Sensor Fusion Frameworks for Nowcasting

Maria Jahja

Department of Statistics and Data Science
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Support: This research was supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1745016, and by a fellowship from the Center for Machine Learning and Health at Carnegie Mellon University.

Keywords: nowcasting, sensor fusion, Kalman filter, constrained regression, digital surveillance, deconvolution, reconvolution, influenza, COVID-19

Abstract

A fundamental task in many online time series settings is to estimate the finalized value of a signal that will only be fully observed at a later time. The goal in nowcasting is to produce such estimates using contemporaneous information; this differs from the task of forecasting, which learns from past data to predict future values. In this thesis, we study sensor fusion (SF), a sequential nowcasting framework derived from a process-agnostic Kalman filter (KF), and detail two (mathematically equivalent) reformulations: first to the standard KF itself via an augmented measurement space, and then to an equality-constrained regression problem. We leverage these equivalences to port several established ideas (e.g., regularization schemes) in regression to dynamical systems.

In settings where only convolved outcomes of the signal can be observed, several new challenges arise: (i) deconvolution to infer the latent state, (ii) subsequent uncertainty propagation through SF, and (iii) reconvolution frameworks to evaluate performance. Towards solving these challenges, we introduce new methodology to perform and evaluate real-time nowcasting by deconvolution with specialized regularization techniques, which can prepend the SF framework. We motivate our work throughout by applications to track disease activity of influenza and COVID-19 in the United States.

Contents

Abstract **v**

List of Figures **ix**

List of Algorithms **xii**

Acknowledgments **xiv**

1 Introduction **1**

1.1 Motivation 1

1.2 Overview of contributions 3

2 Kalman Filter, Sensor Fusion, and Constrained Regression **9**

2.1 Preliminaries 9

2.1.1 Kalman filter (KF) 10

2.1.2 Sensor fusion (SF) 10

2.1.3 Related work 12

2.2 Equivalence between KF and SF 12

2.3 Equivalence between SF and regression 14

2.4 Nowcasting influenza activity in the US 15

2.5 Interpreting the constraints 16

2.6 Influenza nowcasting experiments 18

2.6.1 Setup 18

2.6.2 Results and interpretations 19

2.7 Discussion 20

3 Nowcasting Convolved Signals **22**

3.1 Introduction 22

3.2 COVID-19 background and motivation 24

3.2.1 Surveillance during the pandemic 24

3.2.2 Confounding 26

3.3 Related work 27
3.4 Preliminaries 28
3.4.1 Auxiliary indicators 28
3.4.2 Problem setup 30
3.5 Retrospective deconvolution 31
3.5.1 Convolutional model 32
3.5.2 Estimating the delay distribution 32
3.5.3 Defining ground truth 36
3.6 Real-time deconvolution 38
3.6.1 Incorporating extra regularization 39
3.6.2 Adjusting the delay distribution for truncation 42
3.6.3 Shortening the deconvolution window 44
3.7 Leveraging auxiliary signals 46
3.7.1 Sensor models 46
3.7.2 Sensor missingness 48
3.7.3 Sensor fusion 49
3.8 Evaluation 50
3.8.1 Performance of sensors and sensor fusion 51
3.8.2 Relative performance of sensor fusion methods 52
3.8.3 Post hoc smoothing 53
3.8.4 Correlation to hospitalizations 54
3.9 Discussion 56

4 A Reconvolution Approach for Evaluation **60**
4.1 Introduction 60
4.2 Preliminaries 62
4.2.1 Dimensions of evaluation 62
4.2.2 Review of COVID-19 setup 63
4.2.3 Defining ground truth 65
4.2.4 Problem setup 67
4.3 Related work 69
4.4 Reconvolution to observables, Px 70
4.4.1 Monte Carlo deconvolution 70
4.4.2 Partial reconvolution 72
4.5 Introducing uncertainty in $y \mid Px$ 74
4.6 Evaluation 76
4.6.1 Results of Monte Carlo deconvolution 77
4.6.2 Performance of deconvolution methods 79
4.7 Discussion 81

5 Discussion **85**

5.1 Summary and remarks 85
5.2 Future directions 86

A Proofs and Additional Details for KF-SF **91**
A.1 Proof of Theorem 1 91
A.2 Derivation of (2.12) 92
A.3 Proof of Theorem 2 93
A.4 Further SF-regression equivalences 94
A.4.1 More regularization: covariance shrinkage 94
A.4.2 Less regularization: zero padding 95
A.5 Example of process model selection 96

B Supplementary Deconvolution Material **99**
B.1 ADMM for solving deconvolution problems 99
B.2 Additional evaluation results 100

C Additional Reconvolution Results **104**

Bibliography **107**

List of Figures

1.1 Overview of the basic sensor fusion nowcasting framework. 4

2.1 Simplified hierarchical model of flu nowcasting with 3 levels. 16
2.2 Top row: overlay of data sources, sensors, and nowcasts to the underlying flu target for Pennsylvania. Bottom row: mean absolute errors of nowcasts across five flu seasons. 19

3.1 Illustration of estimating latent COVID-19 infections from reported cases. 30
3.2 Top row: discrepancies in cumulative COVID-19 cases in the US across data reporting sources. Bottom row: inconsistencies in zero delay reporting behavior across US states. 35
3.3 Left: quantiles of estimated onset-to-report delay distributions. Right: overlay of estimated delay distributions for three sample nowcast dates. 37
3.4 Illustration of right truncation with a delay distribution of length 3. . 39
3.5 Comparison of boundary behavior for real-time deconvolution in New York. 40
3.6 Effect of the tapered smoothing penalty as a function of the corresponding tuning parameter γ. 41
3.7 Comparing deconvolution regularization approaches by mean absolute error. 42
3.8 Left: estimated delay distribution overlaid for all nowcast dates. Right: mean ℓ_1 distance to finalized estimate of the delay distribution, as a function of lag $k = t - s$. 46
3.9 Comparing window lengths used in regularized deconvolution by mean absolute error. 47
3.10 Proportion of observed (non-missing) values over the evaluation period from October 1, 2020 to June 1, 2021, and over all locations, as a function of lag $k = 1, \dots, 10$. 49
3.11 Comparing NTF to individual sensor models and the simple average sensor fusion method by MAE. 52

