to the success of this procedure. It must be true that $\boldsymbol{\tau}'$ approximates elements in T_0 as $n \to \infty$. Preferably, the tail of the ordered sequence contains a number of $\boldsymbol{\tau}_{n,k}$ that approximate elements in T_0. In many applications where one feels comfortable assuming some form of central limit theorem, a sufficiently maximal $\boldsymbol{\tau}'$ will not be difficult to formulate.

3.3.1 Variance of Variance Estimators

For the sake of determining the weights in the PAV-algorithm, we need a reasonable approximation of the variance of each variance estimator in the sieve. We use the following shorthand notation in this section. Let D_{ij} denote $D_i(\theta)D_j(\theta)$, and $D_{ijk\ell}$ denote $D_i(\theta)D_j(\theta)D_k(\theta)D_\ell(\theta)$. Let Ω_{ij} be the union of Ω_i and Ω_j. In the context of time-ordered dependence, this would correspond to the union of the two time intervals $[O(i-\boldsymbol{\tau}_i), ..., O(i)]$ and $[O(j-\boldsymbol{\tau}_j), ..., O(j)]$ implied by a candidate $\boldsymbol{\tau}$. Define an indicator function $\gamma(i,j,k,\ell) \equiv I\{\Omega_{ij} \cap \Omega_{k\ell} = \emptyset\}$, which equals one when the intersection between Ω_{ij} and $\Omega_{k\ell}$ is empty. We have found that the following metric works well in practice as an approximation of the variance of variance estimators of the form (3.2).

$$\frac{1}{n^2}\sum_{i,j,k,\ell}\{1-\gamma(i,j,k,\ell)\}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell) \tag{3.3}$$

This choice is inspired by formal calculations approximating the true variance and the use of working models for the components of this approximation. We provide more detail in Appendix A. In general, given that we have assumed the covariance between unit observations is positive and decreases with distance, the variance of variance estimators defined in equation (3.2) is roughly driven by the number of dependent pairs $\{(i,j) : \delta_{\boldsymbol{\tau}}(i,j) = 1\}$ as well as the number of intersecting non-empty unions Ω_{ij}. We also propose using (3.3) as an ordering scheme, as it will tend to put the more unbiased estimators in the tail of the sequence.

At first glance, one might assume calculating (3.3) for a sizable sequence of variance estimates to be computationally impractical. This is certainly true if one uses a naive approach. We have found that careful attention to symmetry, among other things, can reduce computation time by several thousand fold. We discuss this further and

provide optimized code suitable for generalized dependence structures in this paper's supplementary materials.

3.4 Supporting Theory

Theorem 1 establishes conditions under which a generalized SP-variance estimation procedure is consistent for σ_{0n}^2. Beyond an ordering of the sequence concentrating unbiased variance estimators in the tail (a model assumption), the consistency of the SP-variance estimator relies on uniform consistency of $(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$, a process indexed by $\boldsymbol{\tau} \in T$, as an estimator of its limit process $(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$, where $\sigma_{0n}^2(\boldsymbol{\tau}) = \sigma_{0n}^2$ if $\boldsymbol{\tau} \in T_0$. Since this uniform consistency condition is nontrivial, Theorem 2 considers a particular dependence structure on the influence functions for which we formally establish uniform consistency and consistency of the variance estimator under entropy conditions restricting the size of T.

Theorem 1. *Recall $T = T_0 \cup T_0^c$. Assume the following.*

1. $\sup_{\boldsymbol{\tau} \in T} |\sigma_n^2(\boldsymbol{\tau}) - \sigma_{0n}^2(\boldsymbol{\tau})| \to 0$ *in probability for a $\sigma_{0n}^2(\boldsymbol{\tau})$ process, where $\sigma_{0n}^2(\boldsymbol{\tau}) = \sigma_{0n}^2$ for $\boldsymbol{\tau} \in T_0$.*
2. *We are given a mapping $\hat{\boldsymbol{\tau}}$ that takes as input a process $(\sigma^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ and maps it into an element of T. This mapping has the following properties:*
   a) *$\hat{\boldsymbol{\tau}}(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T) \in T_0$ with probability tending to 1;*
   b) *continuity: $\hat{\boldsymbol{\tau}}(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T) - \hat{\boldsymbol{\tau}}(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ converges to zero in probability.*

Then $\boldsymbol{\tau}_n \equiv \hat{\boldsymbol{\tau}}(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ converges to $\hat{\boldsymbol{\tau}}(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T) \in T_0$ with probability tending to 1, and thus $P(\boldsymbol{\tau}_n \in T_0) \to 1$ and $\sigma_n^2(\boldsymbol{\tau}_n) - \sigma_{0n}^2 \to 0$ in probability as $n \to \infty$.

Proving Theorem 1 is straightforward. Note that $\hat{\boldsymbol{\tau}}$ represents the algorithm that is able to map the limit process $(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ into a desired plateau value $\sigma_{0n}^2(\boldsymbol{\tau}_n^0)$ with $\boldsymbol{\tau}_n^0$ in T_0, where T_0 defines the plateau of $(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$. The additional continuity condition 2b on $\hat{\boldsymbol{\tau}}$, combined with the uniform consistency of $(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ to $(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$, then establishes that $\sigma_n^2(\boldsymbol{\tau}_n)$ is consistent for σ_{0n}^2. In our concrete example of the SP method, we have that $\hat{\boldsymbol{\tau}}$ is given by the composition of (1) ordering the set of variance estimators $(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau})$ in some way, (2) applying the weighted PAV algorithm to this ordered sequence, and (3) finding its plateau value.

The conditions of Theorem 1 highlight an underlying tension between selecting T large enough to ensure it contains a nonempty T_0, but not so large that $(\sigma_n^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ fails to approximate its limit process $(\sigma_{0n}^2(\boldsymbol{\tau}) : \boldsymbol{\tau} \in T)$ *uniformly* in T. It also demonstrates the concrete challenge in coming up with $\hat{\boldsymbol{\tau}}$. One needs to construct an algorithm that can map an unordered set of variance estimators, a subset of which are consistent for σ_{0n}^2, into a consistent variance estimator.

Theorem 2 shows that under a particular type of dependence structure on the influence functions, the crucial uniform consistency assumption in Theorem 1 holds,

and $\sigma_n^2(\boldsymbol{\tau}_n)$ is consistent for the true variance σ_{0n}^2 as long as $\boldsymbol{\tau}_n - \boldsymbol{\tau}_0^n \to 0$ for a $\boldsymbol{\tau}_0^n \in T_0$ with probability tending to 1. Theorem 2 also gives a good rate of convergence for the variance estimator (essentially $1/\sqrt{n}$ in many applications). One could weaken the entropy condition stated below while still preserving the asymptotic consistency of the variance estimator, but with a slower rate. However, for the sake of having reliable confidence intervals in finite samples, a good rate of convergence for $\sigma_n^2(\boldsymbol{\tau}_n)$ is important.

Theorem 2. *Let $\mathcal{F} = \{(\boldsymbol{\tau}, \theta) : \boldsymbol{\tau} \in T, \theta \in \Theta\}$. Assume $(\theta_n, \boldsymbol{\tau}_n) \in \mathcal{F}$ with probability tending to 1. Let $\| \cdot \|_{\mathcal{F}}$ be some norm or semimetric on $\mathcal{F}$. Define $\boldsymbol{\tau}^u$ to be such that for all $\boldsymbol{\tau} \in T$, $\boldsymbol{\tau} \leq \boldsymbol{\tau}^u$. Assume $\max_i |\Omega_{i,\boldsymbol{\tau}^u}| < K$ for some $K < \infty$. Define the index set $A = \{(i,j) : i \in \{1, \ldots, n\}, j \in \Omega_{i,\boldsymbol{\tau}^u}\}$. Define*

$$d_n[(\boldsymbol{\tau}, \theta), (\boldsymbol{\tau}_1, \theta_1)] \equiv \frac{1}{n} \sum_{(i,j) \in A} P_0^n \{\delta_{\boldsymbol{\tau}}(i,j) D_{ij}(\theta_n) - \delta_{\boldsymbol{\tau}_1}(i,j) D_{ij}(\theta_1)\}.$$

Let $\sigma_n^2(\boldsymbol{\tau}) = 1/n \sum_{(i,j) \in A} \delta_{\boldsymbol{\tau}}(i,j) D_{ij}(\theta_n)$, and $\sigma_{0n}^2(\boldsymbol{\tau}) = 1/n \sum_{i=1}^n \sum_{j \in \Omega_{i,\boldsymbol{\tau}}} P_0^n \{D_{ij}(\theta_0)\}$. Consider an estimator $\boldsymbol{\tau}_n$ and corresponding variance estimator $\sigma_n^2(\boldsymbol{\tau}_n)$ of σ_{0n}^2.

Assume:

1. *$d_n[(\boldsymbol{\tau}_n, \theta_n), (\boldsymbol{\tau}_n^0, \theta_0)] \xrightarrow{n \to \infty} 0$ in probability, where $P_0^n(\boldsymbol{\tau}_n^0 \in T_0) \xrightarrow{n \to \infty} 1$.*
2. *$\| (\boldsymbol{\tau}_n, \theta_n) - (\boldsymbol{\tau}_n^0, \theta_0) \|_{\mathcal{F}} \to 0$ in probability as $n \to \infty$.*
3. *Sparsity: For each $(\boldsymbol{\tau}, \theta) \in \mathcal{F}$ and $(i,j) \in A \equiv \{(i,j) : \delta_{\boldsymbol{\tau}^u}(i,j) = 1\}$, there exist at most K (universal constant in n) other elements $(k, \ell) \in A$ for which $\delta_{\boldsymbol{\tau}}(i,j) D_{ij}(\theta)$ depends on $\delta_{\boldsymbol{\tau}}(k,\ell) D_{k\ell}(\theta)$.*
4. *Linkage: For each $(\boldsymbol{\tau}, \theta) \in \mathcal{F}$ and all integers $p > 0$, for all $(i,j) \in A$ we have*

$$\{P_0^n (\delta_{\boldsymbol{\tau}}(i,j) D_{ij}(\theta))^p\}^{1/p} \leq C \| (\boldsymbol{\tau}, \theta) \|_{\mathcal{F}}$$

   for a universal $C < \infty$.

5. *Bounded entropy: There exists an $\eta > 0$ so that the entropy integral*

$$\int_0^\eta \sqrt{\log N(\epsilon, \mathcal{F}, \| \cdot \|_{\mathcal{F}})} d\epsilon < \infty.$$

Then,

$$\sigma_n^2(\boldsymbol{\tau}_n) - \sigma_{0n}^2 = O_P\left\{d_n\left[(\boldsymbol{\tau}_n, \theta_n), (\boldsymbol{\tau}_n^0, \theta_0)\right]\right\} + O_P\left\{1/\sqrt{n}\right\},$$

and thus converges to zero in probability as $n \to \infty$.

As a side-result, if we also have $\sup_{\boldsymbol{\tau}} \| (\boldsymbol{\tau}, \theta_n) - (\boldsymbol{\tau}, \theta_0) \|_{\mathcal{F}} \to 0$ in probability, and

$$r(n) \equiv \sup_{\boldsymbol{\tau} \in T} d_n[(\boldsymbol{\tau}, \theta_n), (\boldsymbol{\tau}, \theta_0)] \xrightarrow{n \to \infty} 0 \tag{3.4}$$

in probability, then

$$\sup_{\boldsymbol{\tau}} | \sigma_n^2(\boldsymbol{\tau}) - \sigma_{0n}^2(\boldsymbol{\tau}) | = O_P\{r(n)\} + O_P\{1/\sqrt{n}\}.$$

Theorem 2 is proved in Appendix B. The linkage condition is a weak one, as shown in van der Laan [2014]. It allows us to link the entropy condition (5) on $\mathcal{F}$ to the desired entropy of the functions $\delta_{\boldsymbol{\tau}}(i,j)D_{ij}(\theta)$. The sparsity condition (3) essentially corresponds with assuming that $D_i(\theta)$ is independent of $(D_j(\theta) : j \notin \Omega_{\boldsymbol{\tau}^u,i})$, uniformly in θ. Our proof hinges on establishing the asymptotic equicontinuity of a process $(Z_n(\boldsymbol{\tau},\theta) : (\boldsymbol{\tau},\theta))$, the mean-zero centered empirical average of $\delta_{\boldsymbol{\tau}}(i,j)D_{ij}(\theta)$. This independence allows us to formally analyze this process. If the sparsity condition does not hold marginally, but holds by conditioning on certain characteristics of the observed data, one could simply apply this theorem with P_0^n being this conditional distribution. In that same spirit, a refined theorem that significantly weakens the sparsity condition can be obtained by first orthogonally decomposing the process $Z_n(\boldsymbol{\tau},\theta)$ into

$$Z_n(\boldsymbol{\tau},\theta) = \sum_{m=1}^{M} \mathbb{E}[Z_n(\boldsymbol{\tau},\theta) \mid \mathcal{H}(m)] - \mathbb{E}[Z_n(\boldsymbol{\tau},\theta) \mid \mathcal{H}(m-1)],$$

for an increasing sigma-field $\mathcal{H}(m)$ so that $\mathcal{H}(M) = (O_1,\ldots,O_n)$, and $\mathcal{H}(0)$ is empty; and second, applying the proof in Appendix B for weak convergence of $Z_n(\boldsymbol{\tau},\theta)$ to each m-specific process $\mathbb{E}[Z_n(\boldsymbol{\tau},\theta) \mid \mathcal{H}(m)] - \mathbb{E}[Z_n(\boldsymbol{\tau},\theta) \mid \mathcal{H}(m-1)]$ conditional on $\mathcal{H}(m-1)$. Now one only needs the sparsity assumption conditional on $\mathcal{H}(m-1)$. This is the approach used in van der Laan [2014]. The resulting more general theorem assumes the necessary conditions for each m.

3.5 Simulation Study: Variance of the Sample Mean of a Time Series

We undertook a simulation study to examine the effectiveness of our proposed SP variance estimation approach and compare its performance to other existing methods in estimating the asymptotic variance of the sample mean under several types of time-dependent data generating distributions. Section 5.1 describes the simulated time series. Section 5.2 describes the three main types of SP variance estimators we implemented and briefly outlines the existing estimators to which we compared their performance. In section 5.3, we discuss our results.

3.5.1 Simulated Time Series

We simulated three main types of moving average (MA) time series. The first consisted of processes with constant dependence lag structure over time, i.e. $O(t) = u(t)+\beta_1 u(t-1) + \beta_2 u(t-2) + \cdots + \beta_{\tau_0} u(t-\tau_0)$, $u(t) \sim N(0,1)$, $t = 1,...,n$, where the vector $\beta = (\beta_1,...,\beta_{\tau_0})$ was constant across all time points. This corresponds to a $\boldsymbol{\tau}_0(t)$ that is constant over time and equal to the length of the vector β. We used three different β vectors: 0.9, (0.9, 0.5, 0.1), and (0.9, 0.7, 0.5, 0.3, 0.1), with associated asymptotic variances of the standardized sample mean of 3.61, 6.25 and 12.25, respectively.

We also simulated two additional types of MA time series where the true dependence lag varied with time. In one simulation, $(\boldsymbol{\tau}_0(t) : t = 1,...,n)$ had positive linear drift, starting with $\boldsymbol{\tau}_0(1) = 0$ and ending with $\boldsymbol{\tau}_0(n) = 7$. The asymptotic variance of the

standardized sample mean for this type of time series is approximately 6.68. In another simulation, $(\boldsymbol{\tau}_0(t) : t)$ had a periodic structure bounded between 1 and 5, with either one or two periods. The associated asymptotic variance of the standardized sample mean for this time series type is approximately 7.12.

We simulated 16,384 instances of each time series type at each sample size of 250, 500, 750, and 1000, for a total of $393,216$ iterations in our simulation study.

3.5.2 Types of SP Variance Estimators

The idea behind SP variance estimation is quite general: generate an ordered sequence of proposed dependence relationships that can approximate the truth (a 'sieve'), then use smoothing techniques to find the plateau in the ordered variance estimates. We implemented three different SP estimators, all using an L_1 fit ordering.

Model-based $\boldsymbol{\tau}$. Perhaps one knows the dependence structure in one's data adheres to a particular pattern. For instance, one might know the dependence lags increase over time. This estimator takes advantage of such knowledge. When the true dependence lags were constant over time, $\{\boldsymbol{\tau}_{n,k} : k\}$ consisted of constant-valued n-length vectors of dependence lags, starting with constant values of zero (assuming independence) and ending with an upper bound of constant values of 10, comfortably exceeding the maximum true dependence lag value of 5 for this type of simulated time series. When true dependence lags had positive linear trend, $\{\boldsymbol{\tau}_{n,k} : k\}$ was a collection of randomly generated nondecreasing n-length vectors, each with a unique combination of starting and ending values. The starting values were bounded between 1 and 7, the ending values between 2 and 18, and the maximum span between starting and ending values was 10. When true dependence lags fluctuated periodically, $\{\boldsymbol{\tau}_{n,k} : k\}$ were all periodic n-length vectors, each with a unique combination of number of periods (ranging from 1 to 6), phase shift, and minimum and maximum values (bounded between 1 and 12 in this simulation study).

Constant $\boldsymbol{\tau}$. There are probably more instances where one has limited specific knowledge about the dependence structure in one's data other than some notion of sparsity. One approach in this case is to use a sequence of constant-valued n-length $\boldsymbol{\tau}$ vectors. We found we could improve performance by incorporating some small decision criteria regarding determining a maximum constant value. The algorithm is as follows:

1. For $\{\boldsymbol{\tau}_{n,k} : k\}$ constant in time, start with $\boldsymbol{\tau}_n(t) = 0$ for all time points and compute (3.2). Continue increasing this constant value and computing the corresponding variance estimate until the first decreasing result or the constant value exceeds some large upper bound (30 in the present study). Let $\boldsymbol{\tau}'$ be the constant dependence lag vector associated with the last variance estimate in the sequence.

2. Estimate the plateau value of the resulting sequence using weighted PAVA and density estimation. Denote this result $\widetilde{\sigma}_n^2$.

3. Let $\boldsymbol{\tau}_{\max}$ be a constant vector of the maximum credible dependence lag we expect to observe (we used 10 in this study). If $\boldsymbol{\tau}' \leq \boldsymbol{\tau}_{\max}$, then $\widetilde{\sigma}_n^2$ is our variance estimate. Otherwise, our estimate is the minimum of $\widetilde{\sigma}_n^2$ and (3.2) with $\boldsymbol{\tau} = \boldsymbol{\tau}_{\max}$.

Kitchen sink. Another reasonable response to having limited specific knowledge beyond some degree of sparsity is to generate a diverse set of vectors $\{\boldsymbol{\tau}_{n,k} : k\}$. In our simulation, at each iteration we randomly generated a large collection of about 28,985 unique dependence lag vectors that were constant, linearly increasing, linearly decreasing, periodic, and random walks bounded between zero and five.

We compared these SP variance estimators with the following methods: an oracle benchmark, (3.2) with $\boldsymbol{\tau} = \boldsymbol{\tau}_0$; variance estimates (3.2) with constant n-length vector $\boldsymbol{\tau}_{\max}$ equal to 10 for all time points; and an i.i.d. IF-based estimator, as in (3.2) with constant n-length vector $\boldsymbol{\tau}$ equal to 0 for all time points. We also compared our performance to the stationary block bootstrap (SBB), circular block bootstrap (CBB), and subsampling (SS), with block and subsample sizes selected in three different ways: (1) data-adaptively according to the algorithm in Patton et al. [2009]; (2) enforcing this data-adaptively selected block size to be at least 10; and (3) using a constant proportion as a function of sample size ($b = 0.1n$, SBB and CBB only).

3.5.3 Simulation Results

Our simulation results confirm our proposed approach works well in practice. All SP estimators outperformed all variants of the SS, CBB and SBB estimators included in our study. This was true across all sample sizes and time series types. It is possible some of this performance advantage can be attributed to less than optimal estimated block sizes. However, estimating an optimal b is difficult, and not necessarily tractable for other parameters of interest. Our simulation can be viewed as a comparison of these algorithms as they are typically used.

Table 3.1 compares average normalized MSEs, $\left(\sigma_n^2 - \sigma_0^2\right)^2 / \sigma_0^2$, across estimators and sample sizes. Overall, the SP 'kitchen sink' sample-based estimator performed best, with the corresponding mode-based estimator a close second. As expected, the mode version was slightly less variable and more biased than its sample-based counterpart. Notably, both of these estimators had normalized MSEs that were actually smaller on average than the oracle benchmarks at the smallest sample size $n = 250$. The other SP estimators were close competitors, particularly when $n \geq 250$. The modified constant $\boldsymbol{\tau}$ estimator was the least computationally intensive of the SP estimators we implemented, and performed admirably even when the simulated time series did not have constant $\boldsymbol{\tau}_0$. When the dependence lag exhibited linear drift, for instance, the modified constant $\boldsymbol{\tau}$ estimator actually outperformed the (correct) model-based $\boldsymbol{\tau}$ estimators. For very large sample sizes and a rich collection of proposed dependence relationships, the time required to compute each variance estimate and its PAV regression weight for a 'kitchen sink'-type estimator could become nontrivial. In such settings, the modified constant $\boldsymbol{\tau}$ estimator could be an attractive alternative. However, it may be difficult to implement in settings where underlying dependence is more complex or is not well understood.

estimator	overall	$n = 250$	$n = 500$	$n = 750$	$n = 1000$
oracle, IF-based	0.230 (-0.020)	0.434 (-0.038)	0.224 (-0.020)	0.150 (-0.014)	0.114 (-0.010)
SP 'kitchen sink' (sample)	0.236 (-0.019)	0.426 (-0.019)	0.230 (-0.017)	0.162 (-0.019)	0.127 (-0.020)
SP 'kitchen sink' (mode)	0.248 (-0.073)	0.424 (-0.086)	0.243 (-0.073)	0.179 (-0.069)	0.146 (-0.066)
SP constant $\boldsymbol{\tau}$	0.263 (-0.015)	0.473 (-0.022)	0.259 (-0.015)	0.179 (-0.013)	0.139 (-0.012)
SP model-based (mode)	0.274 (-0.023)	0.491 (-0.032)	0.272 (-0.022)	0.188 (-0.020)	0.146 (-0.019)
SP model-based (sample)	0.279 (0.006)	0.508 (0.004)	0.276 (0.007)	0.188 (0.006)	0.144 (0.005)
SS, $b \geq 10$	0.403 (-0.110)	0.676 (-0.141)	0.403 (-0.111)	0.294 (-0.098)	0.238 (-0.088)
SS, data-adaptive b	0.407 (-0.115)	0.680 (-0.153)	0.406 (-0.116)	0.298 (-0.101)	0.243 (-0.091)
CBB, $b \geq 10$	0.408 (-0.114)	0.669 (-0.148)	0.409 (-0.116)	0.305 (-0.102)	0.250 (-0.091)
CBB, data-adaptive b	0.413 (-0.120)	0.675 (-0.161)	0.413 (-0.121)	0.309 (-0.105)	0.255 (-0.094)
SBB, $b \geq 10$	0.492 (-0.139)	0.807 (-0.183)	0.490 (-0.140)	0.368 (-0.122)	0.304 (-0.111)
SBB, data-adaptive b	0.499 (-0.151)	0.802 (-0.205)	0.502 (-0.152)	0.379 (-0.130)	0.313 (-0.117)
$\boldsymbol{\tau}_{max}$, IF-based	0.592 (-0.045)	1.109 (-0.086)	0.578 (-0.043)	0.387 (-0.030)	0.295 (-0.022)
CBB, $b = 0.1n$	0.987 (-0.125)	1.028 (-0.146)	0.987 (-0.123)	0.971 (-0.117)	0.961 (-0.112)
SBB, $b = 0.1n$	1.186 (-0.204)	1.251 (-0.226)	1.186 (-0.202)	1.160 (-0.198)	1.149 (-0.192)
iid, IF-based	3.600 (-0.677)	3.625 (-0.679)	3.599 (-0.677)	3.591 (-0.676)	3.584 (-0.675)

Table 3.1: Simulation results. Normalized MSE with respect to the true variance, $\left(\sigma_n^2 - \sigma_0^2\right)^2 / \sigma_0^2$. Normalized bias with respect to the true variance is in parentheses.

Boxplots in figures 3.1 and 3.2 clearly illustrate the performance differences between the SP and the blocked sub/resampling estimators. Subsampling with $b \geq 10$ was the best of the latter group, with normalized MSEs on average 2/3 larger than those of the SP 'kitchen sink' sample-based estimator. The corresponding CBB estimator had nearly equal performance. Enforcing the block size to be at least 10 did not seem to affect either very much. The corresponding SBB estimators performed substantially less well, with normalized MSEs on average 22% larger than those of subsampling and a little more than twice that of the SP 'kitchen sink' sample-based estimator. Figure 3.3 examines block sizes as a function of time series type, sample size and normalized bias. Most b were less than 20, but the range of b was substantial, from 2 to 87. Not surprisingly, large block sizes tended to be associated with negative bias overall. Average b increased with increasing $\max\{\boldsymbol{\tau}_0\}$, a reassuring sign that the algorithm we used to select b was performing reasonably.