3.12 Comparing NTF to individual sensor models and the simple average sensor fusion method by relative ranks over common nowcast tasks. 53
3.13 Comparing sensor fusion methods by boxenplots (level 25%, 50%, and 75% quantiles) of nowcasting errors, stratified by small, medium, and large states. 54
3.14 Comparing sensor fusion methods by relative ranks over common nowcast tasks, and considering only lags 1–5. 55
3.15 Post hoc smoothing on nowcast curve. 56
3.16 Lagged and leading correlations of observed case reports and infection nowcasts to finalized hospitalization averaged over October 1, 2020 to May 1, 2021. 57
3.17 Lagged and leading correlations of observed case reports and infection nowcasts to finalized hospitalization over latency k. 58

4.1 Comparison of real-time case incidence rates to finalized 7-day average case incidence rates in New York. 66
4.2 Overview of approaches for the reconvolution problem. 67
4.3 Perturbed infection and case estimates for New York on February 5, 2021 found through Monte Carlo deconvolution. 73
4.4 Quantiles of case forecasts for all aheads in New York. 77
4.5 Left: Mean total log probability summed over all aheads $i = 1, \dots, d$, as a function of the lag $k = t - s$, across noise levels ν. Right: Mean log probability score as a function of the noise level ν. 79
4.6 Mean log probability score as a function of forecast aheads $i = 1, \dots, d$, at fixed lag k, across noise levels ν. 80
4.7 Distributional evaluation of deconvolution estimates as a function of the lag $k = t - s$. 82
4.8 Distributional evaluation of deconvolution estimates as a function of the forecast aheads $i = 1, \dots, d$. 83
4.9 Difference of log probability score between sensor fusion and tapered NTF, averaged over lags k and various aheads i. 84

A.1 Simple process model selection example: outputs from 5 candidate process models, over the last 50 time points. 97

B.1 As in Figure 3.11, but including Google-AA. 101
B.2 As in Figure 3.12, but including Google-AA. 101
B.3 As in Figure 3.13, but including claims-based signals. 102
B.4 As in Figure 3.14, but including claims-based signals. 102
B.5 As in Figure 3.11, but comparing post hoc smoothed sensor fusion to the original sensor fusion and NTF methods. 103

C.1 As in Figure 4.5, but for trend filtering deconvolution. 105
C.2 As in Figure 4.5, but for natural trend filtering deconvolution with a tapered penalty. 105
C.3 As in Figure 4.6, but for trend filtering deconvolution with $k = 2,\ 10$. 106
C.4 As in Figure 4.6, but for natural trend filtering deconvolution with a tapered penalty with $k = 2,\ 10$. 106

List of Algorithms

3.1 Delay distribution estimation, retrospective 36
3.2 Distribution estimation under sequential truncation 44
3.3 Delay distribution estimation in real-time 45

4.1 Monte Carlo deconvolution 72
4.2 Stochastic forecasts via residual sampling 78

5.1 Joint training of sensor fitting and fusion 89
5.2 Sensor boosting 90

Acknowledgments

It is difficult to adequately express my deep gratitude to those who have supported me throughout the course of this thesis. I would like to first thank my advisor, Ryan, for his careful guidance, superhuman patience, and thoughtful insights from start to finish. I attribute the lion's share of my academic and professional growth to him, and owe a great debt for his kind generosity, where at every meeting he was willing to share practical ideas, offer advice, or simply brainstorm. Ryan has been a shaping influence in the way I approach problems, and I feel incredibly fortunate to have learned from and worked with him.

Many thanks to my committee, Roni, Valérie, Larry, and James, for their genuine support and constructive feedback. I am especially grateful to Roni, who welcomed me into the Delphi group and was later willing to serve as my joint mentor. Through the years, Roni has provided invaluable advice on many occasions, and challenged me to think beyond the technical details and grasp the bigger perspective.

I would also like to express my thanks to the Delphi group, especially Logan Brooks, Aaron Rumack, Addison Hu, and Andrew Chin, who worked side-by-side with me through a pandemic. Even under the worst of circumstances, it was a pleasure to collaborate and work with them. In a practical sense, this thesis was enabled by Katie Mazaitis and Christy Melucci, who provided prompt computing and administrative support, and worked tirelessly through the pandemic to ensure I had the resources I needed. They, along with the many other students, staff, and professors in Delphi, have made this thesis possible.

It is not an exaggeration to say that my graduate career would not have begun if not for Laber Labs. I am deeply grateful to Eric Laber, who introduced me to research, motivated me to work towards bigger dreams, and imparted advice that I still use to this day. I would like to also thank James Gilman, Nick Meyer, Nick Kapur, and Marschall Furman, who were kind enough to welcome, teach, and encourage an inexperienced undergraduate.

Several close friendships have made my time here so enjoyable. I am lucky to have met Kayla Frisoli, who, in addition to being an inspiring mentor, showed me that there is so much more to life beyond work. I am also grateful to our book club: Xiao Hui Tan, Kate Lyons, and Alexa Walls, for their unwavering support and cheerful encouragement. I would like to extend thanks to my cohort, who have been with me from the beginning; a special thank you to Matteo Bonvini, Natalia Oliveira, and Riccardo Fogliato for their friendship and good memories. I am grateful to many other friends, both here at Carnegie Mellon and outside, who have left an indelible mark: Ilmun Kim, Heejong Bong, and Jinjin Tian, for motivational office chats; Shannon Gallagher, Ben LeRoy, Mike Stanley, Sasha Podkopaev, Abby Smith, Alden Green, YJ Choe, and Minji Jang, for warm-hearted encouragement; William Qi and Tyler Vuong, for the fun times; Ruth Lee and Jess Thomas, for a decade-long friendship that cannot be captured in words; and to many, many others in the Statistics and Machine Learning departments.