The remaining estimators performed substantially less well. As expected, the estimator ignoring dependence did poorly and significantly underestimated the variance. The $\boldsymbol{\tau}_{max}$ estimators also did not fair well. They were not defined in a way that prevented them from giving negative variance estimates, which they did on rare occasions at the smallest sample size of $n = 250$. They improved substantially as sample size increased, and at $n = 1000$, had slightly better average normalized MSEs than the best performing SBB estimator. The CBB and SBB estimators with block size $b = 0.1n$ had very poor performance, a sobering reminder of the sensitivity of these algorithms to block size selection. Clearly, one should not resort to ad hoc methods for selecting b.

Table 3.2 lists the coverage of the confidence intervals using each of the estimators of the variance of the sample mean, overall and by sample size. With the exception of SBB with $b = 0.1n$, all estimators that assume some form of dependence had adequate coverage. However, SP estimators did have better coverage overall than the sub/resampling estimators, especially at smaller sample sizes. That SS and CBB had reasonable coverage suggests the algorithm we used to estimate b worked as expected.

3.6 Practical Data Analysis: Average Treatment Effect for a Time Series

Environmental Health applications are an area in which SP variance estimation has the potential to be very useful. Not only are exposures and outcomes frequently dependent in time and/or space, but environmental health studies often inform public policy. Reliable, theoretically sound standard error estimates are crucial in this setting.

In this section, we examine the relationship between ventilation rate (VR, the rate at which outdoor air is brought indoors) and illness absence for a single elementary school classroom followed over two years. These data are a subset of a much larger data set consisting of 162 classrooms in three California school districts. In each room, daily VRs (in units of liters per second per person, L/s/p) were estimated using measurements transmitted every five minutes from Ethernet-wired sensors. Daily counts of the number

estimator	overall	$n = 250$	$n = 500$	$n = 750$	$n = 1000$
SP model-based (sample)	0.946	0.942	0.946	0.947	0.948
SP constant $\boldsymbol{\tau}$ (IC)	0.944	0.939	0.943	0.945	0.946
oracle, IF-based	0.943	0.938	0.944	0.946	0.947
SP 'kitchen sink' (sample)	0.943	0.940	0.943	0.944	0.945
SP model-based (mode)	0.942	0.938	0.942	0.944	0.945
SP 'kitchen sink' (mode)	0.936	0.931	0.936	0.938	0.940
$\boldsymbol{\tau}_{max}$, IF-based	0.932	0.914	0.932	0.939	0.942
SS, $b \geq 10$	0.929	0.918	0.928	0.932	0.936
SS, data-adaptive b	0.928	0.917	0.928	0.932	0.935
CBB, $b \geq 10$	0.928	0.918	0.927	0.932	0.935
CBB, data-adaptive b	0.927	0.915	0.927	0.931	0.935
SBB, $b \geq 10$	0.923	0.909	0.923	0.928	0.931
SBB, data-adaptive b	0.921	0.906	0.921	0.927	0.930
CBB, $b = 0.1n$	0.913	0.908	0.912	0.914	0.916
SBB, $b = 0.1n$	0.896	0.890	0.896	0.897	0.900

Table 3.2: Coverage probabilities

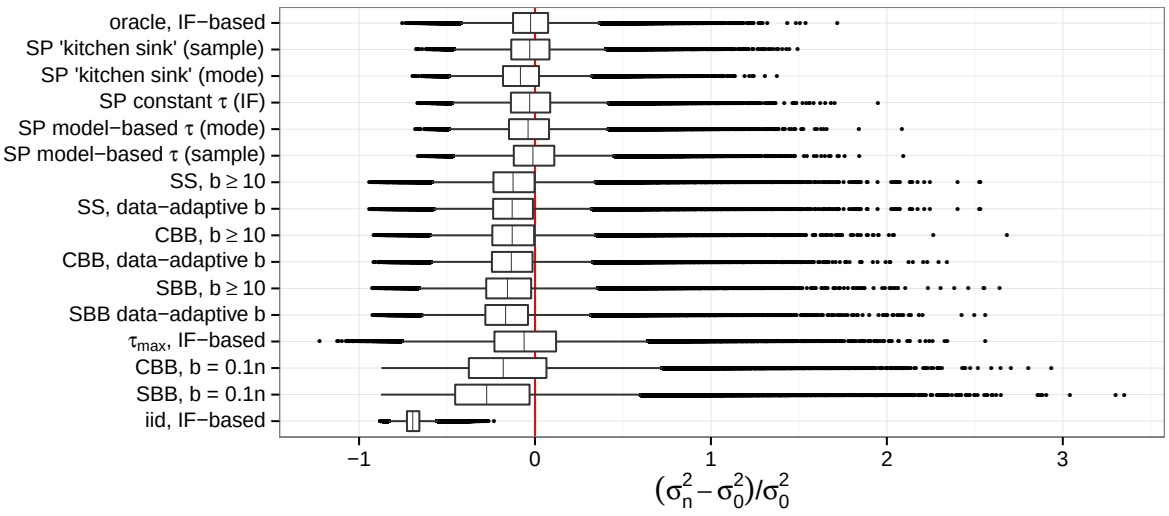


Figure 3.1: Boxplot of overall standardized bias for a subset of estimators. Boxplots are ordered vertically (top is best, bottom is worst) according to average normalized MSE.

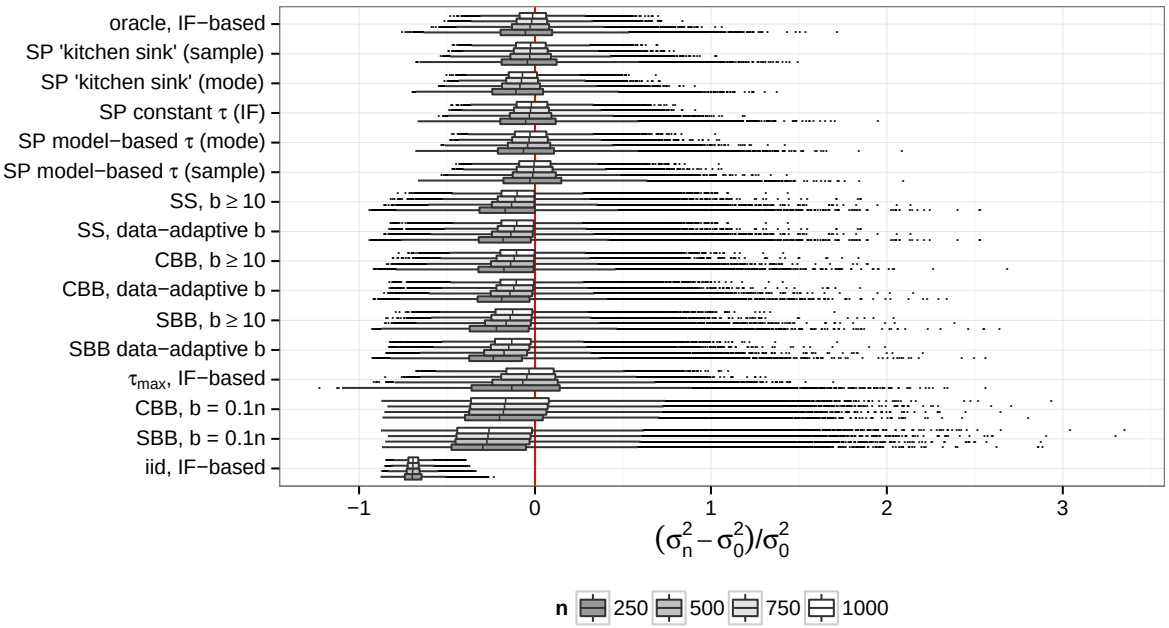


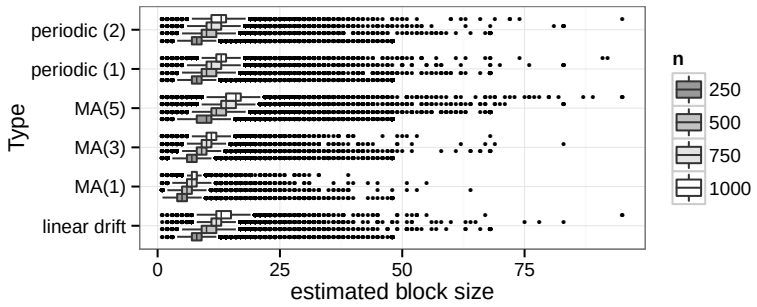
Figure 3.2: Boxplot of standardized bias by sample size for a subset of estimators. Boxplots are ordered vertically (top is best, bottom is worst) according average normalized MSE.

of students per classroom absent due to illness were also collected, along with other classroom level demographic covariates. These are the largest, most detailed school-based VR data in the scientific literature to date. We refer the reader to Mendell et al. [2013] for technical details regarding the sensors used in this project, how the VRs were estimated, and how the schools were sampled for inclusion in the original study.

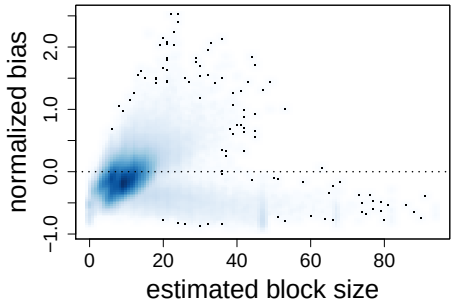
3.6.1 School Ventilation Rates: Public Health Significance

According to the Environmental Protection Agency, the average American (including children) spends about 90% of her time indoors [Klepeis et al., 1996]. Indoor air can contain many irritants and pollutants that adversely affect human health. These can include chemical emissions from the building and its various components (paint, carpet, cleaning product residue, etc.) as well as potentially infectious emissions from the room's occupants. Proper ventilation decreases the indoor concentrations of these potentially harmful substances. Scientists have found associations between lower VRs and adverse human health outcomes (see for instance Li et al. [2007]; Seppänen et al. [1999]). School-age children spend more time in school than in any other indoor environment besides home [Klepeis et al., 1996]. As of 2010, in the state of California alone there were approximately 6,224,000 students in 303,400 K-12 classrooms [California Department of Education, 2012], all spending on average six to seven hours per day inside classrooms.

Given that a significant percentage of the general population is habitually exposed



(a) Boxplot of estimated subsample size by time series simulation type.



(b) Smoothed scatterplot of normalized bias as a function of estimated subsample size.

Figure 3.3: Diagnostic plots of subsample size for SS estimator with b estimated data-adaptively.

to the indoor environment of schools and that we know there is a connection between insufficient ventilation and human health, one might assume the current VR guidelines are informed by scientific studies involving human health outcomes. Surprisingly, this is not the case. Historically, VR guidelines were designed to improve the perceived odor of a room [Mendell and Fisk, 2014]. While there have been efforts over the years to incorporate health concerns into recommended guidelines, these efforts have been relatively ad hoc in nature. The larger goal of the study from which these data are drawn was to provide a more rigorous scientific basis for VR threshold guidelines in school classrooms [Mendell et al., 2013].

3.6.2 Observed Data and Target Parameter

In this subsection, we first familiarize the reader with the observed data. We then introduce the target parameter and its estimator within the context of i.i.d. observations. Next, we outline the assumptions we are willing to make that permit us to use this version of the estimator, even though our data are clearly not i.i.d. Here we highlight the challenges inherent to any estimation problem in this subject matter area. We discuss the limitations of using this estimator, but contend that the implied statistical parameter is still interesting. However, we would like a robust standard error estimate. We use SP variance estimation to do this.

Our outcome of interest, $Y(t)$, is the total number of children absent specifically due to illness on day t. Figure 3.4 shows that while illness absence was not a rare event, the majority of the days in our study had no illness absences. The maximum number of illness absences we observed was four.

Let $A(t)$ be a binary indicator equal to one when the average estimated VR taken over the seven calendar days prior to time t is at least 7.1 L/s/p (the current recommended threshold for classrooms, ASH [2010]), and zero otherwise. This is our exposure of interest. VRs on minimum days were excluded from analyses, when they were less likely to reach equilibrium. The classroom in analysis had at least one minimum day per week. Days with CO_2 values considered implausible for a typical occupied room (below 600 or above 7000 ppm) were also excluded. The majority of VRs included in this analysis (78%) did not meet the recommended threshold. A histogram of VRs included in our analysis (figure 3.4) shows the distribution of VRs varied substantively, with minimum and maximum values of 2.44 and 25.76 L/s/p, respectively. One convenient statistical property of VR in this study (and thus of our exposure of interest) is that daily VRs are independent of past illness absence rates and can be considered conditionally independent of past VRs as well, once one controls for season. The choice of this exposure metric is guided by a sensible evaluation of what little is known with respect to lag times from exposure to disease development for diseases most associated with illness absence in schools. We direct readers to the original paper for a more full discussion. Figure 3.4 plots $Y(t)$ as a function of seven-day average past VR, where it appear lower exposure values are associated with more non-zero $Y(t)$.

Let $W(t)$ denote a vector of real-valued covariates measured before exposure. In this study, $W(t)$ consisted of two variables: an indicator of winter season, and classroom enrollment count. Season is an important potential confounder (common cause of both exposure $A(t)$ and outcome $Y(t)$) in this study. The observed classroom is naturally ventilated with no air conditioning, thus the primary means by which outside air is brought indoors is via windows, which are more likely to be closed when it is cold outside. We would therefore expect VRs to be lower in winter. There might also be a higher baseline illness rate during winter months, regardless of VR threshold attainment. We control for this potential confounding effect by including an indicator variable equal to one when t is a winter day, and zero otherwise. The total number of students present each day during the exposure window is another important factor, as this represents the number of children at risk of being absent in the near future who were actually exposed. Unfortunately, we were unable to obtain this information. Because illness absence counts were typically quite low (about 64% of observed days had no illness absences, with the majority of nonzero counts less than 3, see figure 3.4 for a histogram), we felt a reasonable proxy for the total number at risk was the average number of children enrolled in the previous seven calendar days. Classroom enrollment ranged between 22 and 26 students.

3.6.3 Estimating an Average Treatment Effect Using TMLE

We would like to learn about the effect of the current recommended VR threshold on classroom illness absence. Specifically, we want to estimate the magnitude of the difference in daily illness absence counts when the average VR over the previous seven calendar days always meets or exceeds the current guidelines of 7.1 L/s/p versus when it never meets 7.1 L/s/p. To do this, we estimate an average treatment effect (ATE). The ATE was originally defined within the potential outcomes framework introduced by Rubin (1974) as a missing data problem, but can also be defined using nonparametric structural equation modeling (Pearl, 2010) in terms of interventions on a structural causal model. We use Targeted Maximum Likelihood (TML, van der Laan and Rubin [2006], Gruber and van der Laan [2010], Gruber and van der Laan [2012]) to estimate the ATE and an ensemble machine learning algorithm called Super Learner [Polley and van der Laan, 2012] to estimate the relevant portions of the likelihood.

TML and Super Learner are both components of a general approach called Targeted Learning [Rose and van der Laan, 2011b]. Super Learner is a generalization of the stacking algorithm first introduced by Wolpert [1992] within the context of neural networks and later adapted by Breiman [1996] to the context of variable subset regression. It works by combining predictions from a diverse set of competing prediction algorithms using cross-validation, thus eliminating the need to pick a single prediction algorithm a priori. Theory guarantees Super Learner will perform asymptotically at least as well as the best algorithm in the competing set [van der Vaart et al., 2006, van der Laan et al., 2007]. TML is a procedure for estimating parameters of semiparametric models. They are loss-based defined substitution estimators that work by updating initial estimates in a bias-reduction step targeted toward the parameter of interest instead of the overall density. Provided specific assumptions about the data generating process are met, TML estimators are efficient and unbiased.

We briefly summarize the formal definition of the ATE within the context of TML and i.i.d. data, as this is the version of the estimator we will use. Let $O_i = (O_1, ..., O_n)$ be a data set consisting of n i.i.d. draws from the random variable $O = \{W,\ A,\ Y\}$. O has true probability distribution P_0, contained in the statistical model $\mathcal{M}$. W is one or more covariates, A is a binary treatment or exposure, and Y is a bounded, real-valued outcome. The ATE is defined as the marginal difference in Y if A is set deterministically to 1 versus if A is set deterministically to 0. Under the assumptions of no unmeasured confounding and positivity, the ATE can be written in terms of the observed data distribution P_0 as

$$\psi_0^{ATE} = \mathbb{E}_W\big\{\mathbb{E}_0[Y|A=1,W] - \mathbb{E}_0[Y|A=0,W]\big\}. \tag{3.5}$$

Let $\overline{Q}_0(A,W) \equiv \mathbb{E}_0[Y|A,W]$, and let $g_0(A,W)$ be the true probability of A given W. The influence function for (3.5) is

$$\begin{aligned} D(P_0) = &\left\{\frac{I(A=1)}{g_0(A=1,W)} - \frac{I(A=0)}{g_0(A=0,W)}\right\}\big\{Y - \overline{Q}(A,W)\big\} + \overline{Q}(A=1,W) \\ &- \overline{Q}(A=0,W) - \psi_0^{ATE}. \end{aligned} \tag{3.6}$$

A TML estimator $\Psi(P_n^*)$ of (3.5) solves the efficient influence function (i.e. the canonical gradient) of (3.5), $P_n D(P_n^*) = 0$, which is (4.3) applied to the empirical distribution that has been updated so as to make the optimal bias variance trade-off, denoted P_n^*. The variance of this efficient influence function gives the asymptotic variance of the TML estimator of (3.5).

Of course, we do not have i.i.d. data. Rather, we observe a discontinuous time series $O^n = \{O(t) : t\}$ of n observations over a particular time span, with true joint data generating distribution P_0^n. Van der Laan (2014) derived identifiability results and TML estimators for target parameters like the ATE when the data are observed over time on an interconnected network. While these estimators represent a significant advance in semiparametric causal inference for dependent data, they do require one to have some specific knowledge of the underlying dependence structure and are thus difficult to use in this context. Nevertheless, we believe the dependence in our data is weak; the statistical parameter formulated with respect to an i.i.d. data-generating process is interesting, and an estimate of it will provide scientists with useful information. Below, we summarize what we know about the data generating process we observe, what we are comfortable assuming, and what we are willing to assume for the sake of moving forward.

We believe O^n has a martingale-type dependence structure. Specifically, let $\overline{O}(t) = \{O(t), O(t-1), ..., O(1)\}$, and in an abuse of notation, let $\overline{O}(t-\tau_t) = \{O(t), O(t-1), ..., O(t-\tau_t)\}$ for some t-specific positive integer τ_t that is substantially less than both the sample size n and the total time span covered by the sample. We are willing to assume P_0^n can be factorized as

$$P_0^n = \prod_{t \in T} P_0\{O(t)|\overline{O}(t)\} = \prod_{t \in T} P_0\{O(t)|\overline{O}(t-\tau_t)\},$$

where P_0 is common across all t. We do not know the true dependence lags $(\tau_t : t)$. We feel quite comfortable assuming the time dependence in our likelihood does not come from the distribution of the covariates $W(t)$ or the exposure mechanism $g_0\{A(t), W(t)\}$, for reasons stated above. We believe the randomization assumption holds, since daily VR is not in any way related to past illness absence counts. The problematic relevant portion of the likelihood is $\overline{Q}_0$. Specifically, in order for the i.i.d.-formulated TMLE to be well defined, we must be willing to assume $\overline{Q}_0$ is common across time (i.e. it is not indexed by t); and $\mathbb{E}_0[Y(t)|\overline{A}(t-\tau_t), \overline{W}(t-\tau_t)] = \mathbb{E}_0[Y(t)|A(t), W(t)]$. (Since $W(t)$ varies quite slowly over time, for all practical purposes we can treat it as equivalent to $\overline{W}(t-\tau_t)$.) While we may feel confident that the large majority of $Y(t)$ only depend on VRs captured within $A(t)$, it is not impossible for $Y(t)$ to be a function of VRs in the more distant past for some t. In addition, some $Y(t)$ may be dependent on illness absence counts in the recent past. We have no way of knowing the true dependence lag in either case. However, as we stated above, we still believe an ATE estimator that makes the necessary assumptions about $\overline{Q}_0$ will produce an interesting, useful result, even if we cannot be sure a causal interpretation is appropriate.

The library of learners used in our ATE TML estimation procedure included generalized linear models (with and without interactions); a step-wise GLM; Generalized Additive Models using cubic smoothing splines, parameterized by equivalent degrees of freedom (2, 4 or 6) [Hastie, 1991]; and multivariate adaptive polynomial spline regression [Stone et al., 1997]. To achieve a uniformly bounded loss function, predicted values for all algorithms were truncated to the range of the observed data. Note that these data are not amenable to traditional time series approaches because of their inherently discontinuous nature.

3.6.4 Estimating a Standard Error

Because these results could be used to inform future policy, we feel it is important to obtain a robust standard error estimate. We use SP variance estimation to do this. We assume the influence functions in our analysis are dependent in time in a way we don't fully understand. We are comfortable assuming this dependence is bounded in time and that all true covariances are positive. We do not know enough about the underlying dependence to use a model-based $\boldsymbol{\tau}$ approach. The SP approach assuming constant $\boldsymbol{\tau}$ is possible, but less convenient given the temporal discontinuities in these data. We therefore implement the 'kitchen sink' SP variance estimator, using an upper bound on the temporal extent of the dependence of any two influence function values of $\max(\boldsymbol{\tau}) = 21$. This threshold encompasses the vast majority of incubation periods for infectious respiratory diseases [Lessler et al., 2009]. Our 'kitchen sink' sieve consisted of the following sequences of $\boldsymbol{\tau}$ vectors.

1. Winter step functions. Time points in winter may have a different dependence lags than those in nonwinter: $\boldsymbol{\tau}(t) = \alpha_1 I(t \in \text{winter}) + \alpha_2 I(t \notin \text{winter})$, where $(\alpha_1, \alpha_2) \in \{0, 1, \ldots, 21\}^2$, $\alpha_1 \neq \alpha_2$.

2. Seasonally periodic functions. Periodic $\boldsymbol{\tau}$ vectors were generated so that periodicity corresponded with winter season.

3. $\boldsymbol{\tau}(t)$ depends on $Y(t)$. Time points with more extreme average illness absence may be dependent on more past time points than those with less extreme average illness absence (or vice versa).

4. $\boldsymbol{\tau}(t)$ depends on past VR. Days with more extreme VRs may have had more lasting influence on future $Y(t)$ than days with less extreme VRs.

For demonstration purposes, we implemented nine types of SP 'kitchen sink' variance estimators. We used mode, step and sample-based versions, and we ordered by L_1 fit, complexity (defined using equation (3.3)), and the number of nonzero D_{ij} pairs included in the estimator.

Subsampling and blocked bootstraps are not well suited to this estimation problem for a number of reasons. The computation time required to repeatedly re-estimate the ATE would be considerable. Also, estimating an optimal b would be complicated by the inherently discontinuous nature of the data. Furthermore, one of our confounders,

winter, occurs in two large contiguous blocks, one for each year. In order to avoid positivity violations, b would have to be larger than the length of winter. We have no way of knowing whether or not an optimal b this large even exists. Using SP variance estimation is much more convenient and computationally efficient in this context.

3.6.5 Results

Our TML estimate of the ATE is -0.186. This means that on average, illness absence counts when the average VR over the preceeding seven calendar days at least met the current recommended threshold were 0.186 less than illness absence counts when the preceeding VR failed to meet the recommended threshold of 7.1 L/s/p. Given that the mean illness absence count is 0.48, this is a potentially consequential finding, and may suggest increasing the average VR in this classroom to at least 7.1 L/s/p could substantively reduce illness absence rates. The 0.95 confidence interval ignoring time dependence is $(-0.35, -0.02)$ (p-value = 0.03). The naive practitioner might be tempted to assume the results are both practically and statistically significant. However, our 'kitchen sink' SP estimated standard errors are much larger, providing us with a more realistic sense of the uncertainty of our ATE estimate. Table 3.3 lists variance estimates, p-values and confidence intervals for each of the SP variance estimators. The variance estimates are all more than three times as large as the i.i.d. IF-based estimate and produce confidence intervals containing zero, and p-values that are no longer significant. They are also in relative agreement with one another. Figure 3.5 provide a visual illustration of the estimation procedures, where we can clearly see a plateau in each case.

ordering	type	σ_n^2	0.95 CI	p-value
(none)	i.i.d. IF-based	1.592	(-0.351, -0.021)	0.027
number of nonzeros	mode	5.760	(-0.500, 0.128)	0.245
	value of longest plateau	5.774	(-0.500, 0.128)	0.246
	sample value at longest plateau	5.905	(-0.503, 0.132)	0.251
L_1 fit	mode	5.798	(-0.501, 0.129)	0.247
	value of longest plateau	5.825	(-0.501, 0.129)	0.248
	sample value at longest plateau	6.003	(-0.506, 0.134)	0.255
complexity	mode	5.752	(-0.499, 0.127)	0.245
	value of longest plateau	5.763	(-0.500, 0.128)	0.245
	sample value at longest plateau	5.992	(-0.506, 0.134)	0.254

Table 3.3: ATE variance estimation. Results ignoring dependence, and SP estimators, ordering by number of non-zero elements in the estimator; L_1 fit; and complexity. All SP estimators are of the 'kitchen sink' variety, utilizing $12,956$ unique dependence lag vectors.