My last thanks goes to my parents, Iwan and Frida, and my siblings Davina, Paulina, and Jordan, for their unconditional love and support, which underlies everything I do.

Chapter 1

Introduction

1.1 Motivation

Nowcasting is the task of predicting the finalized value of a not-yet-fully-observed signal close to the present time. Differently to the more common task of forecasting, which seeks to infer the future from historical data, nowcasting aims to predict the present (or very recent past) through contemporaneous data. Recently, nowcasting approaches have grown in popularity as technological advances in digital surveillance infrastructure have enabled the collection of massive data in near-real-time. Such data allows us to make timely predictions, or *nowcasts*, of slow-measured but fast-changing signals, which have shown to be especially valuable in sectors of economics [e.g., Choi and Varian [2012], Curme et al. [2014], Preis et al. [2010]] and health [e.g., Brooks [2020], Farrow [2016], Ginsberg et al. [2009], Jahja et al. [2019], McIver and Brownstein [2014], Yang et al. [2015]].

From the very outset, our contributions were, and are, motivated by problems in predictive epidemiology. We are particularly focused on studying approaches to model disease spread *in real-time.* Models of this form, which necessarily perform nowcasting, are central to informing the public health response in a myriad of ways, including: creating early warning systems of potential surges; enabling proactive resource allocation; informing the timing and severity of intervention measures; and raising situational awareness among the broader public. These actions, in turn, can aid in curtailing the spread and intensity of infection levels, and ultimately diminish the population impact of a disease.

To give some background, epidemiologic models for disease spread generally fall into two camps: mechanistic and/or statistical. Mechanistic models, broadly speaking, are grounded in biological processes of spread, and directly integrate disease transmission dynamics. A prominent example is the compartmental SIR model [Kermack and McKendrick, 1927], which defines a system of differential equations that describe the flow of individuals between stages of disease susceptibility, infection, and recovery. There are many variations of SIR—a few common ones are SIS, SIRS, SEIRS—which introduce various compartment configurations (see Hethcote [2000] for an overview). Traditional mechanistic models are characterized by a handful of model parameters and are informed only by data on the disease incidence itself; this dependence on a single data stream makes such approaches widely applicable to many settings, but can lead to issues when reports are noisy, delayed, or otherwise unreliable.

Several recent mechanistic approaches move beyond this, and incorporate additional side information, e.g., Brooks [2020], Shaman and Kandula [2015], Yang et al. [2014], and show improvements in predictive accuracy for their settings. Generally, such methods also overlap with the latter camp of statistical approaches. Statistical models are empirically grounded and assimilate data with little (or no) formal knowledge of disease dynamics. In the most minimally informed case, statistical models treat the problem as a time series task, which draws solutions from classical regression to recurrent neural networks [e.g., Aiken et al. [2021], Brooks et al. [2015], Farrow [2016], Jahja et al. [2019], McIver and Brownstein [2014], Santillana et al. [2016], Viboud et al. [2014], Wu et al. [2018], Yang et al. [2015], Yuan et al. [2013]].

Such models have grown popular among the epidemiological now– and forecasting community, which have trended towards incorporating auxiliary covariate data into statistical (or mixed mechanistic-statistical) approaches. This shift is, in large part, due to the increasingly commonplace availability of *digital surveillance data*. While traditional epidemiologic surveillance data typically arrives after a waiting period (for example, the standard measure for seasonal influenza levels is first released after a 1 week delay), digital surveillance data correlated to the disease are available in (virtually) real-time, and carry information ahead of official reports. These data streams originate from various sources, roughly categorized by internet (e.g., web search volume for symptoms, relevant social media mentions, online surveys for self-reported cases), electronic health records (e.g., volume of insurance claims, rates of related doctor visits), consumer behavior (e.g., purchasing trends of medication or medical devices), among others.

Digital surveillance data, while a huge win towards reflecting current disease levels, is not perfect. Many (though not all) common data sources are subject to regular *revision* and *backfill*, where, due to the arrival of delayed information, the initial value is updated several times over the future. Critically, retrospective studies to evaluate a nowcasting model must be careful to train on *versioned* data, that is, the model should only access the version of preliminary data that was available at the time of prediction, and not the finalized values recorded after. When training on finalized data, models can be overconfident in their ability to produce accurate predictions for the true future (this is observed in McDonald et al. [2021]). Adjusting models to deal with provisional surveillance data—from both traditional and digital sources—is central to providing accurate real-time nowcasts (for instance, Brooks et al. [2018a] demonstrates the importance of this in the context of influenza).

1.2 Overview of contributions

Driven by these motivations, our thesis work builds on *sensor fusion*, a two-step statistical nowcasting framework pioneered by Farrow [2016]. Sensor fusion first models the auxiliary data inputs to produce *sensors* (eponymous to the title) of the target signal, which are subsequently combined (or "fused") to generate a final prediction. Figure 1.1 depicts this process in a more formal way, where sensor fusion is applied to produce a current-time prediction of a one-step-delayed signal. In a conceptual vein, the ideology underlying sensor fusion is much the same as in ensemble learning, wherein a diverse set of base learners, each contributing complementary information, are assimilated to produce more accurate and robust predictions. Indeed, we consider sensor fusion as an ensemble adapted for online settings, where the base learners are the sensor models, and the assimilation step is a constrained regression problem (whose exact form is studied in Chapter 2).

In following chapters, we propose extensions of the basic sensor fusion framework, broadly grouped in two parts, as follows:

- In the first part, we study the estimator used in the fusion step, and detail several mathematical equivalences between it, the Kalman filter, and regression. Importantly, these reformulations allow us to propose various extensions to the original sensor fusion framework, which can be used to improve modeling of noisy digital surveillance data. We then demonstrate this framework in one of our main applications: seasonal influenza nowcasting, and show that it attains state-of-the-art performance in real settings.
- Next, we describe how sensor fusion can be applied to estimate *infections*, which are an inherently latent signal that requires a additional, non-trivial modeling step prior to sensor fusion. Specifically, this extra step is to perform deconvolution to recover infections from observed case reports, assuming a convolutional relationship where each infection onset is eventually reported after some stochastic delay. Here, our main application is to nowcast symptomatic COVID-19 infections, and we describe, implement, and carry out extensive experiments for a non-mechanistic deconvolution framework. We then propose, implement, and show results for a reconvolution approach to perform distributional evaluation.

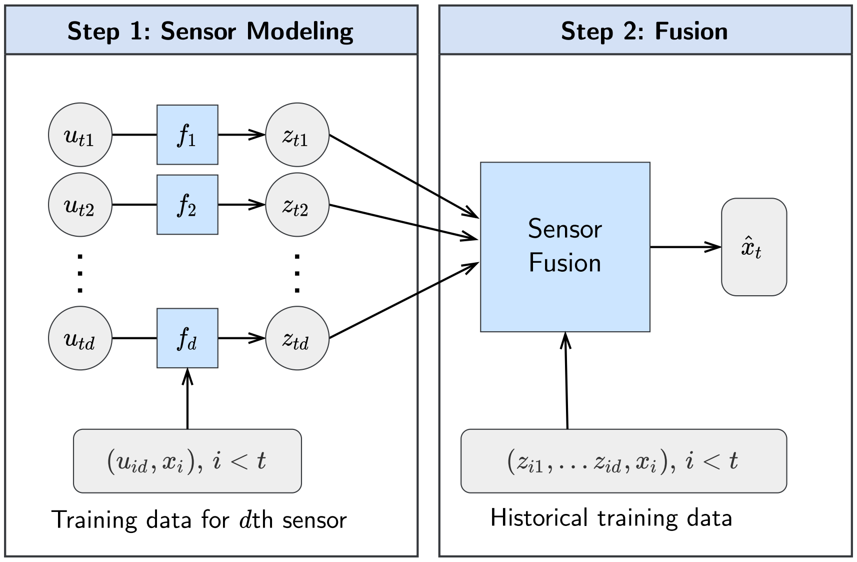


Figure 1.1: *Overview of the basic sensor fusion framework. At prediction time t, the goal is to estimate the state x_t, using d contemporaneous data sources, denoted $u_{t1}, \ldots, u_{td}$. Step 1 produces the latest sensors $z_{t1}, \ldots, z_{td}$, as the outputs of trained sensor models $f_1, \ldots, f_d$. The individual sensor models are trained to predict past states using past observations from their corresponding data source. The sensors (intermediate state estimates), along with historical sensors modeled at past prediction times, are fused in Step 2, to produce a final prediction $\hat{x}_t$.*

At each step, our work has been guided by real data. In particular, we demonstrate that our contributions create practical and implementable systems to nowcast influenza and COVID-19 disease activity (corresponding to the two parts, respectively). Below, we provide an overview and summary of these developments over the three following chapters, where the first chapter corresponds to the first part, and the second and third chapters cover the second part.

Chapter 2 Kalman Filter, Sensor Fusion, and Constrained Regression

> *This work was done in collaboration with David Farrow, Roni Rosenfeld, and Ryan Tibshirani, and contains content that appears in Jahja, Farrow, Rosenfeld, and Tibshirani [2019]. David Farrow motivated the original idea, which was then built on by Ryan Tibshirani and myself, and further developed in joint meetings with Roni Rosenfeld. I handled implementation of the ideas, adapting tools and code originally built by David Farrow and Logan Brooks.*

Within this chapter we investigate a form of sensor fusion (which we distinguish here as KF-SF) derived from the classic Kalman filter [Kalman, 1960], an longstanding algorithm for sequential estimation in linear dynamical systems. KF-SF is the result of a reformulated KF with infinite process noise (corresponding to the process model evolving states forward), which essentially places a flat prior on the state dynamics. This reformulation (which we again note is not our novel contribution, see, e.g., [Brown and Hwang, 2012, Farrow, 2016]) is particularly useful in settings where the process model is unknown or known to be misspecified, leading to poor quality estimates [Heffes, 1966]. However, KF-SF has otherwise been considered as a limiting case of the Kalman filter.

In this work we describe two new and interesting equivalences between KF-SF, the Kalman filter, and regression. First, we show that the Kalman filter can be viewed as a special case of KF-SF when we augment the measurement space; this is a somewhat surprising result, as previous derivations of KF-SF have always mentioned it as a special case of Kalman filter (one without the influence of a process model). This augmentation is quite useful; we can reintroduce state estimates from a candidate (or many candidate) process models, and fold in state dynamics without explicit dependence. Our second equivalence connects KF-SF (and hence the Kalman filter) to regression, and states that—given access to past state observations—KF-SF can be rewritten as a regression problem with equality constraints. This result opens the door for many extensions in Kalman filter methods by transferring established techniques in regression literature. To start, we describe extensions for regularization and gradient boosting.

Finally, we implement and evaluate KF-SF applied to influenza nowcasting in the United States, giving various interpretations of the constrained regression problem along the way, and show that this approach achieves state-of-the-art performance. These experiments build on the operational nowcasting system founded in Farrow [2016] using digital surveillance data and tools provided by the Carnegie Mellon Delphi Research Group [Farrow et al., 2015].

Chapter 3 Nowcasting Convolved Signals

This work was done in collaboration with Andrew Chin and Ryan Tibshirani, and contains content that appears in Jahja, Chin, and Tibshirani [2022]. The methodology was developed by Ryan Tibshirani and myself, and we thank Logan Brooks and Robert Tibshirani for helpful feedback. Andrew Chin and I implemented the ideas and experiments.

This chapter is dedicated to a deconvolution framework for nowcasting a hidden signal when given measurements of its convolved outcomes. Motivated by the COVID-19 pandemic, our underlying goal is to track incidence of COVID-19 *infections*, rather than observed cases or deaths (which do not reflect current disease spread). To give a backdrop to our approach: infections are commonly inferred through case reports, wherein an infected individual undergoes a period of symptom onset, testing, and processing, before their infection is published as recorded case. (Of course, there are qualifications at each stage, e.g., asymptomatic or untested infections; we provide a discussion of these important issues in a section of this chapter.) The time interval between symptom onset to publication is referred to as the *reporting delay*, and, given the distribution of this reporting delay, we can deconvolve case reports to infer infections.