Overall, the practical significance of our ATE estimate is mixed. Its magnitude suggests that meeting the threshold may have a positive effect on illness absence

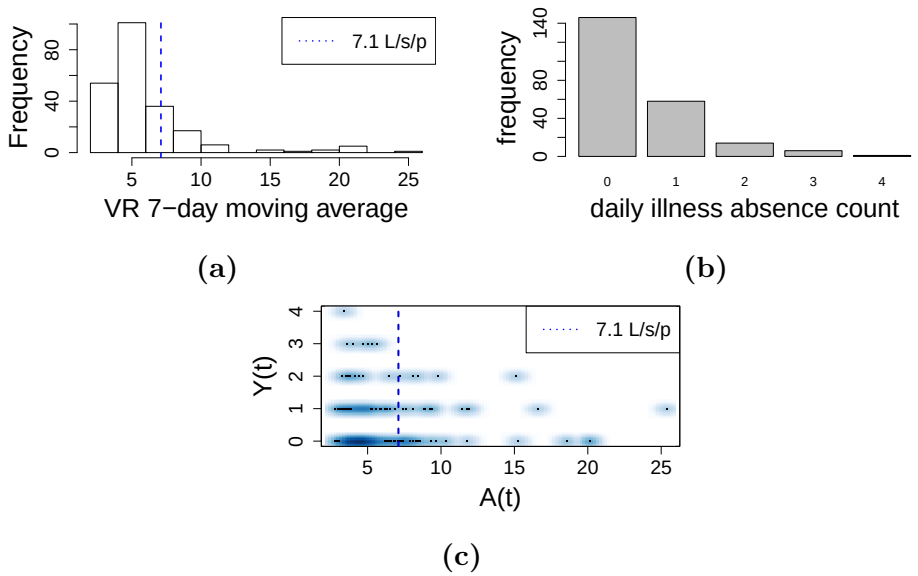


Figure 3.4: Descriptive plots of (a) 7-day average VRs (L/s/p), (b) daily illness absence counts, and (c) a scatterplot of daily illness absence as a function of prior 7-day average VR.

rates, but our more realistic SP-based standard error estimates should give us pause. Furthermore, collapsing a bounded, continuous exposure into a binary one can make interpretation of results somewhat challenging. A curve examining a range of potential thresholds may be a more informative target parameter. This will require using the full data, which include multiple schools, each with multiple, possibly interdependent classrooms, and is future work.

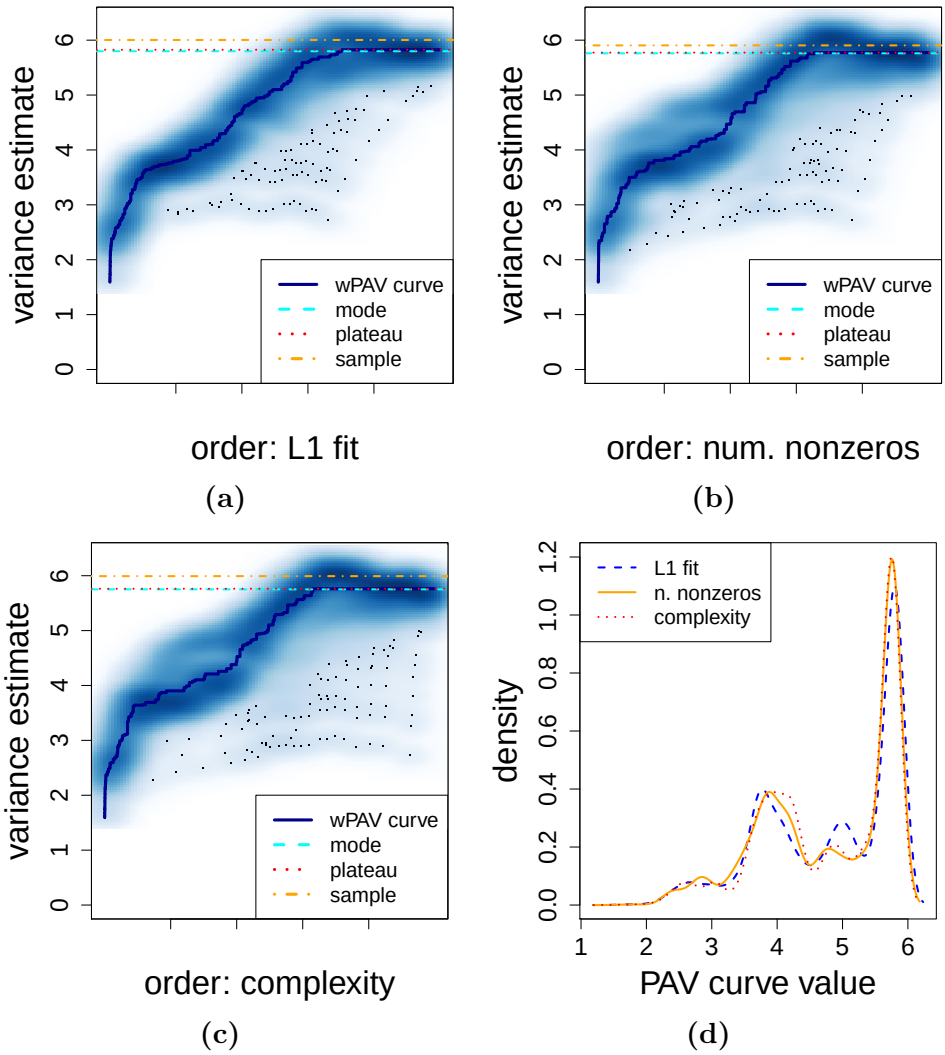
3.7 Discussion and Future Directions

Our original goal in our simulation study was to verify our proposed SP variance estimators could do at least as well as subsampling and blocked bootstrap estimators in a setting where these existing estimators have been well-studied. The chief advantage of the SP approach would then be its general utility: it can be used in situations where sub/resampling are ill-suited. Our simulations showed that in the familiar setting of sample means of time series, all three SP approaches substantively outperformed their competitors, both in bias and variance.

We also presented a practical data analysis, estimating an ATE for the effect of VR threshold attainment on subsequent illness absence counts in an elementary school classroom. This is an important public health issue, and a good example of a setting where sub/resampling would be difficult to use. We showed that in this case, properly accounting for time dependence gave larger, likely more realistic standard error estimates.

SP variance estimators do rely on more assumptions than subsampling, many of which are not testable, and bootstraps can, in some circumstances, capture second

Figure 3.5: Visualizations of SP variance estimation approaches when ordering by (a) L_1fit, (b) complexity, and (c) the number of nonzero D_{ij} pairs included in the estimator. (d) shows the estimated densities of each PAV curve.



order features that influence function-based methods will miss [DasGupta, 2008]. However, if one feels confident that the assumptions we require are reasonable, our approach appears to be an excellent alternative to blocked resampling and subsampling strategies. It avoids the challenge of estimating the nuisance parameter b and can result in significant computational savings.

Another important advantage of SP variance estimation is that it can be implemented even when one has limited understanding of the ordering of one's data. Dependence can have much more complex structure than what we have presented here. For example, spatiotemporal dependence is common in ecology and environmental health, and applications in internet advertising and infectious disease epidemiology often involve dependent observations on a poorly understood network. The SP 'kitchen sink' estimator could be particularly useful in these settings.

A limitation of our approach as it is implemented here is that we require monotonic dependence. While monotonic dependence is common, there are applications where this requirement is not met. Health care settings with finite resources, for instance, or plant species distributions where organisms have an inhibitory effect on their immediate surroundings are some examples. Extending the SP variance estimation algorithm to settings with non-monotonic covariance is an important area for future work.

Chapter 4

Small Increases in Classroom Ventilation Rates May Substantially Reduce Illness Absence: Evidence From a Prospective Study in California Elementary Schools

4.1 Introduction

Indoor air can contain substances known to be deleterious to human health. Such substances include harmful chemical emissions from buildings and their contents as well as potentially infectious emissions from buildings' occupants. Ventilation (bringing outdoor air indoors) is known to decrease the indoor concentration of these potentially noxious substances. A number of studies have found associations between excessively low ventilation rates (VRs) and poor human health outcomes (see for example Seppänen et al. [1999], Li et al. [2007], and Simons et al. [2010]). Determining minimum VR rates consistent with human health, for specification in guidelines or legal standards, has been a long-standing scientific challenge. Mendell and Fisk [2014] and Persily [2015] both provide comprehensive historical overviews of the limited science reflected in previous recommended VRs. The first official VR guidelines were actually not health-related at all. Rather, they were measures intended to control the subjective experience of indoor odors. Current VR guidelines, while intended to minimize the energy costs associated with ventilation, have still made only minimal use of information about acceptable levels of indoor exposures to common toxins and pollutants, or acceptable risks of deleterious health effects. The scientific community needs more formal criteria for so-called evidence based VR guidelines [Mendell and Fisk, 2014], but more relevant evidence from human health studies is needed to support these criteria. Particularly little such evidence is available for schools, despite the importance of these environments. School-aged children spend more time indoors in classrooms than in any other indoor environment except home [Klepeis et al., 1996]. They also comprise a significant proportion of the general population. Bauman and Davis [2013] estimate that in 2010, there were roughly 48 million kindergarten through 12^{th} grade students enrolled in public schools alone in the United States. The goal of this paper is to provide scientists and policy makers with evidence that will be useful for crafting

policy with respect to minimally sufficient VR guidelines in schools.

In this paper, we evaluate the effect of prior VR on subsequent illness absence at the classroom level. To do this, we use data collected over a period of two years from 59 classrooms in a single California school district. Our statistical approach differs from what is typically done in indoor air quality studies, in that our parameter of interest is not a coefficient in a parametric regression. Rather, we use an ensemble machine learning algorithm and a statistical technique called Targeted Maximum Likelihood (TML) to estimate what average illness absence would have been had the average VR taken over the previous seven calendar days failed to meet a particular threshold v, where v is one of 12 possible values ranging between 3 and 25 liters per second per person (L/s/p). These estimates, when ordered by v, can give us important information about the nature of the relationship between VR and illness absence. For instance, we may expect that as v increases, associated average illness absence will decrease, but we may observe a threshold v^* beyond which additional reductions in illness absence are not practically significant. If v^* were less than the current recommended guidelines, this might suggest that guidelines could be safely relaxed without undo negative effects on illness absence rates overall. On the other hand, if v^* is substantively larger than than the current recommended guidelines, policy makers might want to consider more stringent VR guidelines in the future.

These data are a subset of a larger data set obtained from 162 classrooms in three California school districts, and were previously analyzed using traditional parametric regression approaches in Mendell et al. [2013]. Using the full data set for the present analysis would have increased our sample size substantially. However, the three districts differed from one another in important ways that made them difficult to compare. Pooling them into a single analysis would have introduced a level of complexity that ultimately might have limited the usefulness of our results. For example, each district is in a distinct climatic zone. Each also had a distinct distribution of VR system types and VR rates. One district had only air-conditioned classrooms, all with very low average daily VRs, while the other two had some or no air-conditioned rooms, all with much more variable daily VRs. Using all three districts in our analysis would have required us to cope with potential latent regional effects that would have been very difficult to disentangle from the effect of ventilation type. The district we chose for this analysis had no air conditioned classrooms, and a mixture of mechanical and natural ventilation systems. It also had the greatest variety in VRs on any given day across classrooms, and contained the most observed classroom days.

This paper is organized as follows: Section 2 describes the observed data in more detail, and explains our choice of exposure metric. Section 3 defines our parameter of interest and the estimator we will use to learn this parameter. Section 4 highlights some of the statistical challenges we face in this analysis, including unknown time dependence and ambiguous clustering. In order to learn from these data, we must be willing to make some assumptions about this unknown dependence structure. We outline these assumptions in section 4.1. Because these results may influence future policy, we feel it is important to obtain robust estimates of the standard error. Section 5 describes how

we do this using a technique called Sieve Plateau (SP) variance estimation. We present our results in section 6 and conclude in section 7 with a discussion.

4.2 Observed Data and Target Parameter

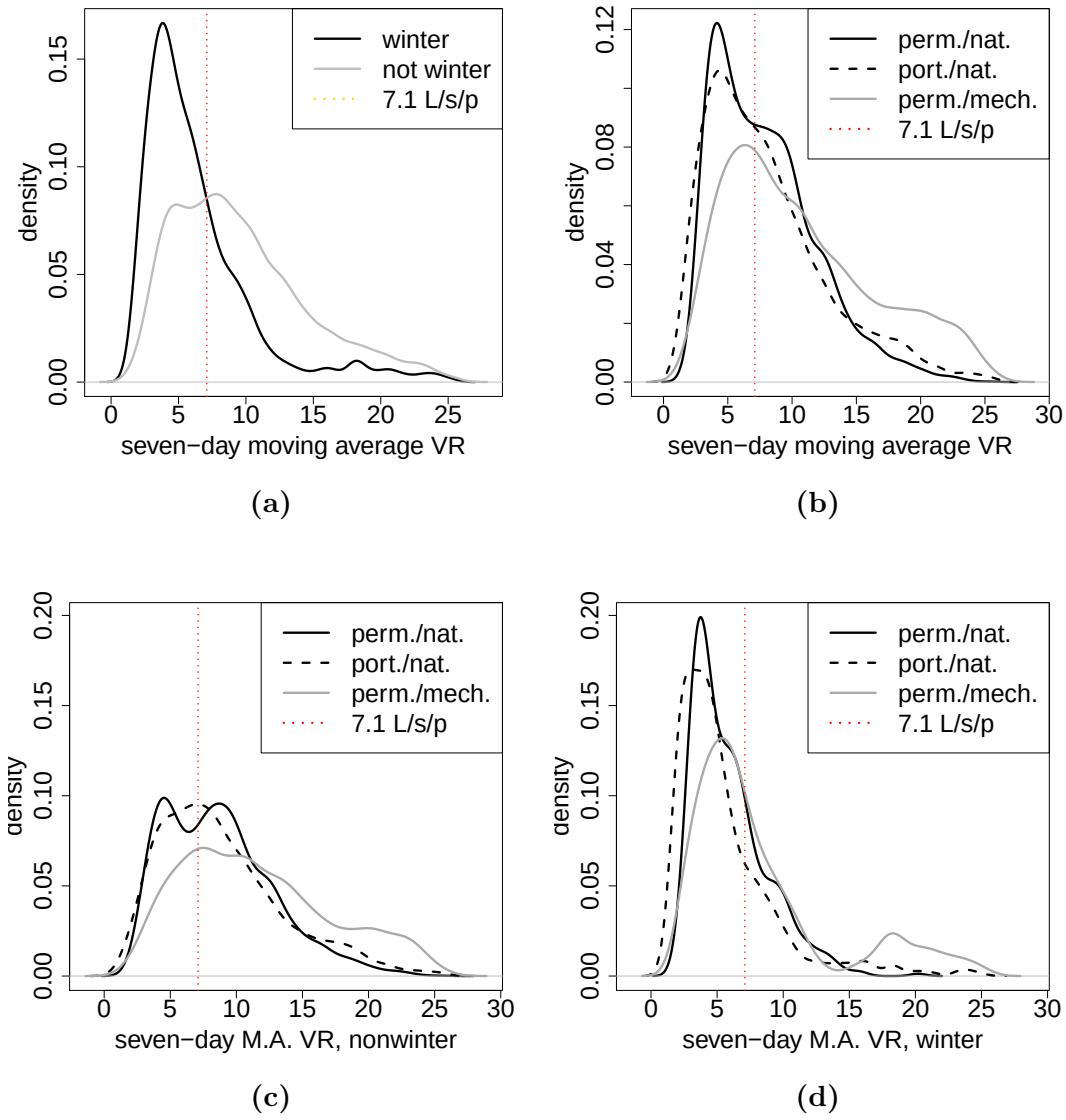
Fifty-nine third, fourth or fifth grade classrooms in 10 schools within the same California school district were followed for a period of up to two years, resulting in a total of $11,467$ observed classroom days. Each room was equipped with a sensor that measured temperature, CO_2 and relative humidity levels at five minute intervals in real time. These measurements were used to estimate daily VRs (in units of liters per second per person, L/s/p) for each classroom using a mass balance model. The mass balance model is a function of estimated indoor CO_2 equilibrium levels. Minimum days (school days less than five hours long) were excluded from our analyses, as these days were less likely to reach true equilibrium with respect to CO_2 values and therefore more likely to have VRs that were systematically biased upward. Days with average CO_2 values that were implausibly low or high for an occupied classroom were also excluded. For additional technical details about how the sensor data were cleaned and how the VRs were calculated, we direct the interested reader to Mendell et al. [2013]. These data are unusual in the scientific literature on school illness absence and indoor air quality because of their precision and temporal extent. Previous studies have tended to extrapolate average yearly ventilation rates from measurements taken over relatively short time periods. In this sense, Figure 4.1 is an interesting finding in and of itself. Many previous studies have estimated annual average VR from measurements collected over short time periods of several days. This study makes clear that in many cases, it may be inappropriate to assume that classroom VRs estimated in this manner are representative of the average VR in that classroom.

Daily classroom VRs have some convenient statistical properties. First, they are independent of past illness absence counts, both within the same classroom and across classrooms within the same school. Second, VRs in different classrooms within the same school can be viewed as conditionally independent, given season, building type and ventilation system type. We discuss these properties in more detail in section 4.

Our outcome of interest, which we denote $Y(t)$, is the total number of students absent due to illness in a single classroom on day t. Note that these are prevalence rather than incidence counts. Figure 4.2 shows bar charts of $Y(t)$ across all classrooms in this study. While the majority of classroom days (about 55%) had no reported illness absence, counts of one or two students were quite common (about 42%). Illness absence counts greater than two were rare. Roughly 4% of observed $Y(t)$ were equal to three, four or five, and only nine $Y(t)$ were six or greater. The maximum $Y(t)$ we observed was eight.

Let $\overline{V}(t)$ be the average VR over the seven calendar days prior to time t. Our exposure of interest is $A_v(t) = I\{\overline{V}(t) < v\}$, a binary indicator equal to one when $\overline{V}(t)$ in a single classroom failed to achieve a particular threshold value v, and zero otherwise. Table 4.1 lists the 12 threshold values we consider in this paper and the corresponding proportion of classroom days where $A_v(t)$ equaled one for each. In about

Figure 4.1: Density plots of $\overline{V}(t)$ for various subcategories.

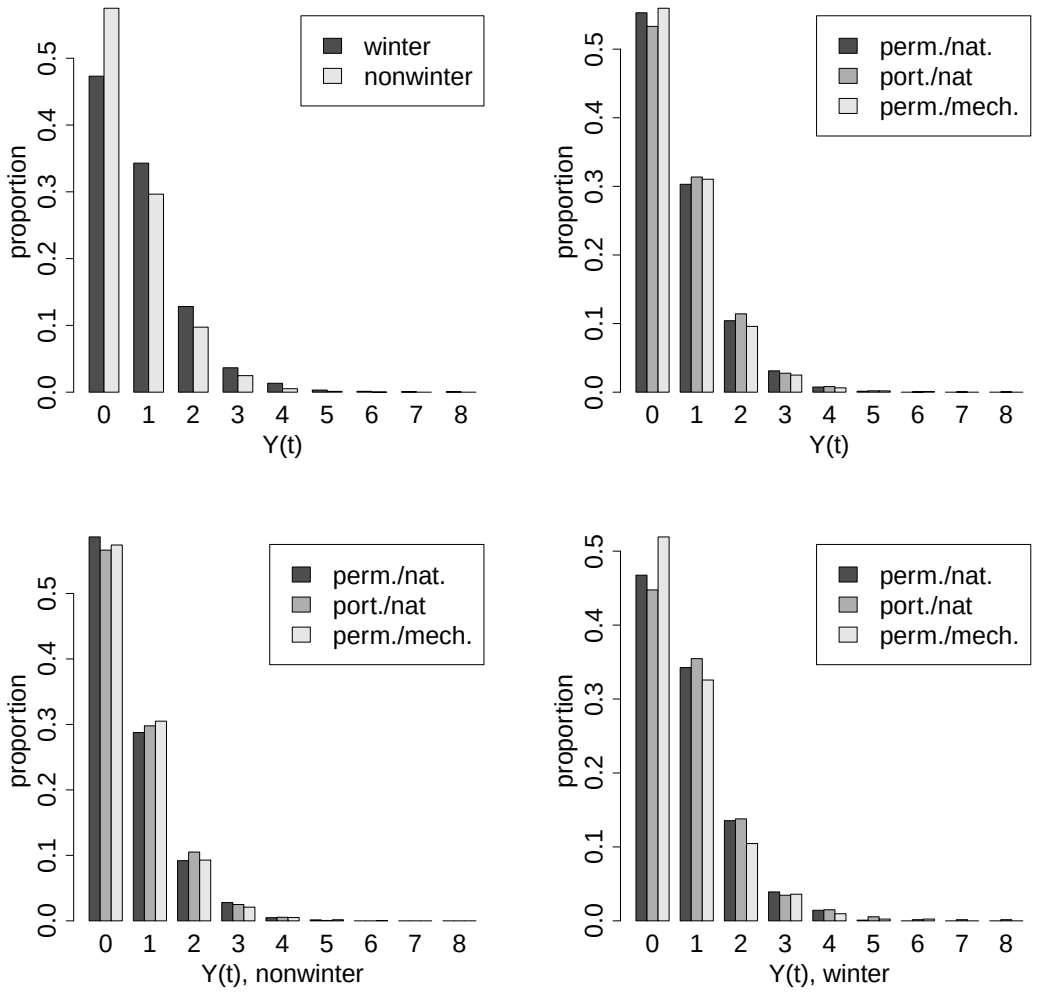


46% of classroom days we observed, $\overline{V}(t)$ failed to achieve the current recommended rate of 7.1 L/s/p. The minimum and maximum $\overline{V}(t)$ we observed were 1.31 L/s/p and 25.69 L/s/p, respectively. Our choice of a seven day moving average of past VR as an exposure metric is informed by what little is known in the scientific literature about disease prevalences in elementary schools and their associated lag times. We refer the reader to the original paper for a more full discussion.

Let $W(t)$ be a vector of the following real-valued covariates:

1. an indicator for building type, equal to one for classrooms in permanent buildings,

Figure 4.2: Bar plots of the proportion of daily classroom illness absence counts, $Y(t)$.



v (L/s/p)	3.41	4.64	5.87	7.10	8.33	9.56	10.79	12.02	13.25	14.48	15.71	25.69
prop. $A(t) = 1$	0.10	0.23	0.35	0.46	0.55	0.65	0.73	0.78	0.83	0.87	0.89	1.00

Table 4.1: Threshold values used to define $A_v(t)$, and the proportion of classroom days where $A_v(t) = 1$ for each of these values. Note that 7.1 L/s/p is the current standard for newly constructed schools in most building codes.

and zero for classrooms in portable buildings;

2. an indicator of ventilation type, equal to one for classrooms with mechanical ventilation, and zero for classrooms with only natural ventilation (no classrooms

in this study were air conditioned);

3. an indicator equal to one when t was a winter day and zero otherwise; and

4. classroom enrollment on day t.

Figure 4.3 shows mean $Y(t)$ and $\bar{V}(t)$ by building type, ventilation type, and winter season. Figure 4.1 shows density plots of VR for these categories.

There were 35 classrooms in this study in permanent buildings. Fourteen of them had mechanical ventilation ($3,170$ classroom days), 21 were naturally ventilated ($3,736$ classroom days). Twenty-four classroom were in portable buildings and naturally ventilated ($4,563$ classroom days). There were no mechanically ventilated classrooms in portable buildings in this study.

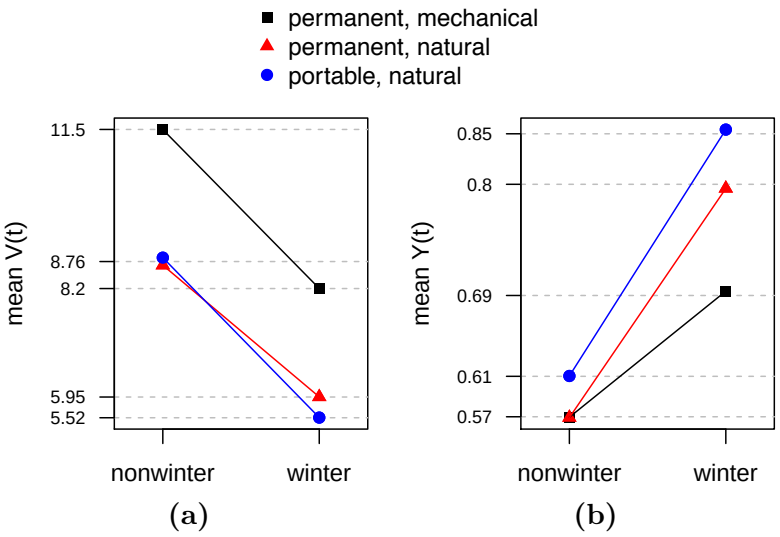
It is possible that building type could have influenced both VR and illness absence rates. Portable buildings may be more tightly sealed than permanent ones, and may also have a somewhat different composition of indoor pollutants than permanent buildings. In figure 4.3, we can see that naturally ventilated rooms had lower VRs than mechanically ventilated ones on average.