Before describing the components of our framework, we remark that the Kalman filter can be (and traditionally is) applied to problems where the target state is always unobserved. However, in our particular setting we have special access to a direct byproduct of our target (case reports), and moreover, we have a model for the relationship between the two. In this sense, we can consider our signal as partially observed through convolved outcomes. Our approach uses this extra information to its advantage, and performs a three step process:

1. using public line list data, estimate the symptom-onset-to-report delay distribution;
2. perform deconvolution on case reports using the estimated delay distribution to get initial infection estimates;
3. apply a sensor fusion layer, which creates and assimilates sensors as in Chapter 2, but which are trained to predict estimated past infections, ultimately yielding the final infection estimates.

At each step, we provide and evaluate various solutions. In the initial step, we describe an adjusted procedure for estimating the (time-varying) reporting delay distribution in real-time. This adjustment is necessary due to data truncation, where the most recently onset infections have yet to be recorded in the line list dataset. Our solution performs an iterative adjustment similar in spirit to the Kaplan-Meier estimator [Kaplan and Meier, 1958], an established technique in survival analysis.

The bulk of our contributions are prompted by the subsequent deconvolution step. Deconvolution is a core topic in signal processing literature, and the direct solution is notoriously known to be ill-posed in the presence of noise or misspecification [Oppenheim and Verghese, 2017]. Given the limitations of real-world data, it is virtually impossible to guarantee the stability of the reported case signal, or exact estimation of reporting delay distribution. Moreover, real-time deconvolution is subject to the *right truncation* effect, which refers to the lack of future case information needed to fully infer the most recent infections falling on the right boundary (where the axis is time). To address both issues, we pose deconvolution as an optimization problem, and develop three forms of regularization to address instability. In stepwise fashion, we work through each regularization term, and demonstrate their utility towards improving nowcast performance.

Lastly, in Step 3 we return to sensor fusion, and train various digital surveillance sensors onto the estimated infections. These are subsequently fused to update the infection estimates *across all past*; that is, we leverage auxiliary information to improve estimation of our target infection signal both for present and historical values (which can help us assess the stability of our deconvolution method across time). Aside from this difference, the estimated infections are in essence treated as observed, and the sensor fusion framework described previously (depicted in Figure 1.1) can be applied, as-is. Notably, our work shows that sensor fusion, in addition to reducing latency in predictions, can be used to mitigate right truncation effects, and contributes significantly towards stabilizing the most recent estimates.

Throughout this work, we evaluate the performance of our methods against "finalized" infections, which is a ground truth signal found by regularized deconvolution long after potential revisions or right truncation effects can occur. However, evaluating our contributions against an observable signal is critical to understanding the fidelity of our nowcasts. This is the focus of our work in the next chapter.

Chapter 4 A Reconvolution Approach for Evaluation

> *This work was done in collaboration with Daniel McDonald, James Sharpnack, and Ryan Tibshirani. While all authors contributed towards the methodology, the final version used ideas proposed by Ryan Tibshirani and Daniel McDonald. Daniel McDonald and I implemented the ideas and experiments.*

This chapter serves as a sequel to the previous Chapter 3, and studies a reconvolution framework to evaluate the nowcasts by propagating the deconvolution solution forwards and measuring the error to future outcomes. Importantly, this work adds a distributional layer, which is central to validating the stability and trustworthiness of our nowcasts. Continuing in the COVID-19 setting, we propose a three step framework to generate distributional case forecasts from the estimated infection solutions, as follows:

1. generate perturbed samples of the infection solutions through a Monte Carlo deconvolution procedure;
2. apply *partial reconvolution* to propagate each sampled infection curve forward to an point estimate of future cases;
3. add appropriate residual noise to each case estimate to form a final density forecast.

We summarize each step in turn. First, we introduce stochasticity into the infection estimates (which would otherwise be considered fixed and known), by reconvolving the infection estimates forwards, adding training residual noise, and resolving the deconvolution problem. We repeat this procedure many times, and the resulting collection of infection solutions are considered to be draws from the solution distribution. To be straightforward, this procedure does not capture the *uncertainty* of our estimates, but rather their *stability*, which is still an important quantity to measure. In following experiments, we show that injecting stochasticity at this step is helpful to produce realistic forecast trajectories.

In the next step, we apply partial reconvolution, a technique to push forward infection estimates into an estimate of future cases without imposing any parametric structure or further assumptions apart from a locally constant reporting delay distribution. The need for partial reconvolution (over the standard "full" reconvolution) is due to missing estimates of infections in the future, which are necessary to calculate future cases. Partial reconvolution bypasses additional modeling of the infection (or case) curve, and propagates forward any available infection estimates with upweighted probability mass such that the full probability mass is carried forward. After passing our infection estimates through partial reconvolution, we have in hand a set of point case forecast trajectories.

In the third and final step, we construct a distribution for each forecast time by repeatedly adding sampled residual noise to the case forecasts. Importantly, to avoid overconfident forecast distributions, we construct the residual distribution using historical predictions, made out-of-sample. This procedure inherently assumes that the current nowcast task will have similar error as past nowcast tasks, but this can be a serious limitation in settings where the underlying target is highly non-stationary. We describe various extensions to create more sophisticated residual banks; for example, a weighting scheme, where higher sampling weights are assigned to residuals that come from nowcast tasks where the historical case signal exhibits similar behavior to current cases. There are many avenues towards identifying similar tasks, and we leave future study and implementation of these approaches as an open direction.

With this framework, we evaluate and compare the COVID-19 nowcasts made by three deconvolution methods, and find evidence supporting our original findings in Chapter 3 (which, recall, performed point evaluation to a finalized infection estimate). This is an encouraging result for both our deconvolution and reconvolution frameworks. Lastly, we provide a discussion on alternative approaches for evaluation that do not rely on reconvolution.