It is also possible that ventilation type could affect both $\bar{V}(t)$ and illness absence counts. All classrooms in this study had operable windows. Naturally ventilated classrooms rely entirely on open windows and doors for ventilation, whereas mechanically ventilated classrooms provide additional air through a mechanical ventilation system. We would therefore expect mechanically ventilated rooms to have higher VRs on average. It's also possible mechanical ventilation could be associated with some additional filtration. In the observed data, $\bar{V}(t)$ in mechanically ventilated rooms was larger on average than in naturally ventilated rooms, and illness absence counts somewhat smaller on average.

The winter season is an important potential confounder (common cause of both the exposure of interest and the outcome) which we control for in our estimation procedure [1]. The majority of classroom days in our study ($8,299$) were from naturally ventilated rooms. In naturally ventilated rooms, the primary means by which outdoor air is brought indoors is through windows. Windows are more likely to be closed when it is cold outside, which means VRs in these rooms are more likely to be low during winter. Figure 4.3 shows that this is indeed the case. Mechanically ventilated rooms also appear to have lower VRs on average during winter. This is not surprising, as all rooms in this study had operable windows and no air-conditioning. Cooler temperatures could decrease the likelihood that a mechanically ventilated room would have open windows, thus we would expect these rooms to have relatively lower VRs during the winter. There is also likely a higher baseline rate of illness absence during the winter season. Figure 4.3 shows that for each building/ventilation category, illness absence was on average higher in winter than in nonwinter seasons. Figure 4.3 also indicates that season and building type combined appear to affect illness absence: differences in average

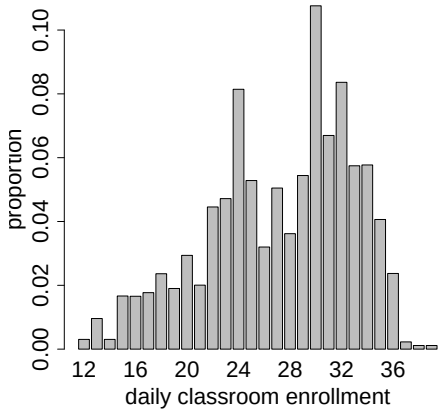
[1] Winter is defined in this study as any day between December 1 and March 1.

Figure 4.3: Mean seven day moving average VR, $\overline{V}(t)$, and daily illness absence count $Y(t)$ by various categories.



ventilation rates between building and ventilation types remains relatively constant across seasons; whereas differences in average illness absences are larger in winter, when infectious agents are presumably more numerous, than in nonwinter seasons. This highlights one of the central challenges of this estimation problem. It may be that under certain conditions, lower ventilation rate will only have an effect on illness absence in the presence of infectious agents. This is a limitation of almost any observational study undertaken in this subject matter area.

Figure 4.4: Barplot of daily classroom enrollment.



The number of students present on any given day is also an important factor for us to consider, as it represents the number of children at risk of being absent in the

future who were exposed to the VR on that day. Unfortunately, this is not a number we were able to obtain. We do, however, have access to classroom enrollment over time. Because illness absences greater than three were relatively rare, we feel this is a reasonable proxy for the number at risk. Figure 4.4 shows a bar plot of daily classroom enrollment, which ranged from a minimum of 12 students (35 classroom days) to a maximum of 40 students (13 classroom days). About half of all observed classroom days had enrollment counts of between 23 and 32 students. Enrollment count for any given classroom tended not to change too much within a single school year. About half of the 112 classroom years we observed had maximum enrollment count changes of between one and three students, and 24% had no change at all. About 7% of classroom years had maximum enrollment count changes of more than five students, with the largest enrollment change being 14 students.

One variable of potential concern is socioeconomic status (SES). In some districts, it may be that schools composed of students with low SES are more likely to have poorly maintained facilities, including ventilation systems, and therefore lower indoor air quality, than schools composed of students of higher SES. It could also be the case that low SES students could have systematically different illness absence mechanisms than students with higher SES. For instance, lower SES students might be less likely to be kept out of school when they are sick because of a lack of available childcare. In this particular case, however, we did not see such patterns. We used the proportion of enrolled students in each classroom who participated in the national free or reduced cost school lunch program (NSLP) as a proxy for SES. We found no systematic relationship between NSLP proportion and VR rates or illness absence counts at the classroom level in these data. This may be because a large percentage of the classrooms we studied are naturally ventilated, and therefore not influenced by poorly maintained mechanical ventilations systems. The climate where this district is located is also generally dry and warm, which could mean that buildings were less subject to mold, dampness and general decay that could be deleterious to student health. Because we found no evidence of a relationship between NSLP, VR and illness absence counts, we did not include SES in the present analysis.

4.3 Estimating the Target Parameter Using TMLE

Recall that we have defined our outcome $Y(t)$ as daily classroom illness absence count, and our exposure $A_v(t)$ as the indicator that the average VR over the seven calendar days prior to time point t failed to reach a particular threshold of v L/s/p. Our target parameter in the present analysis is the so-called *average outcome under treatment*, the expected value of $Y(t)$ if $A_v(t)$ is set deterministically to one for all t in all classrooms, which we denote $\psi_0(v)$. Note that we are only intervening on the prior seven calendar days of VR. This is not the same as deterministically setting all daily VRs to be less than v L/s/p. That could be an interesting target parameter, as it could potentially capture effects of chronic exposure to low VRs, but we do not have the support in the current data to do this.

We use Targeted Maximum Likelihood (TML) [van der Laan and Rubin, 2006,

Rose and van der Laan, 2011a, Gruber and van der Laan, 2012] to estimate $\psi_0(v)$, and an ensemble machine learning algorithm called Super Learner to estimate the relevant parts of the likelihood [van der Laan et al., 2007, Polley et al., 2011b, Polley and van der Laan, 2012].

Super Learner is a generalization of an ensemble prediction method first introduced by Wolpert [1992] within the context of neural networks and later adapted by Breiman [1996] for variable subset regression. It works by combining predictions from a diverse set of competing prediction algorithms in such a way as to minimize cross-validated risk, thus eliminating the need to pick a single prediction algorithm a priori. Theory guarantees Super Learner will perform asymptotically at least as well as the best algorithm in the competing set [van der Vaart et al., 2006, van der Laan et al., 2007]. In the present analysis, the library of competing prediction algorithms used by Super Learner included Generalized Linear Models (with and without interactions, using various subsets of covariates); Generalized Additive Models using cubic smoothing splines[Hastie, 1991]; and multivariate adaptive polynomial spline regression [Stone et al., 1997]. To achieve a uniformly bounded loss function, predicted values for all algorithms were truncated to the range of the observed data.

TML estimation is an approach to estimating parameters of semiparametric models and is applicable to any pathwise-differentiable parameter of a probability distribution. TML estimators are efficient and unbiased, given specific assumptions about the data generating distribution. They work by updating initial estimates in a bias-reduction step targeted toward the parameter of interest instead of the overall density [van der Laan and Rubin, 2006, Gruber and van der Laan, 2010], and can be used to estimate the statistical counterpart to a causal parameter.

We briefly summarize the formal definition of $\psi_0(v)$ within the context of TML and independent and identically distributed (i.i.d.) data, as this is the version of the estimator we will use.

Consider the random variable $O = \{W,\ A_v,\ Y\}$. O has true probability distribution P_0, contained in the semiparametric statistical model $\mathcal{M}$, the set of all possible probability distributions of O. W is a real-valued vector of covariates measured before treatment, A_v is a binary treatment or exposure, and Y is a bounded, real-valued outcome measured after treatment.

Define the parameter mapping $\Psi : \mathcal{M} \to \mathbb{R}$ that maps a probability distribution P, an arbitrary element of the statistical model $\mathcal{M}$, to $\mathbb{E}_P\{\mathbb{E}_P[Y|A_v = 1, W]\}$, where $\mathbb{E}_P$ is the expected value taken with respect to P. Assume positivity, i.e. $P_0(A_v = 1|W) > 0$ for almost all W. Our parameter of interest is this mapping applied to P_0,

$$\Psi(P_0) = \mathbb{E}_{P_0}\big\{\mathbb{E}_{P_0}[Y|A_v = 1, W]\big\}. \tag{4.1}$$

If we are willing to make additional causal assumptions, (4.1) be interpreted as the average outcome had all A_v been set deterministically to one. In particular, we would need to assume randomization (also sometimes referred to as the "no unmeasured confounders" assumption), i.e. the observed treatment A_v is independent of the counterfactual outcome had each observation received a treatment of 1, given W.

Let $\bar{Q}_0(a,w) \equiv \mathbb{E}_0[Y|A_v = a, W = w]$, the expectation of Y given $A_v = a$ and $W = w$, taken with respect to P_0. Let $Q_W(w) = P(W = w)$ be the distribution of the baseline covariates. Let $g_0(1|w) = P_0(A_v = 1|W = w)$ be the true probability of $A_v = 1$, conditional on baseline covariates $W = w$. This is often referred to as the propensity score. Since the parameter mapping Ψ depends only on $Q = (\bar{Q}, Q_W)$, we sometimes use $\Psi(P)$, $\Psi(Q)$ and $\Psi(\bar{Q}, Q_W)$ interchangeably.

Suppose we observe n i.i.d. draws from P_0, which we denote $O_1, ..., O_n$. Let P_n be the empirical distribution of $O_1, ..., O_n$. Let $\bar{Q}_n$ be an estimate of $\bar{Q}_0$, and $Q_{W,n}$ be the empirical distribution of W. We can compute the plug-in estimate of (4.1) as

$$\Psi(\bar{Q}_n, Q_{W,n}) = \int_w \bar{Q}_n(1,W) dQ_{W,n}(w) = \frac{1}{n}\sum_{i=1}^{n} \bar{Q}_n(1, W_i). \tag{4.2}$$

TML estimation is a framework for constructing plug-in estimators. A TML estimator first obtains an initial estimate of $\bar{Q}_0$, which we will denote $\bar{Q}_n^0$. Next, it obtains an estimate of $g_0(1|W)$, which we will denote g_n. Then, it updates $\bar{Q}_n^0$ to $\bar{Q}_n^*$ using $g_n(W)$, and computes the plug-in estimate (4.2) using $\bar{Q}_n^*$ and $Q_{W,n}$. The update estimate $\bar{Q}_n^*$ is constructed so that $\Psi(\bar{Q}_n^*, Q_{W,n})$ is less biased than $\Psi(\bar{Q}_n, Q_{W,n})$, and solves the efficient influence function estimating equation. The efficient influence function for the TML estimator $\Psi(\bar{Q}_n^*, Q_{W,n})$ at a $P \in \mathcal{M}$ is

$$\begin{aligned} D^*\{\bar{Q}, Q_W, g\}(O) = \frac{I(A_v = 1)}{g(A_v = 1|W)}\{Y - \bar{Q}(A_v, W)\} + \bar{Q}(A_v, W) \\ - \Psi(\bar{Q}, Q_W). \end{aligned} \tag{4.3}$$

We refer the interested reader to Rose and van der Laan [2011a] for a more detailed explanation of the updating step in the TML algorithm and for a derivation of the efficient influence function. Because influence functions figure prominently in our robust inference procedure, we explain them in greater detail below.

Asymptotically linear estimators can be written as

$$\sqrt{n}(\psi_n - \psi_0) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} D(P_0) + o_P(1),$$

where $D(P_0)$ is an influence function with mean zero at P_0; and $o_P(1)$ is a negligible remainder term that converges to zero in probability. An influence function is a type of functional derivative that describes how much an estimator changes under slight perturbations of the empirical distribution [Hampel, 1974]. Because influence functions are mean zero at the true parameter value, they can be used as an estimating equation.

All estimators of a given parameter have their own (not necessarily unique) influence function. Among all these influence functions, one has minimal variance. We call this influence function the *efficient* influence function. For an estimator to be efficient, it must be asymptotically linear with influence function equal to the efficient influence function [Bickel et al., 1993, Rose and van der Laan, 2011a].

For regular, asymptotically linear estimators, influence functions provide a linear approximation to that estimator. Thus with independent observations, the variance of the influence function can give us the asymptotic variance of that estimator, and the variance of (4.3) gives the asymptotic variance of the TML estimator of $\psi_0(v)$. One can use a plug-in estimate of (4.3) to obtain standard error estimates from which to compute confidence intervals and p-values.

4.4 Dependence in the Data

In reality, our observed data are not i.i.d. Rather, our observations comprise a clustered, discontinuous multiple time series. Dependence can occur here at several different levels: (1) within the same classroom across time; (2) between classrooms within the same grade across time; and (3) between classrooms within the same school across time. For the purposes of this analysis, we are comfortable treating schools as independent units. It is not unreasonable to assume there is some distance in time beyond which observations from the same classroom are at least conditionally independent of one another. The same assumption can be made about dependence between classrooms in the same grade or the same school. However, it's quite likely that the distance in time required for observations to be independent differs depending on whether observations are from the same school, the same grade, or the same classroom. In fact, the true dependence lags are probably complex functions of past within-classroom VRs, illness absence counts at all levels of the hierarchy, and incubation periods of infectious agents present during the exposure period. This difficulty of not knowing the true underlying dependence structure is endemic to any observational study of school illness absence, and is not something we can realistically learn from the observed data.

There are identifiability results and well-formulated TML estimators for $\psi_0(v)$ when the observed data are generated over time on an interconnected network of units [van der Laan, 2014]. These estimators are a substantial step forward in semiparametric causal inference for dependent data. However, they require specific knowledge of the underlying dependence structure and are thus difficult to use in this context. Still, we feel estimating $\{\psi_0(v), v \in V\}$ is a worthy undertaking, and that the statistical parameter formulated with respect to i.i.d. observations can still provide valuable information to policy makers. We believe the dependence in our data is weak enough that such estimators are unlikely to suffer from bias. That said, we believe obtaining 0.95 confidence intervals under independence assumptions would be inappropriate. For that reason, we will use alternative, more robust standard error estimation methods, which we describe below in section 5.

4.4.1 What we know about the data distribution, and what we are willing to assume

We assume our observed data have a clustered, martingale-like dependence structure. We observe n classrooms over a time period T, $O^n = \{O_i(t) : i = 1, ..., n;\ t \in T\}$. The distribution of these observed data is P_0^n, which is an element of a semiparametric statistical model $\mathcal{M}$. Let $O_i(t) = \{W_i(t), A_i(t), Y_i(t)\}$ be an observation from classroom

i on day t. Let $\Omega_i(t)$ be the set of observations upon which $O_i(t)$ depends. The key defining characteristics of $\Omega_i(t)$ are that it never contains observations from different schools; it never contains observations from future time points $k > t$; and the distance in time between $O_i(t)$ and any observation in $\Omega_i(t)$ is bounded by a classroom-day specific constant $\boldsymbol{\tau}_i(t)$. We believe $\boldsymbol{\tau}_i(t)$ is small compared to sample size and the total time span of T. Let $\overline{O}(t)$ be all observations in O^n that occurred on or before time t. Accordingly, we assume P_0^n can be factorized as

$$P_0^n = \prod_{i=1}^{n} \prod_{t \in T} P_0\{O_i(t)|\overline{O}(t)\} = \prod_{i=1}^{n} \prod_{t \in T} P_0\{O_i(t)|\Omega_i(t)\},$$

where P_0 is common across all i and t.

We do not believe the dependence in our observed data comes from the distribution of the covariates $\{W_i(t) : i = 1, ..., n; t \in T\}$. With respect to the exposure mechanism, the picture is more murky. $\overline{V}_i(t)$ are not affected by past illness absence counts, nor are they affected by past VRs in other classrooms within the same school. We are comfortable assuming VRs of classrooms within the same school independent from one another, conditional on season and building type. On the other hand, $\overline{V}_i(t)$ may, under certain conditions, have a tendency to be correlated over time within the same classroom. This correlation would most likely be due to structural aspects of the room (number and position of windows, for instance, or certain features of a mechanical ventilation system, if present) and how these features interacted with dynamic aspects of the outdoor environment, such as the direction of prevailing winds and outdoor temperature. However, we are comfortable assuming the randomization of exposure assignment holds, conditional on season and building type, as we do not believe violations of this assumption will induce bias in our estimator. It may affect the variance of our estimator, however. We address this point more fully in the next section.

We believe there may dependence in $\bar{Q}_0$. To use the i.i.d.-formulated TML estimator, we must be willing to assume the following:

1. $\bar{Q}_0$ is common across classroom days (i, t), and
2. $\mathbb{E}_0[Y_i(t)|\Omega_i(t)] = \mathbb{E}_0[Y_i(t)|A_{v,i}(t), W_i(t)]$.

Under these assumptions, our estimation problem would become an application of the network model presented in van der Laan [2014]. However, these assumptions clearly do not hold across every time point in all classrooms. While it's likely the majority of $Y_i(t)$ depend only on past VRs captured in $A_{v,i}(t)$, it is possible that a small number of $Y_i(t)$ are functions of more temporally distant VRs within the same classroom. $Y_i(t)$ could also occasionally be functions of recent VRs in other classrooms within the same school. (Perhaps the children participated in a joint activity in that room, for instance.) In addition, $Y_i(t)$ could be functions of recent past illness absence counts in the same classroom, or in different classrooms within the same school. This could happen through contagion processes, for instance.

Other estimators presented in van der Laan [2014] that are appropriate for dependent network structures require one to know $\{\Omega_i(t) : t \in T,\ i = 1, ..., n\}$. In this case, we have no way of knowing the true $\Omega_i(t)$ for each classroom day, nor can we hope to estimate them from the data. However, we believe any dependence in these data is sufficiently weak so that using the i.i.d. formulated TML estimator of $\psi_0(v)$ will still produce useful, scientifically interesting results, even if we cannot apply a rigorous causal interpretation.

4.5 Estimating Standard Errors When Dependence is Poorly Understood

Although we are willing to ignore dependence for the sake of making progress with our estimation problem, we are wary of relying on these same assumptions for our inference. Because these results could be used to inform new VR guidelines, we feel it is important to obtain robust standard error estimates. To do this, we use a technique called SP variance estimation [Davies and van der Laan, May 2014].

SP variance estimators serve the same purpose as blocked bootstraps and subsampling, in that they are a nonparametric way to get inference when the data are dependent in an unknown way [Lahiri, 2013, Politis et al., 1999]. However, blocked bootstraps and subsampling suffer from a significant drawback: they require the user to estimate an optimal subsample or block size. Unfortunately, both procedures are quite sensitive to this nuisance parameter, and obtaining a good estimate of it can be difficult [**?**]. In the present analysis, this task is made more complicated by the fact that our data are from a discontinuous clustered multiple time series where we would expect the true dependence lags at all levels of the dependence hierarchy to be unstable over time. In addition, one of our most important confounders, the winter season, occurs in two large contiguous blocks, one for each year. Using a block or subsample size that is less than the length of winter would lead to positivity violations, but we have no way of knowing whether or not an optimal block or subsample size this large exists. SP variance estimation avoids the difficulty of this nuisance parameter estimation step.

For a full theoretical and algorithmic description of SP variance estimation, we refer the interested reader to Davies and van der Laan [May 2014]. Below we provide an intuitive overview.

Recall that a TML estimator of $\psi_0(v)$, denoted $\psi_n(v)$, based on i.i.d. observations, is asymptotically linear, with specified influence function (4.3). Recall also that for asymptotically linear estimators, the scaled difference between our estimator and the truth converges to a normal limiting distribution with finite variance σ_0^2, i.e. $\sqrt{n}(\psi_n(v) - \psi_0(v)) \to^d N(0, \sigma_0^2)$. To estimate the standard error of $\psi_n(v)$, we need to estimate σ_0^2. When the data are independent, one could use the nonparametric delta method to do this by using the sample variance of the plug-in estimates of the influence function values,

$$\sigma_n^2 = \frac{1}{n} \sum_{i=1}^{n} D_i^2(P_n). \tag{4.4}$$

Consider the dependent data set-up described in section 4.1. Suppose we feel the dependence in our data is sufficiently weak so that we meet the assumptions outlined in Davies and van der Laan [May 2014], section 2. (Recall we have already assumed that any true covariances in our data are non-negative.) This means we still believe $\sqrt{n}(\psi_n - \psi_0)$ converges to a normal limiting distribution, but the proper estimator of the variance of that limiting distribution needs to include the non-zero covariances between dependent influence functions.

Suppose we knew the true dependence structure of our data. For notational convenience, let us index our observed data by classroom day, so $O^K = \{O_k : k = 1, ..., K\}$, where k is a particular classroom day. Denote the true set of observations upon which each observation O_k depends as $\Omega_{0,k}$. Given our assumptions, we could estimate σ_0^2 in a similar spirit to (4.4), using

$$\sigma_n^2 = \frac{1}{K} \sum_{k=1}^{n} \sum_{j \in \Omega_{0,k}} D_k(P_n) D_j(P_n).$$

However, we do not know $\Omega_{0,k}, k = 1, ..., K$. The key intuition behind SP variance estimation is that if we fail to include all pairs of influence functions that are truly dependent on one another, our estimate of the variance of the normal limiting distribution will be biased. Any variance estimator that contains at least all true dependent influence function pairs is unbiased.

SP variance estimation involves first constructing a sieve of variance estimators, where each variance estimate in the sieve assumes a different underlying dependence structure for the influence functions. For instance, in the case of the present analysis, suppose the true maximum dependence lag between any two classrooms in the same school was five calendar days. One could construct a very simple sieve by assuming each school was independent, but all classrooms within the same school were dependent up to τ calendar days into the past, with $0 \leq \tau \leq 10$. Then we could generate a set of 11 variance estimates, each based on a different value of τ.

Next, the sieve is ordered so that the variance estimators start out assuming to few independence assumptions and end up assuming too many. In the case of our toy example above, we could order our estimators by τ, so that the variance estimator that used $\tau = 0$ would come at the beginning of the sequence, and the one using $\boldsymbol{\tau} = 10$ would come at the end. Estimators with $\tau \geq 5$ are all unbiased, and they are all in the tail of the sequence.

Finally, we use a nonparametric regression technique called Pooled Adjacent Violators (PAV, [Turner, 2013, Robertson et al., 1988]) to help us find the plateau in our sequence, where the variance estimators are unbiased.

For this analysis, we constructed a very large sieve of variance estimators relying on the following notions about potential time dependence in our data.

1. Time points in winter may be more extensively dependent on the past than time points in nonwinter (or vice versa).

2. Time points with more extreme average illness absence may be more extensively dependent on past time points than those with less extreme average illness absence (or vice versa).

3. Time points with more extreme VRs may have had more lasting influence on future illness absence than time points with less extreme VRs.

4. All time points may have a constant dependence lag, as in our toy example above. While we do not believe this is true, this notion produces useful sieve elements.

We combined these notions with assumptions about hierarchical dependence. For instance, with respect to our notions of how winter might affect the dependence structure in our data, we defined a step function of temporal distance thresholds, $\boldsymbol{\tau}(t) = \alpha_1 I(t \in \text{winter}) + \alpha_2 I(t \notin \text{winter})$, where $(\alpha_1, \alpha_2) \in \{0, 1, \ldots, 21\}^2,\ \alpha_1 \neq \alpha_2$. Then for each unique classroom day (i, t), we constructed three different $\Omega_i(t)$ sets, consisting of

1. all observations in classroom i at most $\boldsymbol{\tau}(t)$ calendar days in the past,

2. all observations in the same grade within the same school as classroom i at most $\boldsymbol{\tau}(t)$ calendar days in the past, and

3. all observations in the same school as classroom i at most $\boldsymbol{\tau}(t)$ calendar days in the past.

We used an upper bound on the temporal extent of the dependence of any two influence function values of $\max(\boldsymbol{\tau}) = 21$. This threshold encompasses the vast majority of incubation periods for infectious respiratory diseases [Lessler et al., 2009]. Combining our temporal dependence notions with our hierarchical notions gave a sieve of $10,938$ unique variance estimators for each $\psi_n(v)$.

We tried three different sieve orderings:

1. ordering by the number of nonzero pairs of influence functions included in the estimator;

2. ordering by a particular kind of fit with respect to the variance estimate under independence which we refer to as L_1 fit;

3. ordering by *complexity*, which is a number proportional to the variance of each variance estimator.

See Davies and van der Laan [May 2014] for full definitions of L_1 fit and complexity in this context. We also used three different versions of the SP variance estimator algorithm. The first used the mode of the estimated PAV curve as σ_n^2. This mode was estimated by taking the peak of the estimated density of the PAV curve values. The second used the value of the longest plateau in the estimated PAV curve as σ_n^2. The

third used the sample variance estimate closest to the beginning of that longest plateau. Each of these versions is described in detail in Davies and van der Laan [May 2014].

We would expect each of these nine different combinations of orderings and estimator values to give us very similar results. Computing all nine served as a sanity check.

4.6 Results

Our results for each of the 12 threshold values are summarized in Table F. Figure 4.5 shows each $\psi_n(v)$ along with all of its estimated 0.95 confidence intervals. All $\psi_n(v)$ were (point-wise) statistically significant, but our standard error estimates assuming independence were substantially smaller than our more realistic SP-based standard error estimates. The SP variance estimates for each $\psi_n(v)$ were quite similar to one another across all orderings and estimator types. Figure 4.7 shows density plots of the estimated PAV curves for each threshold v and each ordering. We can see that the L_1 fit and number of non-zero pairs orderings are in quite close agreement across all v. The complexity ordering appears to have a stronger tendency toward multimodality, but its largest mode still consistently aligned with the modes of the other two orderings.