Chapter 2

Kalman Filter, Sensor Fusion, and Constrained Regression

This work was done in collaboration with David Farrow, Roni Rosenfeld, and Ryan Tibshirani, and contains content that appears in:

> Maria Jahja, David C. Farrow, Roni Rosenfeld, and Ryan J. Tibshirani. Kalman Filter, Sensor Fusion, and Constrained Regression: Equivalences and Insights. In: *Advances in Neural Information Processing Systems*, pages 13187–13196, 2019.

Python code for this work is available at:
http://github.com/mariajahja/kf-sf-flu-nowcasting.

2.1 Preliminaries

Let $x_t \in \mathbb{R}^k$, $t = 1, 2, 3, \ldots$ denote states and $z_t \in \mathbb{R}^d$, $t = 1, 2, 3, \ldots$ denote measurements evolving according to the time-invariant linear dynamical system:

$$x_t = Fx_{t-1} + \delta_t, \tag{2.1}$$

$$z_t = Hx_t + \epsilon_t. \tag{2.2}$$

We assume the noise terms δ_t, ϵ_t have mean zero and covariances $Q \in \mathbb{R}^{k\times k}$ and $R \in \mathbb{R}^{d\times d}$, respectively, for all t. Also, we assume that the initial state x_0 and all noise terms are mutually independent. We call (2.1) the process model and (2.2) the measurement model.

2.1.1 Kalman filter (KF)

The Kalman filter (KF) [Kalman, 1960] is a method for sequential estimation in the model (2.1), (2.2). Given past estimates $\hat{x}_1, \dots, \hat{x}_t$ and measurements $z_1, \dots, z_{t+1}$, we form an estimate $\hat{x}_{t+1}$ of the state x_{t+1} via

$$\bar{x}_{t+1} = F\hat{x}_t, \tag{2.3}$$
$$\hat{x}_{t+1} = \bar{x}_{t+1} + K_{t+1}(z_{t+1} - H\bar{x}_{t+1}), \tag{2.4}$$

where $K_{t+1} \in \mathbb{R}^{k \times d}$ is called the *Kalman gain* (at time $t+1$). It is itself updated sequentially, via

$$\bar{P}_{t+1} = FP_tF^T + Q, \tag{2.5}$$
$$K_{t+1} = \bar{P}_{t+1}H^T(H\bar{P}_{t+1}H^T + R)^{-1}, \tag{2.6}$$
$$P_{t+1} = (I - K_{t+1}H)\bar{P}_{t+1}. \tag{2.7}$$

where $P_{t+1} \in \mathbb{R}^{k \times k}$ denotes the state error covariance (at time $t+1$). The step (2.3) is often called the *predict* step: we form an intermediate estimate $\bar{x}_{t+1}$ of the state based on the process model and our estimate at the previous time point. The step (2.4) is often called the *update* step: we update our estimate $\hat{x}_{t+1}$ based on the measurement model and the measurement z_{t+1}.

Under the data model (2.1), (2.2) and the conditions on the noise stated above, the Kalman filter attains the optimal mean squared error $\mathbb{E}\|\hat{x}_t - x_t\|_2^2$ among all linear unbiased filters, at each $t = 1, 2, 3, \dots$. When the initial state x_0 and all noise terms are Gaussian, the Kalman filter estimates exactly reduce to the Bayes estimates $\hat{x}_t = \mathbb{E}(x_t | z_1, \dots, z_t)$, $t = 1, 2, 3, \dots$. Numerous important extensions have been proposed, e.g., the ensemble Kalman filter (EnKF) [Evensen, 1994, Houtekamer and Mitchell, 1998], which approximates the noise process covariance Q by a sample covariance in an ensemble of state predictions, as well as the extended Kalman filter (EKF) [Smith et al., 1962] and unscented Kalman filter (UKF) [Julier and Uhlmann, 1997], which both allow for nonlinearities in the process model. Particle filtering (PF) [Gordon et al., 1993] has more recently become a popular approach for modeling complex dynamics. PF adaptively approximates the posterior distribution, and in doing so, avoids the linear and Gaussian assumptions inherent to the KF. This flexibility comes at the cost of a greater computational burden.

2.1.2 Sensor fusion (SF)

If we let the noise covariance in the process model diverge to infinity, $Q \to \infty$ (To make this unambiguous, we may take, say, $Q = aI$ and let $a \to \infty$.) then the Kalman filter estimate in (2.3), (2.4) simplifies to

$$\hat{x}_{t+1} = (H^TR^{-1}H)^{-1}H^TR^{-1}z_{t+1}. \tag{2.8}$$

This can be verified by rewriting the Kalman gain as $K_{t+1} = (\bar{P}_{t+1}^{-1} + H^T R^{-1} H)^{-1} H^T R^{-1}$, and observing that $\bar{P}_{t+1}^{-1} \to 0$ as $Q \to \infty$. Alternatively, we can verify this by specializing to the case of Gaussian noise: as $\mathrm{tr}(Q) \to \infty$, we approach a flat prior, and the Kalman filter (Bayes estimator) just maximizes the likelihood of $z_{t+1}|x_{t+1}$. From the measurement model (2.2) (assuming Gaussian noise), this is a weighted regression of z_{t+1} on the measurement map H, precisely as in (2.8).

We will call (2.8) the *sensor fusion* (SF) estimate (at time $t+1$). We note that "sensor fusion" is typically used as a generic term, similar to "data assimilation"; we use it to specifically describe the estimate in (2.8) to distinguish it from the KF. This is useful when we describe the equivalences in Sections 2.2 and 2.3. In this setting, we will also refer to the measurements as *sensors*.