For each threshold value v, Figure 4.5d shows both the TML-based estimate $\psi_n(v)$ and a corresponding naive estimate, computed by taking the sample average of all $Y_i(t)$ amongst all $A_{v,i}(t) = 1$. We can see that the naive estimates are consistently larger than the TML estimates that adjusted for potential confounders. Figure 4.5d also contains the most conservative (largest) 0.95 CI for each $\psi_n(v)$.

Overall, our results show estimated average illness absence counts decreased as v increased. In fact, the decrease in illness absence was quite marked. The TML estimate $\psi_n(3.41)$ was 0.721, which is about 7% greater than $\psi_n(7.10)$, and about 11% greater than $\psi_n(9.56)$. Increasing v further still appears to improve estimated mean illness absence counts, but these improvements are fairly modest. In this sense, our results suggest there may in fact be a point of diminishing returns. However, our results also suggest that increasing the recommended threshold above 7.1 L/s/p could provide substantive benefits. We find that $\psi_n(7.10) = 0.671$ is 3% larger than $\psi_n(9.56)$ and 4% larger than $\psi_n(12.02)$. If one feels a causal interpretation of our results is warranted, this indicates we may be able to improve daily average classroom illness absence counts substantively by increasing the recommended VR threshold a relatively small amount.

To put these results into perspective, consider that there are roughly $1,438,900$ children enrolled in $51,389$ public third, fourth and fifth grade classrooms in the state of California in any given year [California Department of Education, 2012]. If all these classrooms had average daily illness absence counts of $0.671 = \psi_n(7.10)$, then we would expect to see about $6,206,763$ total illness absence counts annually amongst these classrooms. If all these classrooms increased their average daily classroom illness absence counts to $0.721 = \psi_n(3.41)$, we would expect to see an additional $462,501$ total illness absences annually. On the other hand, if all these classrooms reduced their average illness absence counts from 0.671 to $0.652 = \psi_n(9.56)$, we would expect to see $175,750$ fewer illness absence counts annually amongst these classrooms. While we

may not feel comfortable extrapolating our results to all California elementary schools, these numbers are substantial and warrant further investigation.

4.7 Discussion

In this paper, we examined the relationship between illness absence and recent past VRs in a subset of elementary school classrooms from a single California school district followed for a period of two years. For 12 VR thresholds, we used targeted, data-adaptive methods to estimate what average daily classroom illness absence counts would have been had all $\overline{V}(t)$ been less than v L/s/p. We also used a robust approach to standard error estimation to properly account for unknown dependence in the observed data structure. To our knowledge, this type of analysis is the first of its kind within the context of indoor air quality and human health.

There are very few studies in the literature examining the relationship between VR and/or CO_2 concentrations and school absence. Shendell et al. [2004] found a fairly strong relationship between classroom indoor versus outdoor CO_2 concentrations and average annual all-cause absence rates, using data from 434 permanent and portable classrooms from 22 different U.S. schools in Washington and Idaho. The authors used multivariate linear regression to adjust for grade, building type (permanent or portable), state, and school-level demographic variables. Their results showed a 1 L/s/p increase in classroom VR was associated with a 2.1 to 7.6% relative decrease in total classroom absence (Mendell et al. [2013] converted the author's findings from CO_2 concentrations to units of L/s/p (see Data S7)). Average indoor CO_2 concentrations were estimated from two short-term measurements, each no more than a five minute average.

In a more recent study, Gaihre et al. [2014] examined the relationship between average annual attendance rates and classroom CO_2 concentrations in 60 primary classrooms in 30 schools in Aberdeen, Scotland. Using linear mixed models, they found that an increase of 100 ppm of CO_2 was associated with a reduced annual attendance of 0.2%. An average CO_2 concentration was estimated for each classroom using measurements made in early Summer over a three to five day period.

By contrast, the data used in Mendell et al. [2013] (and the present analysis) was much more temporally extensive and detailed. Daily VRs were estimated using sensors left in place over the entire study period, and only those absences classified as illness-related were included in the analysis. The authors used zero-inflated negative binomial regression to assess the relationship between daily classroom VR and illness absence. With respect to the district in the present study, the authors found that each additional 1 L/s/p of VR was associated with 1.2% decrease in classroom illness absence rate ($p < 0.05$).

The present study is quite different from other work in this area, in that our target parameter is not a coefficient in a parametric model. Rather, we were interested in better understanding the nature of the trend in expected illness absence given that a summary measure of recent past VR failed to meet a threshold level of v L/s/p. We found that as v increased, illness absence counts decreased, but the effect attenuated at

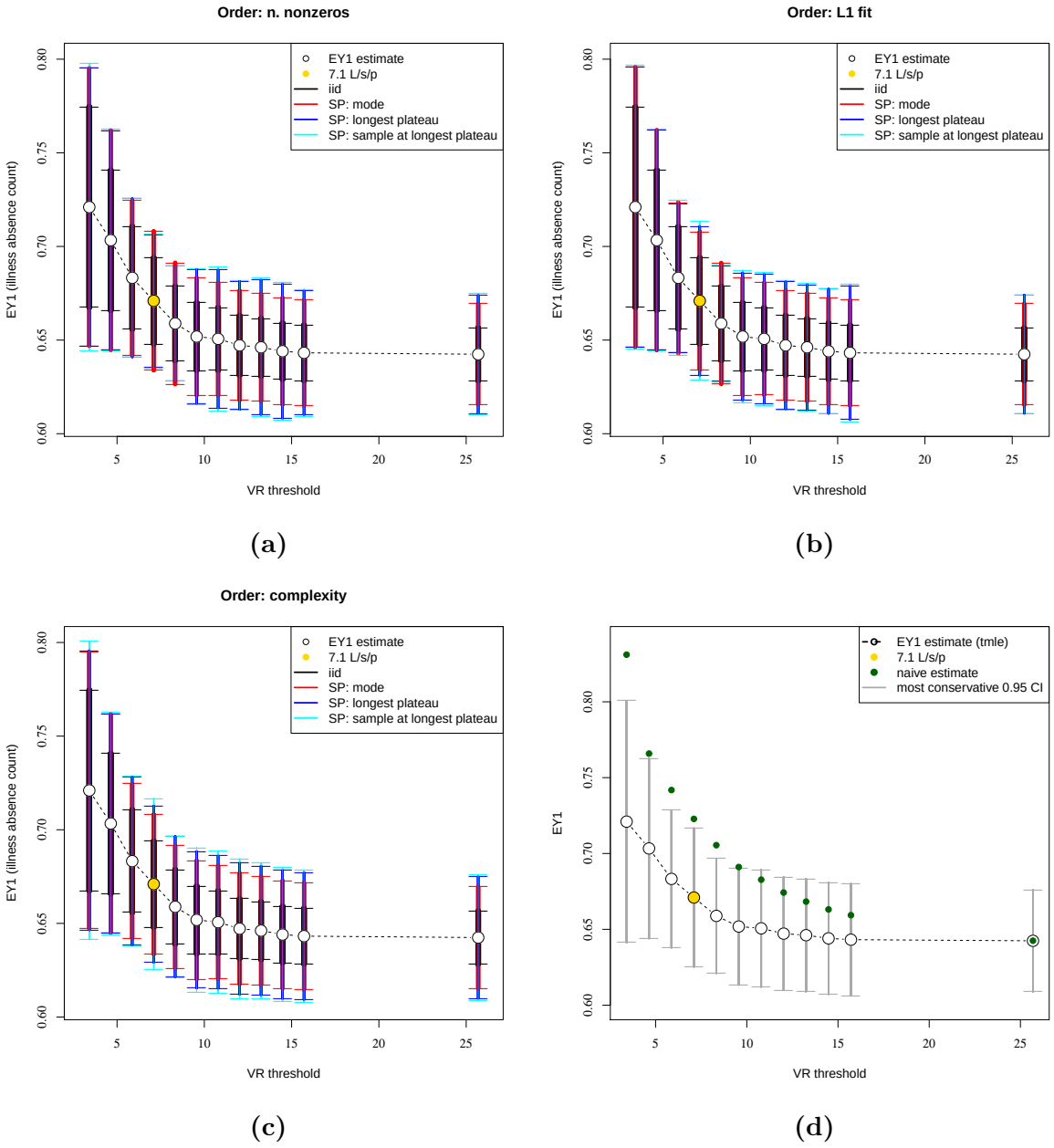


Figure 4.5

Figure 4.6: $\psi_n(v)$ and estimated confidence intervals for each VR threshold v. Estimated 0.95 confidence intervals assuming indepedent observations are substantively smaller than those based on SP variance estimation. All nine SP-based confidence intervals tended to be in close agreement with one another. In Figure (d), $\psi_n(v)$ and naive estimates for each VR threshold v, along with the largest (most conservative) estimated confidence interval.

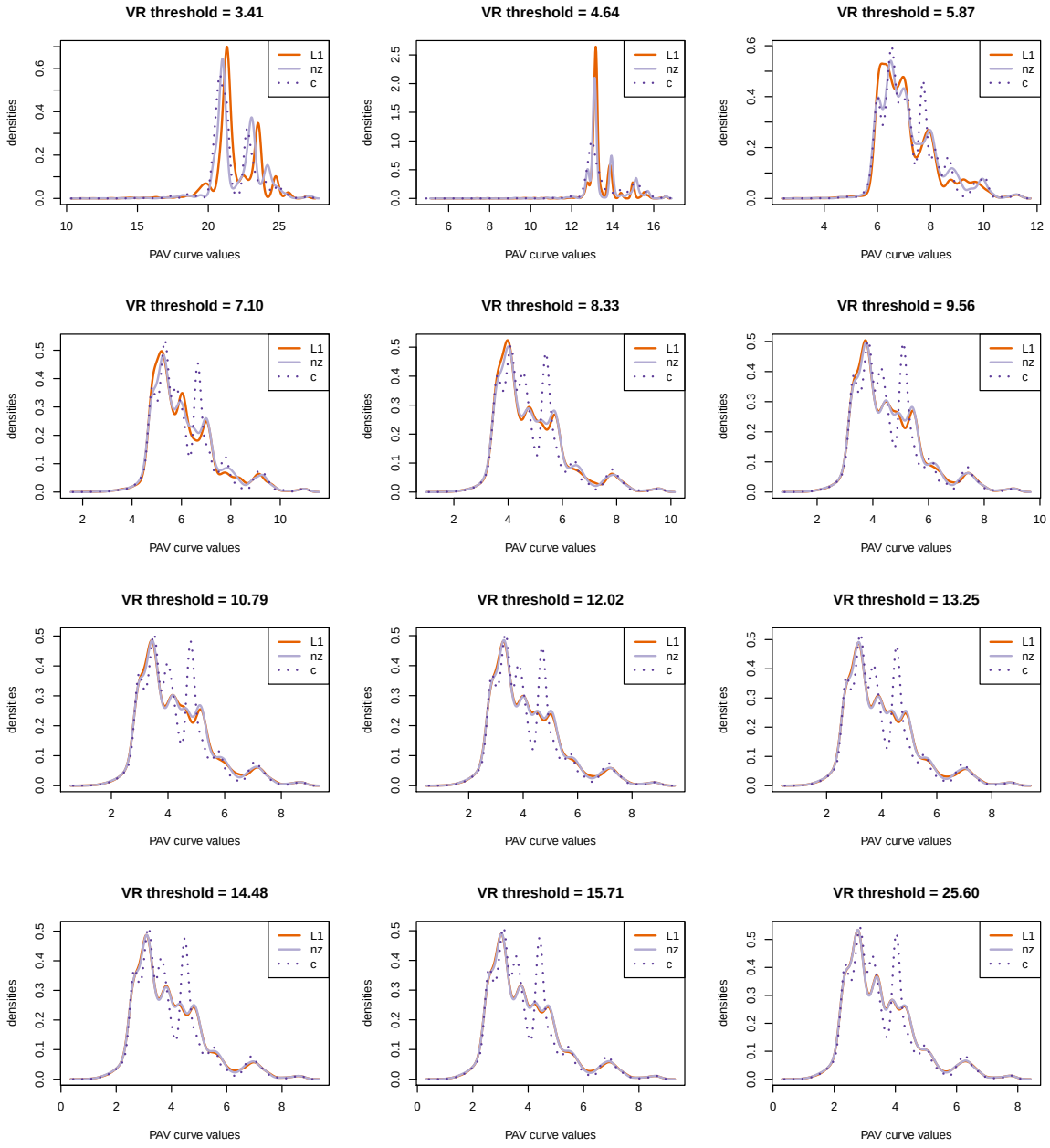


Figure 4.7: Densities of PAV curve values for each v and each sieve ordering. All three orderings tended to have very similar modes, although the complexity ordering exhibited more pronounced multimodality.

higher v. Our results suggest that a modest increase in the current recommended VR threshold might substantially reduce illness absence counts in this particular California school district. Additional analyses will be needed to confirm whether this finding is more generally applicable.

It is important to note here that a large proportion of the classroom days we observed failed to meet the current recommended guidelines of 7.1 L/s/p. From a policy perspective, this suggests that simply increasing the recommended threshold will most likely not lead to immediate reductions in illness absence rates, especially in classrooms with no mechanical ventilation system. Infrastructure improvements would most likely be required in order to increase VRs in these rooms throughout the year.

This analysis was limited to examining relatively short-term relationships. Other exposure metrics could also be illuminating. Longer exposure time windows could provide insight into the effects of chronic exposure, for instance. A nonparametric causal dose response curve as described in Diaz and van der Laan would also be quite useful, and is future work.

Bibliography

a.

b.

S Arlot and A Celisse. A Survey of Cross-validation Procedures for Model Selection. *Statistics Surveys*, 4:44–79, 2010.

ANSI/ASHRAE Standard 62.1-2010: Ventilation For Acceptable Indoor Air Quality. ASHRAE, Inc., Atlanta, Georgia, 2010.

K Bauman and J Davis. Estimates of school enrollment by grade in the american community survey, the current population survey, and the common core survey: SEHSD Working Paper 2014-7, 2013. URL `http://www.census.gov/hhes/school/files/ACS-CPS-CCD`$_0 2 - 18 - 14.pdf$.

PJ Bickel, CAJ Klaassen, Y Ritov, and JA Wellner. *Efficient and adpative estimation for semiparametric models*. The Johns Hopkins University Press, Baltimore, MD, 1993.

L Breiman. Stacked Regressions. *Machine Learning*, 24(1):49–64, JUL 1996.

L Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

California Department of Education. School facilities fingertip facts, 2012.

PS Carmack, WR Schucany, JS Spence, RF Gunst, Q Lin, and RW Haley. Far Casting Cross-validation. *Journal of Computational and Graphical Statistics*, 18(4):879–893, 2009.

J Chen and C Wang. Using Stacked Generalization to Combine SVMs in Magnitude and Shape Feature Spaces for Classification of Hyperspectral Data. *IEEE Transactions on Geoscience and Remote Sensing*, 47(7):2193–2205, 2009.

P Craven and G Wahba. Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

NAC Cressie. *Statistics for Spatial Data*. Wiley Series in probability and mathematical statistics. John Wiley and Sons, Inc., revised edition, 1993.

A DasGupta. *Asymptotic Theory of Statistics and Probability*, chapter chapter 29: The Bootstrap, pages 468–471. Springer, 2008.

MM Davies and MJ van der Laan. Sieve plateau variance estimators: a new approach to confidence interval estimation for dependent data. *U.C. Berkeley Division of Biostatistics Working Paper Series*, (Working Paper 322), May 2014. URL `http://biostats.bepress.com/ucbbiostat/paper322/`.

BM Davis. Uses and Abuses of Cross-Validation in Geostatistics. *Mathematical Geology*, 19(3):241–248, 1987.

I Diaz and MJ van der Laan. Targeted data-adaptive estimation of the causal dose respose curve. *Journal of Causal Inference*, 1:171–192.

PJ Diggle and PJ Ribeiro Jr. *Model Based Geostatistics*. Springer, New York, 2007.

S Dudoit and MJ van der Laan. Asymptotics of Cross-validated Risk Estimation in Estimator Selection and Performance Assessment. *Statistical Methodology*, 2(2):131–154, 2005.

JM Ellers, DH Landers, and DF Brakke. Chemical and Physical Characteristics of Lakes in the Southeastern United States. *Environmental Science and Technology*, 22 (2):172–177, 1988.

M Francisco-Fernandez and JD Opsomer. Smoothing Parameter Selection Methods for Nonparametric Regression with Spatially Correlated Errors. *The Canadian Journal of Statistics*, 33(2):279–295, 2005.

J Friedman, T Hastie, and R Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 2010. URL `http://www.jstatsoft.org/v33/i01/`.

JH Friedman. Greedy Function Approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):367–378, 2001.

R Furrer, D Nychka, and S Sain. *fields: Tools for spatial data*, 2011. URL `http://CRAN.R-project.org/package=fields`. R package version 6.6.1.

S Gaihre, S Semple, J Miller, S Fielding, and S Turner. Classroom carbon dioxide concentration, school attendance, and educational attainment. *Journal of School Health*, 84:569–74, 2014.

S Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

AE Gelfand, PJ Diggle, M Fuentes, and P Guttorp, editors. *Handbook of Spatial Statistics*. CRC Press, 2010.

PJ Green and BW Silverman. *Nonparametric Regression and Generalized Linear Models*. Number 58 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, 1994.

S Gruber and MJ van der Laan. A targeted maximum likelihood estimator of a causal effect on a bounded continuous outcome. *The International Journal of Biostatistics*, 6(1), 2010.

S Gruber and MJ van der Laan. tmle: An R package for targeted maximum likelihood estimation. *Journal of Statistical Software*, 51(13):1–35, 2012. URL `http://www.jstatsoft.org/v51/i13/`.

C Gu. *Smoothing spline ANOVA models*. Springer - New York, 2002.

C Gu. *gss: General Smoothing Splines*, 2012. URL `http://CRAN.R-project.org/package=gss`. R package version 2.0-11.

L Györfi, MAK Kohler, and H Walk. *A Distribution-free Theory of Nonparametric Regression*. Springer Series in Statistics. Springer, August 2002.

F Hampel. The infuence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69:383–93, 1974.

T Hastie. *gam: Generalized Additive Models*, 2011. URL `http://CRAN.R-project.org/package=gam`. R package version 1.04.1.

TJ Hastie. *Statistical models in S*, chapter 7: Generalized Additive Models. Wadsworth and Brooks/Cole, 1991.

HC Huang and CS Chen. Optimal Geostatistical Model Selection. *Journal of the American Statistical Association*, 102(479):1009–1024, 2007.

W Jiang. On Uniform Deviations of General Empirical Risks with Unboundedness, Dependence, and High Dimensionality. *Journal of Machine Learning Research*, 10: 977–996, 2009.

A Karatzoglou, A Smola, K Hornik, and A Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL `http://www.jstatsoft.org/v11/i09/`.

W Kleiber, AE Raftery, and T Gneiting. Geostatistical model averaging for locally calibrated probabilistic quantitative precipitation forecasting. *Journal of the American Statistical Association*, 106(496):1291–1303, 2011.

NE Klepeis, AM Tsang, and JV Behar. Analysis of the national human activity pattern survey (nhaps) respondents from a standpoint of exposure assessment - final epa report. Technical report, Environmental Protection Agency, Washington D.C., 1996.

C Kooperberg. *polspline: Polynomial spline routines*, 2010. URL `http://CRAN.R-project.org/package=polspline`. R package version 1.1.5.

SN Lahiri. *Resampling Methods for Dependent Data*. Springer, New York, 2013.

M LeBlanc and R Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91:1641–1650, 1996.

J Lessler, NG Reich, R Brookmeyer, TM Perl, KE Nelson, and Cummings. Incubation periods of acute respiratory viral infections: a systematic review. *The Lancet Infectious Diseases*, 9:291–300, 2009.

S Li. *FNN: Fast Nearest Neighbor Search Algorithms and Applications*, 2012. URL `http://CRAN.R-project.org/package=FNN`. R package version 0.6-3.

Y Li, GM Leung, JW Tang, X Yang, CYH Chang, JZ Lin, JW Lu, PV Nielsen, J Niu, H Qian, AC Sleigh, HJJ Su, J Sundell, and TW Wong. Role of ventilation in airborne transmission of infectious agents in the built environment - a multidisciplinary systematic review. *Indoor Air*, 17:2–18, 2007.

A Liaw and M Wiener. Classification and regression by randomforest. *R News*, 2(3): 18–22, 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

T Lumley. An empirical process limit theorem for sparsely correlated data. *UW Biostatistics Working Paper Series*, (255), 2005.

E Mammen. When does the bootstrap work? Asymptotic results and simulations. In *Springer Lecture Notes in Statistics 77*. Singer Verlag, Heidelberg, 1992.

B Matérn. *Spatial Variation*. Springer - New York, 2nd edition, 1986.

MJ Mendell and WJ Fisk. Developing evidence-based prescriptive ventilation rate standards for commercial buildings in california: a proposed framework. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA., 2014.

MJ Mendell, EA Eliseeva, MM Davies, M Spears, A Lobscheid, WJ Fisk, and MG Apte. Association of classroom ventilation with reduced illness absence: a prospective study in california elementary schools. *Indoor Air*, 23(6):515–528, 2013.

R Neugebauer and J Bullard. *DSA: Deletion/Substitution/Addition algorithm*, 2010. URL `http://www.stat.berkeley.edu/ laan/Software/`. R package version 3.1.4.

JD Opsomer, Y Wang, and Y Yang. Nonparametric regression with correlated errors. *Statistical Science*, 16(2):134–153, 2001.

A Patton, DN Politis, and H White. CORRECTION TO "Automatic block-length selection for the dependent bootstrap" by D. Politis and H. White. 2009.

A Persily. Challenges in developing ventilation and indoor air quality standards: The story of ASHRAE standard 62. *Building and Environment*, 2015.

DN Politis and JP Romano. Nonparametric resampling for homogeneous strong mixing random fields. *Journal of Multivariate Analysis*, 47:301–328, 1993.

DN Politis, JR Romano, and M Wolf. *Subsampling*. Springer, New York, 1999.

E Polley and MJ van der Laan. *SuperLearner: Super Learner Prediction*, 2012. URL `http://CRAN.R-project.org/package=SuperLearner`. R package version 2.0-6.

EC Polley, S Rose, and MJ van der Laan. *Targeted Learning: Casual Inference for Observational and Experimental Data*, chapter 3: Super Learning, pages 43–65. Springer, New York, 2011a.

EC Polley, S Rose, and MJ van der Laan. *Targeted Learning: Casual Inference for Observational and Experimental Data*, chapter 3: Super Learning, pages 43–65. Springer, New York, 2011b.

Eric C. Polley and Mark J. van der Laan. Super learner in prediction. *U.C. Berkeley Division of Biostatistics Working Paper Series*, 2010.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`.

JR Ribeiro and PJ Diggle. geoR: a package for geostatistical analysis. *R-NEWS*, 1(2): 14–18, June 2001. URL `http://CRAN.R-project.org/doc/Rnews/`.

G Ridgeway. *gbm: Generalized Boosted Regression Models*, 2010. URL `http://CRAN.R-project.org/package=gbm`. R package version 1.6-3.1.

T Robertson, FT Wright, and RL Dykstra. Wiley, New York, 1988.

S Rose and MJ van der Laan. *Targeted Learning: Casual Inference for Observational and Experimental Data*, chapter 4: Introduction to TMLE, pages 69–82. Springer, New York, 2011a.

S Rose and MJ van der Laan. *Targeted Learning: Casual Inference for Observational and Experimental Data*. Springer, New York, 2011b.

M Rossi, F Guzzetti, P Reichenbach, AC Mondini, and S Peruccacci. Optimal landslide susceptibility zonation based on multiple forecasts. *Geomorphology*, 114(3):129–142, JAN 15 2010. ISSN 0169-555X. doi: 10.1016/j.geomorph.2009.06.020.

O Schabenberger and CA Gotway. *Statistical Methods for Spatial Data Analysis*. Texts in Statistical Science. Chapman & Hall, CRC, 2005.

O Seppänen, WJ Fisk, and MJ Mendell. Association of ventilation rates and co2 concentrations with health and other responses in commerical and institutional buildings. *Indoor Air*, 9(4):226–252, 1999.

DG Shendell, R Prill, WJ Fisk, MG Apte, D Blake, and D Faulkner. Associations between classroom co2 concentrations and student attendance in washington and idaho. *Indoor Air*, 14:333–341, 2004.

BW Silverman. *Density Estimation*. London: Chapman and Hall, 1986.

E Simons, S Hwang, EF Fitzgerald, C Kielb, and S Lin. The impact of school building conditions on student absenteeism in upstate New York. *American Journal of Public Health*, pages 1679–1686, 2010.

SE Sinisi and MJ van der Laan. The deletion/substitution/addition algorithm in loss function based estimation. *Journal of Statistical Methods in Molecular Biology*, 3(1), 2004.