Summary of contributions

As defined, sensor fusion is a special case of the Kalman filter when there is infinite process noise; said differently, it is a special case of the Kalman filter when there is no process model at all. Thus, looking at (2.8), the state dynamics have apparently been completely lost. Perhaps surprisingly, as we will show shortly, these dynamics can be exactly recovered by augmenting the measurement vector z_{t+1} with the KF intermediate prediction $\bar{x}_{t+1} = F\hat{x}_t$ in (2.3) (and adjusting the map H and covariance R appropriately). We summarize this and our other contributions:

1. We show in Section 2.2 that, if we take the KF intermediate prediction $\bar{x}_{t+1}$ in (2.3), append it to the measurement vector z_{t+1}, and perform SF (2.8) (with an appropriately adjusted H, R), then the result is exactly the KF estimate (2.4).
2. We show in Section 2.3 that, if we are in a problem setting in which past states are observed (at some lag, which is the case in the flu nowcasting application), and we replace the noise covariance R from the measurement model by the empirical covariance on past data, then the sensor fusion estimate (2.8) can be written as $\hat{B}^T z_{t+1}$, where $\hat{B} \in \mathbb{R}^{d\times k}$ is a matrix of coefficients that solves a regression problem of the states on the measurements (using past data), subject to the equality constraint $H^T \hat{B} = I$.
3. We demonstrate the effectiveness of our new regression formulation of SF in Section 2.6 by describing an application of this methodology to nowcasting the incidence of weekly flu in the US. This achieves state-of-the art performance in this problem.
4. Later, in Section 5.2, we detail some extensions of the regression formulation of SF; they do not have direct equivalences to SF (or the KF), but are intuitive and extend dynamical systems modeling in new directions (e.g., using ℓ_1 penalization to perform a kind of process model selection).

2.1.3 Related work

The Kalman filter and its extensions, as previously referenced (EnKF, EKF, UKF), are the de facto standard in state estimation and tracking problems; the literature surrounding them is enormous and we cannot give a thorough treatment. Various authors have pointed out the simple fact that maximum likelihood estimate in (2.8), which we call sensor fusion, is the limit of the KF as the noise covariance in the process model approaches infinity (see, e.g., Chapter 5.9 of Brown and Hwang [2012]). We have not, however, seen any authors note that this static model can recover the KF by augmenting the measurement vector with the KF intermediate prediction (Theorem 1).

Along the lines of our second equivalence (Theorem 2), there is older work in the statistical calibration literature that studies the relationships between the regressions of y on x and x on y (for multivariate x, y, see Brown [1982]). This is somewhat related to our result, since we show that a *backwards* or *indirect* approach, which models $z_{t+1}|x_{t+1}$, is actually equivalent to a *forwards* or *direct* approach, which predicts x_{t+1} from z_{t+1} via regression. However, the details are quite different.

Finally, our SF methodology in the flu nowcasting application blends together individual predictors in a way that resembles *linear stacking* [Breiman, 1996, Wolpert, 1992]. In fact, one implication of our choice of measurement map H in the flu nowcasting problem, as well as the constraints in our regression formulation of SF, is that all regression weights must sum to 1, which is the standard in linear stacking as well. However, the equality constraints in our regression formulation are quite a bit more complex, and reflect aspects of the sensor hierarchy that linear stacking would not.

2.2 Equivalence between KF and SF

As already discussed, the sensor fusion estimate (2.8) is a limiting case of the Kalman filter (2.3), (2.4), and initially, it seems, one rather limited in scope: there is effectively no process model (as we have sent the process variance to infinity). However, as we show next, the KF is actually itself a special case of SF, when we augment the measurement vector by the KF intermediate predictions, and appropriately adjust the measurement map H and noise covariance R. The proof is elementary, a consequence of the Woodbury matrix and related manipulations. It is given in A.1 of the appendix.

Theorem 1. *At each time $t = 0, 1, 2, \dots$, suppose we augment our measurement vector by defining $\tilde{z}_{t+1} = (z_{t+1}, \bar{x}_{t+1}) \in \mathbb{R}^{d+k}$, where $\bar{x}_{t+1} = F\hat{x}_t$ is the KF intermediate prediction at time $t+1$. Suppose that we also augment our measurement map by defining $\tilde{H} \in \mathbb{R}^{(d+k)\times k}$ to be the rowwise concatenation of H and the identity matrix $I \in \mathbb{R}^{k\times k}$.*

Furthermore, suppose we define an augmented measurement noise covariance

$$\tilde{R}_{t+1} = \begin{bmatrix} R & 0 \\ 0 & \bar{P}_{t+1} \end{bmatrix}, \tag{2.9}$$

where $\bar{P}_{t+1}$ is the KF intermediate error covariance at time $t+1$(as in (2.5)). Then applying SF to the augmented system produces an estimate at $t+1$ that equals the KF estimate,

$$(\tilde{H}^T \tilde{R}_{t+1}^{-1} \tilde{H})^{-1} \tilde{H}^T \tilde{R}_{t+1}^{-1} \tilde{z}_{t+1} = \bar{x}_{t+1} + K_{t+1}(z_{t+1} - H\bar{x}_{t+1}), \tag{2.10}$$

where K_{t+1} is the Kalman gain at $t+1$ (as in (2.6)).

We give several remarks.

Remark 1. We can think of the last state estimate $\hat{x}_t$ in the theorem (which is propagated forward via $\bar{x}_{t+1} = F\hat{x}_t$) as the previous output from SF itself, when applied to the appropriate augmented system. More precisely, by induction, Theorem 1 says that iteratively applying SF to $\tilde{z}_{t+1}, \tilde{H}, \tilde{R}_{t+1}$ across times $t = 0, 1, 2, \ldots$, where each $\bar{x}_{t+1} = F\hat{x}_t$ is the intermediate prediction using the last SF estimate $\hat{x}_t$, produces a sequence $\hat{x}_{t+1}$, $t = 0, 1, 2, \ldots$ that matches the state estimates from the KF.