CJ Stone, M Hansom, C Kooperberg, and YK Truong. The use of polynomial splines and their tensor products in extended linear modeling (with discussion). *Annals of Statistics*, 25:1371–1470, 1997.

M Stone. Cross-validatory choice and assessment of statistical procedures. *Journal of the Royal Statistical Society, Series B*, 36(2):111–147, 1974.

E Todini. Influence of parameter estimation on uncertainty in Kriging: Part 1 - Theoretical Development. *Hydrology and Earth Systems Sciences*, 5(2):215–223, 2001.

R Turner. *Iso: Functions to perform isotonic regression*, 2013. URL `http://CRAN.R-project.org/package=Iso`. R package version 0.0-15.

MJ van der Laan. Causal inference for a population of causally connected units. *Journal of Causal Inference*, 2(1):13–74, 2014.

MJ van der Laan and S Dudoit. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. *U.C.Berkeley Division of Biostatistics Working Paper Series*, (130), 2003.

MJ van der Laan and D Rubin. Targeted maximum likelihood learning. *The International Journal of Biostatistics*, 2(1), 2006.

MJ van der Laan, S Dudoit, and S Keles. Asymptotic optimality of likelihood based cross-validation. *Statistical Applications in Genetics and Molecular Biology*, 3(1): Article 4, 2004.

MJ van der Laan, EC Polley, and AE Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

AW van der Vaart, S Dudoit, and MJ van der Laan. Oracle inequalities for multi-fold cross validation. *Statistics and Decisions*, 24:351–371, 2006.

CKI Williams. *Learning in Graphical Models*, chapter Prediction with Gaussian processes: from linear regression to linear prediction and beyond, pages 599–621. The MIT Press, Cambridge, MA, 1999.

DH Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, 1992.

I Zaier, C Shu, TBMJ Ouarda, O Seidou, and F Chebana. Estimation of ice thickness on lakes using artificial neural network ensembles. *Journal of Hydrology*, 383:330–340, 2010.

Appendix A

Spatial Prediction - Oracle Inequality for Independent, Nonidentical Experiments and Quadratic Loss

Let $\mathbf{O}^n = (O_1, \ldots, O_n) \sim P_0^n$ be a vector of independent, nonidentical observations, where each $O_i = (\mathbf{X}_i, Y_i)$ consists of two components: a d-dimensional covariate vector $\mathbf{X}_i \in \mathbb{R}^d$, and a univariate outcome $Y_i \in \mathbb{R}$. We associate with each O_i an index $s_i \in S$. The true unknown data generating distribution for each O_i is denoted $P_{0,O_i}(O) = P_{0,O|S}(O|s_i) \in \{P_{0,O|s} : s \in S\}$. Let $P_{s,O}$ be the joint distribution of (S, O), defined by a degenerate marginal distribution of S, $I(S = s)$, and the conditional distribution of O given $S = s$, $P_{O|s}$. We can formulate $\mathbf{O}^n$ as n independent draws $(S_i, O_i) \sim P_{0,(s_i,O_i)}$, $i = 1, \ldots, n$, with empirical probability distribution P_n. Let $\mathcal{M}$ be a set of possible probability distributions of $P_{O|S}$. Define a parameter $\Psi : \mathcal{M} \to \mathbf{\Psi}$, and let $\psi_0 = \Psi(P_{0,O|S})$ be the true value of that parameter. Let $\mathbf{B}_n \in \{0,1\}^n$ be a random vector indicating splits into a training sample, $\{i : \mathbf{B}_n(i) = 0\}$, and validation sample, $\{i : \mathbf{B}_n(i) = 1\}$. Let $p = 1/n \sum_{i=1}^n \mathbf{B}_n(i)$ be the proportion of observations in the validation sample, and let $P^0_{n,\mathbf{B}_n}$ and $P^1_{n,\mathbf{B}_n}$ be the empirical distributions of the training and validation samples, respectively. Define an average joint distribution: $\overline{P}^1_{0,\mathbf{B}_n} = (np)^{-1} \sum_{i:\mathbf{B}_n(i)=1} P_{0,(s_i,O_i)}$. Let $L(\psi)(S, O)$ be a loss function such that for all i, $P_{0,(s_i,O)} L(\psi_0) = \min_{\psi \in \mathbf{\Psi}} P_{0,(s_i,O_i)} L(\psi)$. Let $\{\widehat{\Psi}_k(P_n) : k = 1, \ldots, K_n\}$ be a set of K_n estimators of ψ_0. Assume $\mathbb{P}(\widehat{\Psi}_k(P_n) \in \Psi) = 1$ for all $k = 1, \ldots, K_n$. We write the true cross-validated risk of ψ_0 as $\widetilde{\Theta}_{opt} = \mathbb{E}_{\mathbf{B}_n}\big[\, \overline{P}^1_{0,\mathbf{B}_n} L(\psi_0)\big]$. We denote the true conditional cross-validated risk of any estimator $\widehat{\Psi}_k$ as

$$\begin{aligned}
\widetilde{\Theta}_{n(1-p)}(k) &\equiv \mathbb{E}_{\mathbf{B}_n}\left[\overline{P}^1_{0,\mathbf{B}_n} L\left(\widehat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right)\right] \\
&= \mathbb{E}_{\mathbf{B}_n}\left[\frac{1}{np} \sum_{i:\mathbf{B}_n(i)=1} P_{0,(O_i|s_i)} L\left(\widehat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right)(s_i, O_i)\right],
\end{aligned}$$

and a benchmark (oracle) selector as $\tilde{k}_{n(1-p)} = \text{argmin}_k \widetilde{\Theta}_{n(1-p)}(k)$.

We denote the cross-validated risk of any estimator $\widehat{\Psi}_k$ as

$$\widehat{\Theta}_{n(1-p)}(k) \equiv \mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n} L\left(\widehat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right]\right)\right] = \mathbb{E}_{\mathbf{B}_n}\left[\frac{1}{np}\sum_{i:\mathbf{B}_n(i)=1} L\left(\widehat{\Psi}\left[P^0_{n,\mathbf{B}_n}\right]\right)(s_i, O_i)\right],$$

and the cross-validation selector as $k_n = \text{argmin}_k\ \widehat{\Theta}_{n(1-p)}(k)$. Finally, we define a loss-based dissimilarity $d_n(\psi, \psi_0) \equiv \mathbb{E}_{\mathbf{B}_n}\left[\overline{P}^1_{0,\mathbf{B}_n}\Big(L[\psi] - L[\psi_0]\Big)\right]$.

Assumptions.

A1. There exists a real-valued $M_1^* < \infty$ such that
$\sup_{\psi\in\mathbf{\Psi}}\left\{\sup_{i,s_i,O_i}\left|L(\psi)(s_i,O_i) - L(\psi_0)(s_i,O_i)\right|\right\} \leq M_1^*$, where the supremum over O_i is taken over the support of the distribution $P_{0,O_i|s_i}$ of O_i.

A2. There exists a real-valued $M_2 < \infty$ such that

$$\sup_{i,\psi\in\mathbf{\Psi}}\left\{\frac{\text{Var}_{P_{0,(s_i,O_i)}}\big[L(\psi) - L(\psi_0)\big](S,O)}{\mathbb{E}_{P_{0,(s_i,O_i)}}\big[L(\psi) - L(\psi_0)\big](S,O)}\right\} \leq M_2.$$

Definitions. We define the following constants: $M_1 = 2M_1^*$; $C(M_1, M_2, \delta) \equiv 2(1+\delta)^2\left(\frac{M_1}{3} + \frac{M_2}{\delta}\right)$.

Finite sample result. For any $\delta > 0$, we have

$$\begin{aligned}\mathbb{E}\left[d_n\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right] \leq\ & (1+2\delta)\,\mathbb{E}\left[d_n\left(\widehat{\Psi}_{\tilde{k}_{n(1-p)}}\left[P^0_{n,\mathbf{B}_n}\right], \psi_0\right)\right] \\ & + 2\,C(M_1, M_2, \delta)\,\frac{1+\log K_n}{np}.\end{aligned} \tag{A.1}$$

Asymptotitic implications. (A.1) has the following asymptotic implications:

$$\frac{\log K_n}{np\,\mathbb{E}\left[\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}\right]} \xrightarrow{n\to\infty} 0 \Longrightarrow \frac{\mathbb{E}\left[\widetilde{\Theta}_{n(1-p)}(k_n) - \widetilde{\Theta}_{opt}\right]}{\mathbb{E}\left[\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}\right]} \xrightarrow{n\to\infty} 1.$$

$$\frac{\log K_n}{np\,\left(\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}\right)} \xrightarrow{p} 0 \Longrightarrow \frac{\widetilde{\Theta}_{n(1-p)}(k_n) - \widetilde{\Theta}_{opt}}{\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}} \xrightarrow{p} 1. \tag{A.2}$$

(A.2) follows from the fact that, given a sequence of random variables $X_1, X_2, \ldots$, and a positive function $g(n)$, $\mathbb{E}|X_n| = O\{g(n)\}$ implies $X_n = O_P\{g(n)\}$. This is a direct consequence of Markov's inequality.

Proof of theorem. We have

$$
\begin{aligned}
0 &\leq \widetilde{\Theta}_{n(1-p)}(k_n) - \widetilde{\Theta}_{opt} && \text{(A.3a)}\\
&= \mathbb{E}_{\mathbf{B}_n}\left[\overline{P}^1_{0,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right]\\
&\quad - (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right]\\
&\quad + (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right]\\
&\leq \mathbb{E}_{\mathbf{B}_n}\left[\overline{P}^1_{0,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3b)}\\
&\quad - (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right]\\
&\quad + (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{\tilde{k}_{n(1-p)}}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right]\\
&= \mathbb{E}_{\mathbf{B}_n}\left[\overline{P}^1_{0,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3c)}\\
&\quad - (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{k_n}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3d)}\\
&\quad + (1+\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{\tilde{k}_{n(1-p)}}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3e)}\\
&\quad - (1+2\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{\tilde{k}_{n(1-p)}}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3f)}\\
&\quad + (1+2\delta)\,\mathbb{E}_{\mathbf{B}_n}\left[P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_{\tilde{k}_{n(1-p)}}\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\right] && \text{(A.3g)}
\end{aligned}
$$

(A.3a) follows from the definition of $\widetilde{\Theta}_{opt}$. (A.3b) follows from the definition of the cross-validation selector k_n, such that for all k, $\widehat{\Theta}_{n(1-p)}(k_n) \leq \widehat{\Theta}_{n(1-p)}(k)$. Let R_{n,k_n} represent the first two terms in the last expression, (A.3c) and (A.3d). Let $T_{n,\tilde{k}_{n(1-p)}}$ represent the second two terms of the last expression, (A.3e) and (A.3f). The last term, (A.3g), is the benchmark and can be written as $(1+2\delta)\left[\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}\right]$. Hence,

$$
\begin{aligned}
0 \leq \widetilde{\Theta}_{n(1-p)}(k_n) - \widetilde{\Theta}_{opt} \leq (1+2\delta)\ &\left[\widetilde{\Theta}_{n(1-p)}\left(\tilde{k}_{n(1-p)}\right) - \widetilde{\Theta}_{opt}\right]\\
&+ R_{n,k_n} + T_{n,\tilde{k}_{n(1-p)}}.
\end{aligned}
\qquad \text{(A.4)}
$$

We now show that $\mathbb{E}R_{n,k_n} + \mathbb{E}T_{n,\tilde{k}_{n(1-p)}} \leq 2C(M_1, M_2, \delta)\ (1+\log K_n)/(np)$. We introduce the following notation:

$$
\begin{aligned}
\widehat{H}_k &\equiv P^1_{n,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\\
\widetilde{H}_k &\equiv \overline{P}^1_{0,\mathbf{B}_n}\left\{L\left(\widehat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}\\
R_{n,k}(\mathbf{B}_n) &\equiv (1+\delta)\left[\widetilde{H}_k - \widehat{H}_k\right] - \delta\widetilde{H}_k\\
T_{n,k}(\mathbf{B}_n) &\equiv (1+\delta)\left[\widehat{H}_k - \widetilde{H}_k\right] - \delta\widetilde{H}_k
\end{aligned}
$$

Note that $R_{n,k} = \mathbb{E}_{\mathbf{B}_n}[R_{n,k}(\mathbf{B}_n)]$; $T_{n,k} = \mathbb{E}_{\mathbf{B}_n}[T_{n,k}(\mathbf{B}_n)]$; and that by definition of ψ_0, $\widetilde{H}_k \geq 0$ for all k. Note also that given an arbitrary $k \in \{1, \ldots K_n\}$,

$$\begin{aligned}\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n) > s \mid P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] &= \mathbb{P}\left[\widetilde{H}_{k_n} - H_{k_n} > \frac{s + \delta\widetilde{H}_{k_n}}{1+\delta} \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] \\ &\leq K_n \max_k \mathbb{P}\left[\widetilde{H}_k - \widehat{H}_k > \frac{s + \delta\widetilde{H}_k}{1+\delta} \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].\end{aligned}$$

Similarly for $T_{n,\tilde{k}_{n(1-p)}}(\mathbf{B}_n)$,

$$\mathbb{P}\left[T_{n,\tilde{k}_{n(1-p)}}(\mathbf{B}_n) > s \mid P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] = K_n \max_k \mathbb{P}\left[\widehat{H}_k - \widetilde{H}_k > \frac{s + \delta\widetilde{H}_k}{1+\delta} \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].$$

Conditional on $P^0_{n,\mathbf{B}_n}$ and $\mathbf{B}_n$, consider the np random variables for which $\mathbf{B}_n(i) = 1$, $Z_{k,i} \equiv \left\{L\left(\widehat{\Psi}_k\left[P^0_{n,\mathbf{B}_n}\right]\right) - L(\psi_0)\right\}(s_i, O_i)$. We can rewrite $\widehat{H}_k$ and $\widetilde{H}_k$ in terms of $Z_{k,i}$,

$$\begin{aligned}\widehat{H}_k &= \frac{1}{np}\sum_{i=1}^{np} Z_{k,i}, \\ \widetilde{H}_k &= \frac{1}{np}\sum_{i=1}^{np} \mathbb{E}\left[Z_{k,i} | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right].\end{aligned}$$

Then $\widetilde{H}_k - \widehat{H}_k$ is the sum of np mean zero centered random variables. By assumption **A1** above, the random variables $Z_{i,k}$ are bounded, with $|Z_{i,k}| \leq M_1$ a.s. By assumption **A2**, we also have

$$\sigma^2_{k,i} \equiv \mathrm{Var}\left[Z_{k,i} | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] \leq M_2\, \mathbb{E}\left[Z_{k,i} | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right],$$

which implies

$$\sigma^2_k \equiv \frac{1}{np}\sum_{i=1}^{np}\sigma^2_{k,i} \leq M_2 \frac{1}{np}\sum_{i=1}^{np}\mathbb{E}\left[Z_{k,i} | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] = M_2\widetilde{H}_k.$$

We will apply Bernstein's inequality to the centered empirical mean $\widetilde{H}_k - \widehat{H}_k$ and obtain a tail probability bounded by $\exp\{-npq/c\}$, where c is a finite, real-valued constant. This will show that the risk dissimilarities converge at a rate of $(\log K_n)/np$. We state Bernstein's inequality for ease of reference. A proof is given in Lemma A.2 on page 594 in Györfi et al. [2002].

Lemma 1. *Bernstein's inequality.*
Let $Z_i, i = 1, \ldots, n$ be independent, real valued random variables such that $Z_i \in [a, b]$ with probability one. Let $0 < \sum_{i=1}^n Var(Z_i)/n \leq \sigma^2$. Then, for all $\epsilon > 0$,

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n (Z_i - \mathbb{E}Z_i) > \epsilon\right) \leq \exp\left\{\frac{-n\epsilon^2}{2(\sigma^2 + \epsilon(b-a)/3)}\right\}.$$

This implies

$$\mathbb{P}\left(\left| \frac{1}{n}\sum_{i=1}^{n}(Z_i - \mathbb{E}Z_i) > \epsilon \right| \right) \leq 2\exp\left\{ \frac{-n\epsilon^2}{2(\sigma^2 + \epsilon(b-a)/3)} \right\}.$$

By Bernstein's lemma, for $q > 0$,

$$\begin{aligned}
\mathbb{P}\left[R_{n,k}(\mathbf{B}_n) > q | P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] &= \mathbb{P}\left[\widetilde{H}_k - \widehat{H}_k > \frac{1}{1+\delta}\left(q + \delta\widetilde{H}_k\right) \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n \right] \\
&\leq \mathbb{P}\left[\widetilde{H}_k - \widehat{H}_k > \frac{1}{1+\delta}\left(q + \frac{\delta\sigma_k^2}{M_2}\right) \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n \right] \\
&\leq \exp\left\{ -\left(\frac{np}{2[1+\delta]^2}\right)\left(\frac{\left[q + \delta\sigma_k^2/M_2\right]^2}{\sigma_k^2 + \frac{M_1}{3(1+\delta)}\left[q + \delta\sigma_k^2/M_2\right]}\right)\right\}.
\end{aligned}$$

Note that

$$\frac{\left[q + \delta\sigma_k^2/M_2\right]^2}{\sigma_k^2 + \frac{M_1}{3(1+\delta)}\left[q + \delta\sigma_k^2/M_2\right]} = \frac{\left[s + \delta\sigma_k^2/M_2\right]^2}{\frac{\sigma_k^2}{q+\sigma_k^2/M_2} + \frac{M_1}{3(1+\delta)}} \geq \frac{\left[s + \delta\sigma_k^2/M_2\right]^2}{\frac{M_2}{\delta} + \frac{M_1}{3}} \geq \frac{s}{\frac{M_2}{\delta} + \frac{M_1}{3}}.$$

This shows that for $q > 0$,

$$\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n) > q \;\middle|\; P^0_{n,\mathbf{B}_n}, \mathbf{B}_n\right] \leq K_n \exp\left\{(-npq)/C(M_1, M_2, \delta)\right\}.$$

In particular, this provides us with a bound for the marginal probability of $R_{n,k_n}(\mathbf{B}_n)$,

$$\mathbb{P}\left[R_{n,k_n}(\mathbf{B}_n) > q\right] \leq K_n \exp\left\{(-npq)/C(M_1, M_2, \delta)\right\}.$$

As in the proof of theorem 1 in van der Laan et al. [2004] and Dudoit and van der Laan [2005], for each $u > 0$, we have

$$\mathbb{E}\left[R_{n,k_n}\right] \leq u + \int_u^\infty K_n \exp\left\{(-npq)/C(M_1, M_2, \delta)\right\} \mathrm{d}q.$$

The minimum is attained at $u_n = C(M_1, M_2, \delta)\log K_n/np$ and is given by $C(M_1, M_2, \delta)(\log K_n + 1)/np$. Thus $\mathbb{E}R_{n,k_n} \leq C(M_1, M_2, \delta)(1 + \log K_n)(np)$. The same applies for $\mathbb{E}T_{n,\tilde{k}_{n(1-p)}}$.

Taking the expected values of the quantities in (A.4) yields the following finite sample result:

$$\begin{aligned}
0 &\leq \mathbb{E}\left[\widetilde{\Theta}_{n(1-p)}(k_n)\right] - \widetilde{\Theta}_{opt} \\
&\leq (1+2\delta)\left(\mathbb{E}\left[\widetilde{\Theta}_{n(1-p)}\right] - \widetilde{\Theta}_{opt}\right) + 2C(M_1, M_2, \delta)\left[\frac{1 + \log K_n}{np}\right].
\end{aligned}$$

This completes the proof. ■

Appendix B

Spatial Prediction - Tables

Algorithm	Parameters	Avg FVU
Super Learner		0.24 (0.22)
DSA (v, m, k, p)	$(5, 5, 3, 10)$	0.62 (0.29)
	$(5, 5, 3, 5)$	0.61 (0.26)
	$(5, 5, 4, 10)$	0.62 (0.28)
	$(5, 5, 4, 5)$	0.61 (0.26)
GAM (degree)	2	0.65 (0.28)
	3	0.64 (0.29)
	4	0.65 (0.30)
	5	0.65 (0.31)
	6	0.66 (0.32)
GBM (degree)	1	0.64 (0.28)
	2	0.56 (0.28)
	3	0.53 (0.30)
	4	0.52 (0.30)
	5	0.51 (0.31)
	6	0.50 (0.31)
GLM		0.69 (0.25)
GLMnet (α)	0.25	0.69 (0.25)
	0.5	0.69 (0.25)
	0.75	0.69 (0.25)
GP (Bessel)	$(1, 0.5, 2)$	1.12 (1.52)
	$(1, 1, 2)$	0.81 (0.81)
	$(1, 2, 2)$	0.74 (0.67)
(linear)		0.69 (0.25)
(poly)	$(1, 0.001, 1)$	0.69 (0.25)
	$(1, 0.01, 1)$	0.69 (0.25)
	$(1, 0.1, 1)$	0.69 (0.25)
	$(1, 1, 1)$	0.69 (0.25)
	$(3, 0.001, 1)$	0.66 (0.27)
	$(3, 0.01, 1)$	0.65 (0.29)
	$(3, 0.1, 1)$	0.83 (0.65)
	$(3, 1, 1)$	0.84 (0.68)
(RBF)		0.92 (0.90)
KNNreg (k)	1	0.53 (0.38)
	5	0.40 (0.31)
	10	0.48 (0.32)
	20	0.60 (0.33)
Kriging (trend)	$\mathbf{s}$	0.46 (0.34)
f_3, f_4 only	$\mathbf{s}, w_1$	0.41 (0.26)
f_5 only	$\mathbf{s}, w_1, w_2$	0.51 (0.15)
f_6 only	$\mathbf{s}, w_1, \ldots, w_6$	0.26 (0.15)
	none	0.49 (0.35)
f_3, f_4 only	w_1	0.42 (0.28)
f_5 only	w_1, w_2	0.52 (0.16)
f_6 only	$w_1, \ldots, w_6$	0.25 (0.14)

Algorithm	Parameters	Avg FVU
Mean		1.01 (0.01)
Polymars		0.73 (0.36)
Random Forest		0.45 (0.26)
SVM (Bessel; eps)	$(1, 1, 1)$	0.65 (0.27)
	$(1, 1, 2)$	0.57 (0.28)
	$(1, 2, 1)$	0.66 (0.27)
	$(1, 2, 2)$	0.59 (0.27)
(Bessel; nu)	$(1, 1, 1)$	0.67 (0.27)
	$(1, 1, 2)$	0.62 (0.27)
	$(1, 2, 1)$	0.68 (0.27)
	$(1, 2, 2)$	0.63 (0.27)
(linear; eps)		0.71 (0.25)
(linear; nu)		0.72 (0.28)
(poly; eps)	$(1, 0.001, 1)$	0.92 (0.12)
	$(1, 0.1, 1)$	0.71 (0.25)
	$(1, 1, 1)$	0.71 (0.25)
	$(3, 0.001, 1)$	0.85 (0.17)
	$(3, 0.1, 1)$	0.64 (0.28)
	$(3, 1, 1)$	0.83 (0.61)
(poly; nu)	$(1, 0.001, 1)$	0.97 (0.08)
	$(1, 0.1, 1)$	0.71 (0.25)
	$(1, 1, 1)$	0.72 (0.28)
	$(3, 0.001, 1)$	0.91 (0.14)
	$(3, 0.1, 1)$	0.66 (0.29)
	$(3, 1, 1)$	0.92 (0.71)
(RBF; eps)		0.48 (0.34)
(RBF; nu)		0.50 (0.32)
(tanh; eps)	$(0.01, 0.25)$	0.76 (0.21)
	$(0.01, 1)$	0.82 (0.18)
	$(0.005, 0.25)$	0.81 (0.19)
	$(0.005, 1)$	0.87 (0.16)
	$(0.002, 0.25)$	0.88 (0.15)
	$(0.002, 1)$	0.93 (0.11)
(tanh; nu)	$(0.01, 0.25)$	0.82 (0.19)
	$(0.01, 1)$	0.88 (0.16)
	$(0.005, 0.25)$	0.88 (0.16)
	$(0.005, 1)$	0.93 (0.12)
	$(0.002, 0.25)$	0.94 (0.11)
	$(0.002, 1)$	0.98 (0.07)
TPS (λ)	(GCV)	0.42 (0.36)
	0	0.52 (0.45)
	0.0001	0.44 (0.39)
	0.001	0.44 (0.35)
	0.01	0.54 (0.33)
	0.1	0.69 (0.28)

Table B.1: Simulation results for full library. For each algorithm, average Fraction of Variance Unexplained, (Avg FVU, standard deviation in parentheses) is the FVU averaged over all spatial processes, sample sizes, sampling designs, and noise condidtions. At each iteration, MSEs were calculated using all unsamped locations. Note that of the eight Kriging algorithms, only two were used to predict all spatial processes.

		$\widehat{\text{MSE}}$ ($\widehat{\text{FVU}}$)	
Algorithm	**S**	**CV**	**Full**
Super Learner			0.00 (0.03)
DSA			
	$\mathbf{x}, w$	0.13 (0.85)	0.12 (0.80)
	$\mathbf{x}, w_\ell$	0.09 (0.57)	0.07 (0.46)
GAM (degree)			
2	$\mathbf{x}, w$	0.09 (0.58)	0.07 (0.49)
	$\mathbf{x}, w_\ell$	0.08 (0.51)	0.07 (0.45)
3	$\mathbf{x}, w$	0.08 (0.54)	0.07 (0.43)
	$\mathbf{x}, w_\ell$	0.08 (0.51)	0.06 (0.41)
4	$\mathbf{x}, w$	0.08 (0.53)	0.06 (0.40)
	$\mathbf{x}, w_\ell$	0.08 (0.50)	0.06 (0.39)
GBM (degree)			
2	$\mathbf{x}, w$	0.07 (0.49)	0.05 (0.32)
	$\mathbf{x}, w_\ell$	0.07 (0.49)	0.05 (0.32)
4	$\mathbf{x}, w$	0.08 (0.50)	0.04 (0.29)
	$\mathbf{x}, w_\ell$	0.07 (0.49)	0.05 (0.32)
6	$\mathbf{x}, w$	0.07 (0.49)	0.04 (0.28)
	$\mathbf{x}, w_\ell$	0.07 (0.49)	0.04 (0.29)
GLM			
	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
GLMnet (α)			
0.25	$\mathbf{x}, w$	0.12 (0.79)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.55)	0.08 (0.50)
0.5	$\mathbf{x}, w$	0.11 (0.73)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
0.75	$\mathbf{x}, w$	0.11 (0.75)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
GP (Bessel)			
(1, 0.5, 2)	$\mathbf{x}, w$	0.13 (0.83)	0.04 (0.25)
	$\mathbf{x}, w_\ell$	0.16 (1.03)	0.03 (0.21)
(1, 1, 2)	$\mathbf{x}, w$	0.14 (0.90)	0.04 (0.27)
	$\mathbf{x}, w_\ell$	0.15 (0.97)	0.03 (0.22)
(1, 2, 2)	$\mathbf{x}, w$	0.16 (1.08)	0.04 (0.29)
	$\mathbf{x}, w_\ell$	0.17 (1.10)	0.04 (0.25)
GP (linear)			
	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
GP (RBF)			
	$\mathbf{x}, w$	0.22 (1.45)	0.02 (0.16)
	$\mathbf{x}, w_\ell$	0.22 (1.46)	0.01 (0.08)
GP (poly.)			
(1, 0.001, 1)	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
(1, 0.01, 1)	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
(1, 0.1, 1)	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)
(1, 1, 1)	$\mathbf{x}, w$	0.11 (0.74)	0.10 (0.67)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.50)

		$\widehat{\text{MSE}}$($\widehat{\text{FVU}}$)	
Algorithm	**S**	**CV**	**Full**
KNNreg (k)			
1	$\mathbf{x}$	0.17 (1.12)	0.00 (0.00)
	$\mathbf{x}, w$	0.11 (0.73)	0.00 (0.00)
5	$\mathbf{x}$	0.12 (0.76)	0.08 (0.52)
	$\mathbf{x}, w$	0.08 (0.55)	0.06 (0.38)
10	$\mathbf{x}$	0.11 (0.73)	0.09 (0.61)
	$\mathbf{x}, w$	0.08 (0.53)	0.07 (0.43)
20	$\mathbf{x}$	0.11 (0.72)	0.10 (0.66)
	$\mathbf{x}, w$	0.09 (0.56)	0.08 (0.50)
Kriging			
(OK)		0.11 (0.71)	0.00 (0.00)
	w	0.09 (0.56)	0.00 (0.00)
	w_ℓ	0.07 (0.46)	0.00 (0.00)
(UK)	$\mathbf{x}$	0.11 (0.72)	0.00 (0.00)
	$\mathbf{x}, w$	0.09 (0.60)	0.00 (0.00)
	$\mathbf{x}, w_\ell$	0.08 (0.51)	0.00 (0.00)
Mean		0.15 (1.00)	0.15 (1.00)
Polymars			
	$\mathbf{x}, w$	0.10 (0.63)	0.04 (0.27)
	$\mathbf{x}, w_\ell$	0.08 (0.56)	0.05 (0.36)
RF			
	$\mathbf{x}, w$	0.08 (0.50)	0.02 (0.11)
	$\mathbf{x}, w_\ell$	0.08 (0.50)	0.02 (0.12)
SVM (Bessel; eps)			
(1, 1, 2)	$\mathbf{x}, w$	0.09 (0.57)	0.06 (0.43)
	$\mathbf{x}, w_\ell$	0.08 (0.55)	0.06 (0.42)
(1, 2, 1)	$\mathbf{x}, w$	0.09 (0.56)	0.08 (0.51)
	$\mathbf{x}, w_\ell$	0.08 (0.52)	0.07 (0.46)
(1, 2, 2)	$\mathbf{x}, w$	0.09 (0.56)	0.07 (0.45)
	$\mathbf{x}, w_\ell$	0.08 (0.55)	0.07 (0.44)
SVM (Bessel; nu)			
(1, 1, 2)	$\mathbf{x}, w$	0.09 (0.57)	0.07 (0.48)
	$\mathbf{x}, w_\ell$	0.09 (0.61)	0.07 (0.46)
(1, 2, 1)	$\mathbf{x}, w$	0.1 (0.64)	0.08 (0.56)
	$\mathbf{x}, w_\ell$	0.08 (0.56)	0.07 (0.48)
(1, 2, 2)	$\mathbf{x}, w$	0.09 (0.59)	0.07 (0.49)
	$\mathbf{x}, w_\ell$	0.09 (0.58)	0.07 (0.47)
SVM (poly, eps)			
(1, 0.001, 1)	$\mathbf{x}, w$	0.15 (0.97)	0.14 (0.96)
	$\mathbf{x}, w_\ell$	0.14 (0.94)	0.14 (0.92)
(1, 0.1, 1)	$\mathbf{x}, w$	0.12 (0.78)	0.10 (0.69)
	$\mathbf{x}, w_\ell$	0.08 (0.52)	0.08 (0.50)
(1, 1, 1)	$\mathbf{x}, w$	0.12 (0.78)	0.11 (0.69)
	$\mathbf{x}, w_\ell$	0.08 (0.53)	0.08 (0.50)
(3, 0.001, 1)	$\mathbf{x}, w$	0.14 (0.92)	0.13 (0.89)
	$\mathbf{x}, w_\ell$	0.12 (0.81)	0.12 (0.78)
SVM (poly, nu)			
(1, 0.001, 1)	$\mathbf{x}, w$	0.15 (0.98)	0.15 (0.97)
	$\mathbf{x}, w_\ell$	0.15 (0.97)	0.14 (0.95)
(1, 0.1, 1)	$\mathbf{x}, w$	0.11 (0.73)	0.11 (0.70)
	$\mathbf{x}, w_\ell$	0.08 (0.55)	0.08 (0.53)
(1, 1, 1)	$\mathbf{x}, w$	0.11 (0.74)	0.11 (0.70)
	$\mathbf{x}, w_\ell$	0.08 (0.54)	0.08 (0.52)
(3, 0.001, 1)	$\mathbf{x}, w$	0.14 (0.94)	0.14 (0.92)
	$\mathbf{x}, w_\ell$	0.13 (0.89)	0.13 (0.87)
TPS (GCV)	$\mathbf{x}$	0.11 (0.71)	0.08 (0.53)

Table B.2: Lake acidity results for full library. **S** denotes the variable subset each algorithm was given. Risks were estimated via cross-validation (CV) or on the full dataset (Full).

Appendix C

SP Variance Estimation - Approximating the Variance of Variance Estimators

We illustrate how we derived (3.3) by first deriving an approximation of the true variance of a sieve element when this element is in the plateau, and then using a working model on this approximation. We use notation previously defined above in section 3.1.

Consider the variance of an estimator of the form (3.2) with arbitrary $\boldsymbol{\tau} \in T_0$. Note that if $\Omega_{ij} \cap \Omega_{k\ell} = \emptyset$, then $P_0^n D_{ijk\ell} = (P_0^n D_{ij})(P_0^n D_{k\ell})$. We make the following assumption on the independence structure of the observed data:

$$\begin{aligned}
\mathbb{E}_0\left[\sigma_n^4(\boldsymbol{\tau})\right] &= \frac{1}{n^2}\sum_{i,j,k,\ell}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ijk\ell} \\
&= \frac{1}{n^2}\sum_{i,j,k,\ell}\gamma(i,j,k,\ell)\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ij}P_0^n D_{k\ell} \\
&\quad + \frac{1}{n^2}\sum_{i,j,k,\ell}[1-\gamma(i,j,k,\ell)]\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ijk\ell} \qquad \text{(C.1)} \\
&= \frac{1}{n^2}\sum_{i,j,k,\ell}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ij}P_0^n D_{k\ell} + o(1).
\end{aligned}$$

Specifically, we assume the term (C.1) is negligible. Given this assumption, we can approximate the variance of (3.2) as

$$\begin{aligned}
\text{Var}\left[\sigma_n^2(\boldsymbol{\tau})\right] &= \mathbb{E}_0\left\{\sigma_n^2(\boldsymbol{\tau})\right\}^2 - \left\{\mathbb{E}_0\sigma_n^2(\boldsymbol{\tau})\right\}^2 \\
&\approx \frac{1}{n^2}\left\{\sum_{i,j,k,\ell}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ijk\ell} - \sum_{i,j,k,\ell}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)P_0^n D_{ij}P_0^n D_{k\ell}\right\} \\
&= \frac{1}{n^2}\sum_{i,j,k,\ell}\left\{1-\gamma(i,j,k,\ell)\right\}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)\left\{P_0^n D_{ijk\ell} - P_0^n D_{ij}P_0^n D_{k\ell}\right\}.
\end{aligned}$$

Let $\rho(i,j,k,l) = P_0^n D_{ijk\ell} - P_0^n D_{ij}P_0^n D_{k\ell}$. Suppose for the sake of illustration we define the following working models for ρ: $\alpha^2 = P_0^n D_i^2$, common across i; $\rho_1 = P_0^n D_{ij}$, common

across $\{(i,j) : \delta_{\boldsymbol{\tau}}(i,j) = 1, i \neq j\}$, and $\rho_2 = P_0^n D_{ijk\ell}$, common across $\{(i,j,k,\ell) : \delta_{\boldsymbol{\tau}}(i,j) = \delta_{\boldsymbol{\tau}}(k,\ell) = 1,\ \gamma(i,j,k,\ell) = 0\}$. We could estimate these quantities by taking empirical averages across the relevant elements. An estimator of ρ_1 for an arbitrary $\boldsymbol{\tau}$ is given by $\rho_{1,n} = \{\sum_{i,j} \delta_{\boldsymbol{\tau}}(i,j) I(i \neq j) D_{ij}\}/\{\sum_{i,j} \delta_{\boldsymbol{\tau}}(i,j) I(i \neq j)\}$. Similarly, $\alpha_n^2 = 1/n \sum_i D_i^2$, and

$$\rho_{2,n} = \frac{\sum_{i,j,k,\ell} \delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)(1-\gamma(i,j,k,\ell))D_{ijk\ell}}{\sum_{i,j,k,\ell} \delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)\{1-\gamma(i,j,k,\ell)\}}.$$

This defines $\rho_n(i,j,k,\ell)$ for each (i,j,k,ℓ) as having three possible values: α_n^2, $\rho_{1,n}$ or $\rho_{2,n}$. The corresponding variance estimator using this working model is

$$\mathrm{Var}\left\{\sigma_n^2(\boldsymbol{\tau})\right\} = 1/n^2 \sum_{i,j,k,\ell} \{1-\gamma(i,j,k,\ell)\}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)\rho_n(i,j,k,\ell).$$

Assuming common ρ across all time points is in most cases inappropriate. However, it does show us that the variance of the sieve elements can be viewed as proportional to (3.3), $1/n^2 \sum_{i,j,k,\ell}\{1-\gamma(i,j,k,\ell)\}\delta_{\boldsymbol{\tau}}(i,j)\delta_{\boldsymbol{\tau}}(k,\ell)$. We have found using (3.3) works well in practice as both a basis for a weight in PAV algorithms and as a sieve ordering.

Appendix D

SP Variance Estimation - Code for Computing the Variance of Variance Estimator

Our estimator of the variance of variance estimators (3.3) requires computing a quadruple sum. For the SP simulation study, we needed to compute roughly $2^{14} \times 4 \times 6 \times 30,000 = 11,796,480,000$ of these, so speed was paramount. Below is a commented version of highly optimized C/C++ code to perform this task. It was written to take advantage of specific features of an Intel compiler, which, when utilized correctly, can produce machine-level instructions that run much faster than would otherwise be possible. It was also written to work extremely efficiently under relatively sparse dependence (less than 100 "neighbors" for any given data point) and modest sample sizes (no more than n = 2000 or so units). At these sample sizes and dependence extents, almost everything remains in cache. In fact, this code actually runs slower under multithreading. The code can be modified to work effectively with large sample sizes and more extensive dependence, but there will be a noticeable drop-off in speed when the data structures must be stored in RAM.

The version of code presented here is completely general: it does not assume dependence is ordered in any way. A specialized version of this algorithm that takes advantage of short cuts appropriate for time series, which we do not present here, can run orders of magnitude faster. For instance, the general version of this code takes about 3.5 minutes on a reasonably well-equipped laptop to run one kitchen sink variance estimation procedure's worth of quadruple sums with $n = 250$ and maximum allowable dependence lag of 30 (roughly 30,000 unique sums). The time series version of this code can run the same computations in about 11 seconds. By contrast, a straightforward algorithm in C using non-bitset data structures and making only limited use of symmetry can take on the order of seven days.

The comments are extensive and meant to serve as an informal tutorial for future readers. Statistics and biostatistics graduate students are often well-trained in algorithms and other aspects of basic computer science. It is rare, however, for us to have the opportunity to learn how to optimize code from an engineering perspective. It is not always necessary for us to use these techniques; many computational problems

we encounter can be 'parallelized away', for instance. In this case, it would have taken just as much effort to make the algorithm fast enough so that the simulation study could be run in parallel within reasonable constraints (50 to 70 parallel processes, in less than a month) as it did to make it runnable on a well-equipped desktop.

This code was compiled into an R package [R Development Core Team, 2012] built using Rcpp [rcp, a,b], where it can be called from within R using the function `bm_quads_smrt()`. It takes as an input argument an adjacency list. An adjacency list is essentially a sparse representation of an adjacency matrix. Consider a time series of length $n = 10$, with a constant temporal dependence lag of 2. An adjacency matrix representing this dependence structure would be the following[1].

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

An equivalent adjacency list eliminates redundancies.

1
1 2
1 2 3
2 3 4
3 4 5
4 5 6
5 6 7
6 7 8
7 8 9
8 9 10

D.0.1 Code

```
#include <iostream>
#include <vector>
```

[1] We are technically abusing the term "adjacency matrix" here. In graph theoretic notation, this adjacency matrix would imply a graph with self-loops. Here we are using it to express the presence or absence of nonzero covariances between observations.

```
3 #include <algorithm>
4 #include <bitset>
5 #include <stdint.h>
6 #include <stdio.h>
7 #include <x86intrin.h>
8 #include <Rcpp.h>
9
10 #define BITSET_BITS 2048
11 #define UNIONS_SIZE 200000
12
13 #define MIN(x, y) ((x) < (y) ? (x) : (y))
14 #define MAX(x, y) ((x) > (y) ? (x) : (y))
15 // "x < y ? x : y":"  ~ if (x < y) x; else y;
16
17 typedef uint8_t bs_t[BITSET_BITS/8];
18
19 bs_t initial_bs[BITSET_BITS];
20
21 /* The following means one copy per process, but probably not one process
       per thread. This is faster than the simplest alternative.
22 Defining these as globals is faster here, but might not be considered
      best practice by some.
23 Globals mean that space is already reserved for these data structures as
      a part of the binary. We are hard-coding our space allocation guesses,
       and this way, if things fail, they fail obviously and informatively.
24 /**/
25 uint32_t unions_i[UNIONS_SIZE];
26 uint32_t unions_j[UNIONS_SIZE];
27 uint64_t unions_firstCheck[UNIONS_SIZE];
28 uint64_t unions_lastCheck[UNIONS_SIZE];
29 bs_t unions_bs[UNIONS_SIZE];
30
31 using namespace Rcpp;
32
33 /* This is a way to erase a function call from the source. It's
      equivalent to commenting out every instance of Rprintf()
34 /**/
35 #define Rprintf(args...) // remove debugging statements from final
36
37 void bs_set(bs_t *bitset, size_t pos) {
38 /* PURPOSE: set a bit at position position, where that position
      represents whether that element is an element of the set.
39 The built-in C++ version is 10x slower than this.
40 bitset := binary array where each bit is an indicator function, i.e.
      bitset element i = 0 means that the corresponding element i is not a
      member of the set. It's only useful to use this construction when
      there are a constrained number of elements that can be part of the set
      .
41 size_t := int that is guaranteed to be large enough to represent the size
       of any object, but practically speaking, we can think of it as size_t
       ~ uint64_t (64 bit unsigned int).
```

```
42 Think of "pos" as "i" in the above explanation.
43 We make the assumption that a bit = 8 bytes. This almost surely true, but
       there is nothing preventing it from being otherwise ...
44 /**/
45   size_t byteOffset = pos / 8;
46   size_t bitPosition = pos % 8;
47   uint8_t *bytePtr = (uint8_t *)bitset; // treat bitset as pointer to
       bytes
48 /* We are operating on the global bitset in this case. Below, this is
       equivalent to
49      bytePtr[byteOffset] = bytePtr[byteOffset] | (1 << bitPosition);
50 We set a byte using a binary or. This is the correct way to do this in C.
       We could potentially do this twice as fast, but it's not worth it
       here, since this is only called once.
51 /**/
52   bytePtr[byteOffset] |= (1 << bitPosition);
53 }
54
55 #ifdef __AVX__
56
57 // This section is run if we have a 256-bit register (versus 128)
58 typedef __m256i ymm_t;
59 /* Above is a compiler-specific name for a 256-bit vector integer with a
       bad name, so we give it something better. These cannot be added, but
       we can check them for intersection. The advantage of this type is that
       we can combine two 256-bit integers in a single operation. It takes
       slightly longer to get things into and out of them, but once you have
       them set up, you can do parrallel math on them very quickly.
60 /**/
61
62 /* the largest single variable this processor can hold is 32 bytes
63 /**/
64 #define VECSIZE 32
65
66 /* "inline" - hint to the compiler to use this a bit like a Macro, i.e.
       don't compile this as a separate function, just insert this text.
67 'static inline' is a strong hint: the compiler usually gives an error
       message if it is not able to inline the specificed function. 'static'
       also allows the function to be compiled more efficiently, and tells
       the compiler this function won't ever be used outside this file,
68 /**/
69 static inline int bs_intersects(bs_t *in0, bs_t *in1, uint64_t firstCheck
       , uint64_t lastCheck) {
70 /* This returns 1 as soon as it finds an intersection, returns 0 once it'
       s checked everything.
71 /**/
72   uint8_t *b0 = (uint8_t *)in0; // for convenient, 1-based math, we cast.
73   uint8_t *b1 = (uint8_t *)in1;
74 /* The first time we run this, we're going from 0 to 2048 (that's what
       firstChceck and lastCheck are passed as). These check variables can
       save us a lot of time when we use this function to check for
```

```
    intersections between unions. Data being clumpy (e.g. time dependence,
     spatial clusters) is what this has been opimized for.
75 /**/
76   for (uint64_t i = firstCheck; i < lastCheck; i += VECSIZE) {
77 // load the whole bitset in 32-byte chunks
78     ymm_t vec0 = _mm256_loadu_si256((ymm_t *)&b0[i]);
79 /* The above function is what is called a vector intrinsic. If the CPU
    supports this operation, it will become a single assembly instruction.
     Some assembly operations are really slow. There's no fast way to do
    simulataneous division, for instance. Multiplication takes 4 cycles.
    Adding takes 1 cycle. Logical 'or' operations take one cycle, plus
    they operate on 256 bits simultaneously.
80 _mm256_testz_si256 := compiler built-in function that operates on the
    entire 256-bit integer at once, equivalent to
81         if (vec0 & vec1 == 0)
82 This is illegal syntax in C++, and would require operator overloading to
    work. We are checking 32 bytes at a time for at least 1 intersection.
83 /**/
84     if (! _mm256_testz_si256(vec0, *(ymm_t *)&b1[i])) return 1;
85   }
86   return 0;
87 }
88
89 /* This function will write 3 outputs: (1) bitset that represents the
    union of the two rows; and will write values to (2) firstCheck_ptr and
     (3) lastCheck_ptr
90 /**/
91 void bs_create_union(bs_t *in0, bs_t *in1, bs_t *out,
92          uint64_t *firstCheck_ptr, uint64_t *lastCheck_ptr) {
93 /* Note, we only call this function when there is an intersection. This
    has not been tested to make sure it will work otherwise. It is used
    like this:
94   bs_create_union(&initial_bs[i], &initial_bs[j], &unions_bs[u],
95           &unions_firstCheck[u], &unions_lastCheck[u]);
96 *in0, *in1 := pointers to the bitsets (w/ size 2048 bits) that are
    indicators of set membership whose union we are computing.
97 *out := pointer to a 2048 bit array, which should start with all zeros.
    We are pre-addressing the output here, which is faster than re-
    allocating over and over again.
98 *firstCheck_ptr, *lastCheck_ptr := pointers to store some output.
99 /**/
100
101   size_t bitsetBytes = BITSET_BITS / 8; // how many bytes are in a bitset
     ?
102
103 // casting, to make math with pointers easier (lets us count in bytes):
104   uint8_t *b0 = (uint8_t *)in0;
105   uint8_t *b1 = (uint8_t *)in1;
106   uint8_t *bits = (uint8_t *)out;
107
108   uint64_t foundFirst = 0; // changes once, when you find the first
```

```
109   uint64_t foundLast = 0;  // internal boolean flag only, does not set
110   // a return value
111
112 /* 'i' is not an int below because it turns out to be a little more
      efficient to have the size of the pointer (eg b0) be the same as the
      size of the index (i, in this case).
113 /**/
114   for (uint64_t i = 0; i < bitsetBytes; i += VECSIZE) {
115     ymm_t vec0 = _mm256_loadu_si256((ymm_t *)&b0[i]);
116     ymm_t vec1 = _mm256_loadu_si256((ymm_t *)&b1[i]);
117     ymm_t vecOut = _mm256_or_si256(vec0, vec1);
118 /* vecOut is the union of all the bits in vec0 and vec1. vec0 and vec1
      are 32-bit chunks of b0 and b1. If vec0 and vec1 are both all zeros,
      this means that these particular chunks of b0 and b1 have nothing
      adjacent to them. This means both b0 and b1 have no dependent
      observations in this 256-bit range.
119 /**/
120     int vecsIntersect = ! _mm256_testz_si256(vecOut, vecOut);
121     if (! foundFirst && vecsIntersect) {
122       // if nobody has had any 'friends' in any prior chunks, BUT
123       //  currently at least one of them now has a 'friend',
124       //  save this byte offset for future use.
125       foundFirst = 1;
126       *firstCheck_ptr = i;
127     }
128
129 /* FIXME: Logic error.  Uncomment line below and test for correctness in
      the case where nothing is set. The use-cases of this function thus far
       would prevent this function from ever encountering such a state, but
      this is an area where there is room for future improvement.
130 /**/
131 // if (foundLast && vecsIntersect) foundLast = 0;
132
133     if (foundFirst && ! foundLast && ! vecsIntersect) {
134 /* Translation of if statement: If you've already found a first, but not
      yet found a last, and currently neither has any friends in this chunk.
       NOTE: This is another spot where logic could be improved. Instead, we
       should use
135   if (foundFirst && ! vecsIntersect) {
136 /**/
137       foundLast = 1;
138       *lastCheck_ptr = i;
139     }
140     _mm256_storeu_si256((ymm_t *)&bits[i], vecOut);
141   }
142   if (! foundFirst) *firstCheck_ptr = bitsetBytes; // no intersection,
       shouldn't happen
143   if (! foundLast) *lastCheck_ptr = bitsetBytes; // expected if there's
       an intersection in last vec
144   Rprintf("firstCheck (%d), lastCheck (%d)\n", *firstCheck_ptr, *
       lastCheck_ptr);
```

```
145 }
146 #else // ! __AVX__, presume SSE4.1
147 typedef __m128i xmm_t;
148 #define VECSIZE 16
149
150
151 static inline int bs_intersects(bs_t *in0, bs_t *in1, uint64_t firstCheck
      , uint64_t lastCheck) {
152   uint8_t *b0 = (uint8_t *)in0;
153   uint8_t *b1 = (uint8_t *)in1;
154   for (uint64_t i = firstCheck; i < lastCheck; i += VECSIZE) {
155     ymm_t vec0 = _mm_loadu_si128((xmm_t *)&b0[i]);
156     ymm_t vec1 = _mm_loadu_si128((xmm_t *)&b1[i]);
157     if (! _mm_testz_si128(vec0, vec1)) return 1;
158   }
159   return 0;
160 }
161
162 void bs_create_union(bs_t *in0, bs_t *in1, bs_t *out, uint64_t *
     firstCheck_ptr, uint64_t *lastCheck_ptr) {
163   size_t bitsetBytes = BITSET_BITS / 8;
164   uint8_t *b0 = (uint8_t *)in0;
165   uint8_t *b1 = (uint8_t *)in1;
166   uint8_t *bits = (uint8_t *)out;
167   uint64_t foundFirst = 0;
168   uint64_t foundLast = 0;
169   for (uint64_t i = 0; i < bitsetBytes; i += VECSIZE) {
170     xmm_t vec0 = _mm_loadu_si128((xmm_t *)&b0[i]);
171     xmm_t vec1 = _mm_loadu_si128((xmm_t *)&b1[i]);
172     xmm_t vecOut = _mm_or_si128(vec0, vec1);
173     int vecsIntersect = ! _mm256_testz_si256(vecOut, vecOut);
174     if (! foundFirst && vecsIntersect) {
175       foundFirst = 1;
176       *firstCheck_ptr = i;
177     }
178     // FIXME: logic error for foundLast, see above
179     if (foundFirst && ! foundLast && ! vecsIntersect) {
180       foundLast = 1;
181       *lastCheck_ptr = i;
182     }
183     _mm_storeu_si128((xmm_t *)&bits[i], vecOut);
184   }
185   if (! foundFirst) *firstCheck_ptr = bitsetBytes;  // no intersection,
      shouldn't happen
186   if (! foundLast) *lastCheck_ptr = bitsetBytes;    // expected if
      intersection in last vec
187   Rprintf("firstCheck (%d), lastCheck (%d)\n", *firstCheck_ptr, *
      lastCheck_ptr);
188 }
189 #endif
190
```

```
191 // set starting cycles to current 64-bit rdtsc value
192 #define RDTSC_START(cycles)                     \
193   do {                                          \
194     uint64_t cyc_high, cyc_low, cyc_combined;   \
195     asm volatile("cpuid\n"                      \
196           "rdtsc /* writes %0 and %1 */":       \
197           "=d" (cyc_high), /* %rdx written by rdtsc */ \
198           "=a" (cyc_low):  /* %rax written by rdtsc  */ \
199           /* no reads */:                       \
200           "%rbx", "%rcx" /* clobbered by cpuid */  \
201           );                                    \
202     cyc_combined = (cyc_high << 32) | cyc_low;  \
203     (cycles) = cyc_combined;                    \
204   } while (0)
205
206 // set final cycles to current 64-bit rdtsc value
207 #define RDTSC_FINAL(cycles)                     \
208   do {                                          \
209     uint64_t cyc_high, cyc_low, cyc_combined;   \
210     asm volatile("rdtscp\n"                     \
211           "movq %%rdx, %0\n"                    \
212           "movq %%rax, %1\n"                    \
213           "cpuid":                              \
214           "=r" (cyc_high), /* %rdx written by rdtsc */ \
215           "=r" (cyc_low):  /* %rax written by rdtsc  */ \
216           /* no reads */:                       \
217           "%rax", "%rbx", "%rcx", "%rdx" /* clobbered by cpuid */); \
218     cyc_combined = (cyc_high << 32) | cyc_low;  \
219     (cycles) = cyc_combined;                    \
220   } while(0)
221
222
223 // [[Rcpp::export]]
224 uint64_t bm_quads_smrt(List amat){
225 /* This is timing code below, used to measure exactly how many cycles
       something takes. Uncomment for useful information. (and uncomment
       below.)
226   uint64_t cyclesStart, cyclesFinal;
227   RDTSC_START(cyclesStart);
228
229   n is the sample size.
230   amat is a sparse representation of an adjacency matrix.
231 **/
232   int n = amat.length();
233   if (n >= BITSET_BITS) {
234     stop("List too large for bitmap");
235   }
236
237 /* clear the bitsets to prevent reuse of old values. this takes time, but
       we can get bugs with for-loops otherwise. Allocating a new one would
       take even longer.
```

```
238 /**/
239   memset(initial_bs, 0, sizeof(initial_bs));
240   memset(unions_bs, 0, sizeof(unions_bs));
241
242 /* The following code creates the matrix representation of the sparse
       array we passed it. Converting a list of lists to something where each
       row is a bs_t of fixed size of 2048 The entire matrix is an array of
       bs_ts.
243 /**/
244   for(int i=0; i<n; i++){
245     IntegerVector row = amat(i);
246     if (row.size() > BITSET_BITS) {
247       stop("Row too large for initial");
248     }
249     for(int j=0; j<row.size(); j++){
250       bs_set(&initial_bs[i], row[j]);
251     }
252   }
253
254 /* The following code will find all possible unions. Take every row and
       all the rows that occur at or below it (because we are clever and will
       be taking advantage of symmetry). This creates our omega_ij's for all
       omega_i, omega_j that have non-empty intersection. Note that we aren'
      t counting anything yet!
255 /**/
256   size_t unions_size = 0; // initially, we haven't saved any unions.
257 // i and j are row indices of our adjacency matrix that we just made,
      initial_bs[]
258   for(int i=0; i<n; i++){
259     for(int j=i; j<n; j++){
260
261       uint64_t firstCheck = 0; // check first byte of bitset
262       uint64_t lastCheck = BITSET_BITS / 8; // to last byte of bitset
263
264 // If at least one bit at the same position is set in both vectors,
265       if(bs_intersects(&initial_bs[i], &initial_bs[j], firstCheck,
      lastCheck)) {
266         size_t u = unions_size++; // increment unions_size up one.
267 /* Save 5 different values, all at location 'u'. We are basically
      creating the following struct below:
268
269     struct {
270       int i;
271       int j;
272       int firstCheck;
273       int lastCheck;
274       bs_t bitset;
275     } union_t;
276
277    Then we could have done something like
278
```

```
279   union_t allUnions[UNIONS_MAX]
280
281   This is an AOS, and gives us:
282
283  [ ] [ ] [ ] [ ] [ 256 bitset ...                ]
284  [ ] [ ] [ ] [ ] [ 256 bitset ...                ]
285  [ ] [ ] [ ] [ ] [ 256 bitset ...                ]
286  [ ] [ ] [ ] [ ] [ 256 bitset ...                ]
287
288  Instead, we are doing something like:
289
290   struct {
291     unions_i : [i] [i] ...
292     unions_j : [j] [j] ...
293     ...
294    unions_bs : [256 B] [256 B] ...
295   } allUnions_t;
296
297  This is faster to search than the more traditional AOS. The reason has
       to do with the memory layout. We always need to check elements that
       come first in the AOS, but we don't often have to check the bitset
       part. Our construction lets us keep things we will always need in
       cache. When we read unions_i[0], we are reading in 128 Bytes of i,
       each i = 8 Bytes. This means what we actually get is unions_i[0:15].
298  Key information re. memory management:
299  There exists a prefetcher, which will pre-request things if it sees you
       are reading forward. Prefetcher is fetching from RAM to L3 cache.
300  Suppose the CPU asks to load a register from an address. If the value
       of that address is in L1 cache, the answer arrives in the register in
       5 cycles. (L1 cache line = 64B)
301  If not, then L1 asks L2, and if it's in L2, it arrives in the register
       in 12 cycles. (L2 cache line = 64B)
302  If not, L2 asks L3, and 30 cycles later, the answer arrives in the
       register. L3 also has a cache line of 64B.
303  If L3 doesn't have it, then L3 fetches it from RAM, in 2 64B cache
       lines. L3 only sends the relevant cache line to L2. Approximately 128
       cycles later, it will show up in the register.
304  Our goal here: don't transfer it unless you need it, and make sure that
       everything you transfer is something you use.
305  Note that the prefetcher stops every 4096. This avoids triggering a TLB
       -miss (translation look-aside buffer miss). You could include software
       prefetch requests, but here it's not important for us to do that.
306  /**/
307         unions_i[u] = i;
308         unions_j[u] = j;
309         bs_create_union(&initial_bs[i], &initial_bs[j], &unions_bs[u],
310                         &unions_firstCheck[u], &unions_lastCheck[u]);
311       }
312     }
313   }
314
```

```
315  // Now we are counting unions!
316  uint64_t nQ = n;  // aaaa
317  for(size_t x=0; x<unions_size; x++){
318    uint64_t i = unions_i[x];  // first pair (i,j)
319    uint64_t j = unions_j[x];
320    uint64_t firstCheck_x = unions_firstCheck[x];
321    uint64_t lastCheck_x = unions_lastCheck[x];
322
323    if (i == j) {
324      // if x == y, already added 1
325      for (size_t y = x + 1; y < unions_size; y++) {
326        //        iterations++;
327
328        // only need to check for intersection in overlap of active vecs
329        uint64_t firstCheck = MAX(firstCheck_x, unions_firstCheck[y]);
330        uint64_t lastCheck = MIN(lastCheck_x, unions_lastCheck[y]);
331        if(! bs_intersects(&unions_bs[x], &unions_bs[y], firstCheck,
     lastCheck)) continue;
332        // intersections++;
333
334        uint64_t k = unions_i[y];  // second pair (k,l)
335        uint64_t L = unions_j[y];
336
337        if (k == L) {
338          nQ += 2;     // aabb, bbaa
339          Rprintf("2: k == L (%d, %d), (%d, %d)\n", i, j, k, L);
340        } else {
341          nQ += 4; // aabc, aacb, cbaa, bcaa
342          Rprintf("4: else (%d, %d), (%d, %d)\n", i, j, k, L);
343        }
344      }
345    } else { // i != j
346      // if y == x: i == k; j == L, k != L
347      nQ += 4; // abab, baab, baba, abba
348      Rprintf("4: y == x (%d, %d)\n", i, j);
349      for (size_t y = x + 1; y < unions_size; y++) {
350        //        iterations++;
351
352        // only need to check for intersection in overlap of active vecs
353          uint64_t firstCheck = MAX(unions_firstCheck[x],
     unions_firstCheck[y]);
354        uint64_t lastCheck = MIN(unions_lastCheck[x], unions_lastCheck[y
     ]);
355        if(! bs_intersects(&unions_bs[x], &unions_bs[y], firstCheck,
     lastCheck)) continue;
356        // intersections++;
357
358        uint64_t k = unions_i[y];  // second pair (k,l)
359        uint64_t L = unions_j[y];
360
361        if (k == L) {
```

```
362             nQ += 4; // abcc, bacc, ccba, ccab
363             Rprintf("4: k == L (%d, %d), (%d, %d)\n", i, j, k, L);
364           } else {
365             nQ += 8;
366             // abcd, bacd, bacd, badc
367             // cdab, cdba, dcba, dcab
368             // abac, abca, caab, caba
369             // abbc, abcb, cbab, cbba
370             // abcc, bacc, ccba, ccab
371             // abbb, bbab, bbba, babb
372             Rprintf("8: else (%d, %d), (%d, %d)\n", i, j, k, L);
373           }
374         }
375       }
376     }
377
378     return nQ;
379 }
```

Appendix E

SP Variance Estimation - Proof of Theorem 2

We start by proving the last result of the theorem. Define the process $(Z_n(\boldsymbol{\tau}, \theta) : (\boldsymbol{\tau}, \theta) \in \mathcal{F})$ by

$$Z_n(\boldsymbol{\tau}, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \Omega_{i,\boldsymbol{\tau}}} \{D_i(\theta) D_j(\theta) - P_0^n D_i D_j(\theta)\}.$$

We have

$$\begin{aligned} \sigma_n^2(\boldsymbol{\tau}) - \sigma_{0n}^2(\boldsymbol{\tau}) = \{Z_n(\boldsymbol{\tau}, \theta_n) - Z_n(\boldsymbol{\tau}, \theta_0)\} + Z_n(\boldsymbol{\tau}, \theta_0) + \\ \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \Omega_{i,\boldsymbol{\tau}}} P_0^n \{D_i D_j(\theta_n) - D_i D_j(\theta_0)\}. \end{aligned} \tag{E.1}$$

By condition (3.4), the last term in equation (E.1) converges to zero in probability uniformly in $\boldsymbol{\tau}$. We note that $Z_n(\boldsymbol{\tau}, \theta)$ can be represented as

$$\frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \Omega_{i,\boldsymbol{\tau}^u}} I(j \in \Omega_{i,\boldsymbol{\tau}}) \{D_i(\theta) D_j(\theta) - P_0^n D_i D_j(\theta)\},$$

showing $Z_n(\boldsymbol{\tau}, \theta)$ is a sum over $\mathcal{A} \equiv \{(i,j) : i = 1, \ldots, n, j \in \Omega_{i,\boldsymbol{\tau}^u}\}$. Let n^u be the size of $\mathcal{A}$. Then $Z_n(\boldsymbol{\tau}, \theta)$ can be represented as

$$\frac{1}{n^u} \sum_{k=1}^{n^u} f_k(\boldsymbol{\tau}, \theta) - P_0^n f_k(\boldsymbol{\tau}, \theta),$$

where $f_k(\boldsymbol{\tau}, \theta) = n^u D_{i(k)} D_{j(k)}(\theta)$ and $k \to (i(k), j(k))$ maps the index k into the corresponding index $(i(k), j(k))$ that makes up the index set $A = \{(i,j) : j \in \Omega_{i,\boldsymbol{\tau}^u}\}$. By the sparsity condition (3), linkage condition (4) and entropy condition (5), Theorem 3 in van der Laan [2014] proves that the process $(\sqrt{n} Z_n(\boldsymbol{\tau}, \theta) : (\boldsymbol{\tau}, \theta) \in \mathcal{F})$ is asymptotically equicontinuous, so that, in particular, the following holds: if $\| (\boldsymbol{\tau}_n, \theta_n) - (\boldsymbol{\tau}, \theta_0) \|_{\mathcal{F}} \to 0$ in probability, then $\sqrt{n}\{Z_n(\boldsymbol{\tau}_n, \theta_n) - Z_n(\boldsymbol{\tau}, \theta_0)\} = o_P(1)$, and

$$\sup_{(\boldsymbol{\tau}, \theta) \in \mathcal{F}} | Z_n(\boldsymbol{\tau}, \theta) | = O_P(1/\sqrt{n}).$$

As a consequence, we can conclude that if $\sup_{\boldsymbol{\tau}} \parallel (\boldsymbol{\tau}, \theta_n) - (\boldsymbol{\tau}, \theta_0) \parallel_{\mathcal{F}} \to 0$ in probability, then

$$\sqrt{n} \sup_{\boldsymbol{\tau}} |Z_n(\boldsymbol{\tau}, \theta_n) - Z_n(\boldsymbol{\tau}, \theta_0)| = o_P(1),$$

and we also have $\sup_{\boldsymbol{\tau}} |Z_n(\boldsymbol{\tau}, \theta_0)| = O_P(1/\sqrt{n})$. Condition (5) thus establishes that

$$\sqrt{n} \sup_{\boldsymbol{\tau}} |Z_n(\boldsymbol{\tau}, \theta_n) - Z_n(\boldsymbol{\tau}, \theta_0)| = o_P(1).$$

This proves

$$\sup_{\boldsymbol{\tau}} |\sigma_n^2(\boldsymbol{\tau}) - \sigma_{0n}^2(\boldsymbol{\tau})| = r(n) + O_P\left(1/\sqrt{n}\right).$$

We now prove the consistency of $\sigma_n^2(\boldsymbol{\tau}_n)$ as an estimator of σ_{0n}^2 when

$$\parallel (\theta_n, \boldsymbol{\tau}_n) - (\theta_0, \boldsymbol{\tau}_n^0) \parallel \to 0$$

and $d_n[(\boldsymbol{\tau}_n, \theta_n), (\boldsymbol{\tau}_n^0, \theta_0)] \to 0$ in probability for a $\boldsymbol{\tau}_n^0 \in T_0$ (we can act as if the latter holds with probability 1, by a standard simple argument). Firstly, we note that $\sigma_{0n}^2 = \sigma_{0n}^2(\boldsymbol{\tau}_n^0)$. Now we have

$$\begin{aligned} \sigma_n^2(\boldsymbol{\tau}_n) - \sigma_{0n}^2(\boldsymbol{\tau}_n^0) = {} & Z_n(\boldsymbol{\tau}_n, \theta_n) - Z_n(\boldsymbol{\tau}_n^0, \theta_0) + Z_n(\boldsymbol{\tau}_n^0, \theta_0) \\ & + \frac{1}{n} \sum_{(i,j) \in A} \{P_0^n f_{(i,j),\boldsymbol{\tau}_n,\theta_n} - P_0^n f_{(i,j),\boldsymbol{\tau}_n^0,\theta_0}\}. \end{aligned} \tag{E.2}$$

By the asymptotic equicontinuity of Z_n, we have

$$Z_n(\boldsymbol{\tau}_n, \theta_n) - Z_n(\boldsymbol{\tau}_n^0, \theta_0) = o_P(1/\sqrt{n}),$$

and $Z_n(\boldsymbol{\tau}_n^0, \theta_0) = O_P(1/\sqrt{n})$. The third term in equation (E.2) equals $d_n[(\boldsymbol{\tau}_n, \theta_n), (\boldsymbol{\tau}_n^0, \theta_0)]$, which is assumed to converge to zero in probability. Thus, this proves that $\sigma_n^2(\boldsymbol{\tau}_n) - \sigma_{0n}^2 \to 0$ in probability. □

Appendix F

Ventilation and Illness Absence - Table

Table F.1: Estimated mean IA counts when $\overline{V}(t)$ failed to reach v L/s/p, and associated 0.95 CIs.

v (L/s/p)	$\psi_n(v)$	ordering	type	σ_n^2	0.95 CI
3.41	0.721	none	iid	10.940	(0.667, 0.775)
		complexity	mode	20.826	(0.647, 0.795)
		n. nonzeros	mode	20.989	(0.647, 0.795)
		n. nonzeros	plateau	21.003	(0.647, 0.795)
		complexity	plateau	21.266	(0.646, 0.796)
		L_1 fit	mode	21.321	(0.646, 0.796)
		L_1 fit	plateau	21.328	(0.646, 0.796)
		L_1 fit	sample	21.873	(0.645, 0.797)
		n. nonzeros	sample	22.436	(0.644, 0.798)
		complexity	sample	24.257	(0.641, 0.801)
4.64	0.703	none	iid	5.359	(0.666, 0.741)
		complexity	mode	12.974	(0.645, 0.762)
		complexity	plateau	12.991	(0.645, 0.762)
		n. nonzeros	plateau	13.108	(0.645, 0.762)
		n. nonzeros	mode	13.115	(0.645, 0.762)
		L_1 fit	mode	13.173	(0.645, 0.762)
		L_1 fit	plateau	13.189	(0.645, 0.762)
		n. nonzeros	sample	13.403	(0.644, 0.763)
		L_1 fit	sample	13.412	(0.644, 0.763)
		complexity	sample	13.444	(0.644, 0.763)
5.87	0.683	none	iid	2.868	(0.656, 0.711)
		L_1 fit	plateau	6.026	(0.644, 0.723)
		L_1 fit	mode	6.164	(0.643, 0.723)
		n. nonzeros	mode	6.494	(0.642, 0.724)
		L_1 fit	sample	6.499	(0.642, 0.725)
		complexity	mode	6.538	(0.642, 0.725)
		n. nonzeros	plateau	6.832	(0.641, 0.726)
		n. nonzeros	sample	6.917	(0.641, 0.726)
		complexity	plateau	7.694	(0.638, 0.728)
		complexity	sample	7.842	(0.638, 0.729)
7.10	0.671	none	iid	2.033	(0.648, 0.694)
		n. nonzeros	plateau	4.791	(0.636, 0.706)

Continued on next page

Table F.1 – *Continued from previous page*

v (L/s/p)	$\psi_n(v)$	ordering	type	σ_n^2	0.95 CI
		n. nonzeros	sample	4.920	(0.635, 0.707)
		L_1 fit	mode	5.195	(0.634, 0.708)
		n. nonzeros	mode	5.270	(0.634, 0.708)
		complexity	mode	5.332	(0.634, 0.708)
		L_1 fit	plateau	6.049	(0.631, 0.711)
		complexity	plateau	6.654	(0.629, 0.713)
		L_1 fit	sample	6.856	(0.629, 0.713)
		complexity	sample	7.966	(0.625, 0.717)
8.33	0.659	none	iid	1.503	(0.639, 0.679)
		n. nonzeros	plateau	3.567	(0.628, 0.689)
		L_1 fit	plateau	3.601	(0.628, 0.69)
		n. nonzeros	sample	3.625	(0.628, 0.69)
		L_1 fit	sample	3.752	(0.628, 0.69)
		L_1 fit	mode	3.983	(0.627, 0.691)
		n. nonzeros	mode	4.015	(0.626, 0.691)
		complexity	mode	4.099	(0.626, 0.692)
		complexity	plateau	5.376	(0.621, 0.696)
		complexity	sample	5.446	(0.621, 0.697)
9.56	0.652	none	iid	1.249	(0.634, 0.67)
		L_1 fit	mode	3.732	(0.621, 0.683)
		nonzeros	mode	3.763	(0.62, 0.683)
		complexity	mode	3.844	(0.62, 0.684)
		L_1 fit	plateau	4.429	(0.618, 0.686)
		L_1 fit	sample	4.796	(0.616, 0.687)
		n. nonzeros	plateau	4.872	(0.616, 0.688)
		n. nonzeros	sample	5.035	(0.616, 0.688)
		complexity	plateau	5.080	(0.615, 0.688)
		complexity	sample	5.665	(0.613, 0.69)
10.79	0.651	none	iid	1.071	(0.634, 0.667)
		L_1 fit	mode	3.427	(0.621, 0.681)
		n. nonzeros	mode	3.444	(0.621, 0.681)
		complexity	mode	3.508	(0.62, 0.681)
		L_1 fit	plateau	4.525	(0.616, 0.685)
		L_1 fit	sample	4.780	(0.615, 0.686)
		complexity	plateau	4.787	(0.615, 0.686)
		n. nonzeros	plateau	5.182	(0.614, 0.687)
		complexity	sample	5.543	(0.613, 0.689)
		n. nonzeros	sample	5.675	(0.612, 0.689)
12.02	0.647	none	iid	0.974	(0.631, 0.663)
		L_1 fit	mode	3.283	(0.618, 0.677)
		n. nonzeros	mode	3.286	(0.618, 0.677)
		complexity	mode	3.354	(0.618, 0.677)
		L_1 fit	plateau	4.414	(0.613, 0.681)
		n. nonzeros	plateau	4.447	(0.613, 0.681)
		L_1 fit	sample	4.555	(0.613, 0.682)
		n. nonzeros	sample	4.586	(0.613, 0.682)
		complexity	plateau	4.686	(0.612, 0.682)
		complexity	sample	5.293	(0.61, 0.684)
13.25	0.646	none	iid	0.908	(0.631, 0.662)

Continued on next page

Table F.1 – *Continued from previous page*

v (L/s/p)	$\psi_n(v)$	ordering	type	σ_n^2	0.95 CI
		L_1 fit	mode	3.156	(0.617, 0.675)
		n. nonzeros	mode	3.160	(0.617, 0.675)
		complexity	mode	3.229	(0.617, 0.675)
		L_1 fit	plateau	4.271	(0.613, 0.68)
		L_1 fit	sample	4.464	(0.612, 0.68)
		complexity	plateau	4.528	(0.612, 0.681)
		n. nonzeros	plateau	4.932	(0.61, 0.682)
		complexity	sample	4.999	(0.61, 0.682)
		n. nonzeros	sample	5.266	(0.609, 0.683)
14.48	0.644	none	iid	0.868	(0.629, 0.659)
		L_1 fit	mode	3.094	(0.616, 0.673)
		n. nonzeros	mode	3.113	(0.616, 0.673)
		complexity	mode	3.165	(0.615, 0.673)
		L_1 fit	plateau	4.211	(0.611, 0.677)
		L_1 fit	sample	4.293	(0.611, 0.678)
		complexity	plateau	4.472	(0.61, 0.678)
		n. nonzeros	plateau	4.881	(0.608, 0.68)
		complexity	sample	4.888	(0.608, 0.68)
		n. nonzeros	sample	5.164	(0.607, 0.681)
15.71	0.643	none	iid	0.839	(0.628, 0.658)
		L_1 fit	mode	3.028	(0.615, 0.671)
		n. nonzeros	mode	3.047	(0.615, 0.671)
		complexity	mode	3.099	(0.615, 0.672)
		n. nonzeros	plateau	4.172	(0.61, 0.676)
		n. nonzeros	sample	4.388	(0.609, 0.677)
		complexity	plateau	4.398	(0.609, 0.677)
		complexity	sample	4.773	(0.608, 0.679)
		L_1 fit	plateau	4.829	(0.608, 0.679)
		L_1 fit	sample	5.188	(0.606, 0.68)
25.69	0.642	none	iid	0.759	(0.628, 0.657)
		n. nonzeros	mode	2.777	(0.615, 0.669)
		L_1 fit	mode	2.777	(0.615, 0.669)
		complexity	mode	2.836	(0.615, 0.67)
		L_1 fit	plateau	3.796	(0.611, 0.674)
		n. nonzeros	plateau	3.805	(0.611, 0.674)
		L_1 fit	sample	3.833	(0.611, 0.674)
		complexity	plateau	4.018	(0.61, 0.675)
		n. nonzeros	sample	4.025	(0.61, 0.675)
		complexity	sample	4.271	(0.609, 0.676)