Remark 2. The result in Theorem 1 can be seen from a Bayesian perspective, as was pointed out by an anonymous reviewer of the work. When the initial state x_0 and all noise terms in (2.1), (2.2) are Gaussian, recall the KF reduces to the Bayes estimator. Here the posterior is the product of a Gaussian likelihood and Gaussian prior, and is thus itself Gaussian. (The proof of this standard fact uses similar arguments to the proof of Theorem 1.) Meanwhile, in augmented SF, we can view the Gaussian likelihood being maximized as the product of the Gaussian density of z_{t+1} and that of $\bar{x}_{t+1}$. This matches the posterior used by the KF, where the density of $\bar{x}_{t+1}$ plays the role of the prior in the KF. Therefore in each case, we are defining our estimate to be the mean of the same Gaussian distribution.

Remark 3. The equivalence between SF and KF can be extended beyond the case of linear process and linear measurement models. Given a nonlinear process map f and a nonlinear process model h, suppose we define $\bar{x}_{t+1} = f(\hat{x}_t)$, $F_{t+1} = Df(\hat{x}_t)$ (the Jacobian of f at $\hat{x}_t$), and $H_{t+1} = Dh(\bar{x}_{t+1})$ (the Jacobian of h at $\bar{x}_{t+1}$). Suppose we define the augmented measurement vector as

$$\tilde{z}_{t+1} = \big(z_{t+1} + H_{t+1}\bar{x}_{t+1} - h(\bar{x}_{t+1}),\, \bar{x}_{t+1}\big), \tag{2.11}$$

where we have offset the measurement z_{t+1} by the residual $H_{t+1}\bar{x}_{t+1} - h(\bar{x}_{t+1})$ from linearization. Suppose, as in the theorem, we define the augmented measurement map $\tilde{H}_{t+1} \in \mathbb{R}^{(d+k)\times k}$ to be the rowwise concatenation of H_{t+1} and $I \in \mathbb{R}^{k\times k}$, and define $\tilde{R}_{t+1} \in \mathbb{R}^{(d+k)\times(d+k)}$ as in (2.9), for $\bar{P}_{t+1}$ as in (2.5), but with F_{t+1}, H_{t+1} in place of F, H. In A.2, we prove that

$$(\tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{H}_{t+1})^{-1} \tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{z}_{t+1} = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big), \tag{2.12}$$

where K_{t+1} is as in (2.6), but with F_{t+1}, H_{t+1} in place of F, H. The right-hand side above is precisely the *extended* KF (EKF). The left-hand side is what we might call *extended* SF (ESF).

2.3 Equivalence between SF and regression

Suppose that in our linear dynamical system, at each time t, we observe the measurement z_t, make a prediction $\hat{x}_t$ for x_t, then later observe the state x_t itself. We may assume without a loss of generality that we observe the full past states x_i, $i = 1, \dots, t$ (if this is not the case, and we observe only some subset of the past, then the only changes to make in what follows are notational). This setup indeed describes the influenza nowcasting problem, which is one of our central motivating examples.

Assuming the measurement noise covariance R is unknown, we may use

$$\hat{R}_{t+1} = \frac{1}{t} \sum_{i=1}^{t} (z_i - Hx_i)(z_i - Hx_i)^T, \tag{2.13}$$

the empirical (uncentered) covariance based on past data, as an estimate. Under this choice, it turns out that sensor fusion (2.8) is exactly equivalent to a regression of states on measurements, subject to certain equality constraints. The proof is elementary, but requires detailed arguments. It is provided in A.3 of the appendix.

Theorem 2. *Let $\hat{R}_{t+1}$ be as in* (2.13) *(assumed to be invertible). Consider the SF prediction at time $t+1$, with $\hat{R}_{t+1}$ in place of R. Denote this by $\hat{x}_{t+1} = \hat{B}^T z_{t+1}$, where*

$$\hat{B}^T = (H^T \hat{R}_{t+1}^{-1} H)^{-1} H^T \hat{R}_{t+1}^{-1}$$

(and $H^T \hat{R}_{t+1}^{-1} H$ is assumed invertible). Each column of $\hat{B}$, denoted $\hat{b}_j \in \mathbb{R}^d$, $j = 1, \dots, k$, solves

$$\begin{aligned} \underset{b_j \in \mathbb{R}^d}{\text{minimize}} \quad & \sum_{i=1}^{t} (x_{ij} - b_j^T z_i)^2 \\ \text{subject to} \quad & H^T b_j = e_j, \end{aligned} \tag{2.14}$$

where $e_j \in \mathbb{R}^d$ is the jth standard basis vector (all 0s except for a 1 in the jth component).

As discussed earlier, the interpretation of $(H^T \hat{R}_{t+1}^{-1} H)^{-1} H^T \hat{R}_{t+1}^{-1} z_{t+1}$ as the coefficients from regressing z_{t+1}(the response) onto H (the covariates) is more or less immediate. Interpreting the same quantity as $\hat{B}^T z_{t+1} = (\hat{b}_1^T z_{t+1}, \dots, \hat{b}_k^T z_{t+1})$, the predictions from historically regressing x_i, $i = 1, \dots, t$ (the response) onto z_i, $i = 1, \dots, t$ (the covariates), however, is much less obvious. The latter is a *forwards* or *direct* regression approach to predicting x_{t+1}, whereas SF was originally defined via the *backwards* or *indirect* perspective inherent to the measurement model (2.2).

2.4 Nowcasting influenza activity in the US

To return to our motivating example, we now describe the background and setup for the influenza (or flu) nowcasting problem. The state variable of interest is the weekly percentage of weighted influenza-like illness (wILI), a measure of flu incidence provided by the Centers for Disease Control and Prevention (CDC), in each of the $k = 51$ US states (including DC). Because it takes time for the CDC to collect and compile this data, they release wILI values with a 1 week delay. Meanwhile, various proxies for the flu (i.e., data sources that are potentially correlated with flu incidence) are available in real time, e.g., web search volume for flu-related terms, site traffic metrics for flu-related pages, pharmaceutical sales for flu-related products, etc. We can hence train (using historical data) sensors to predict wILI, one from each data source, and plug them into sensor fusion (2.8) in order to "nowcast" the current flu incidence (that would otherwise remain unknown for another week).

Such a sensor fusion system for flu nowcasting, using $d = 308$ sensors (flu proxies), is described in Chapter 4 of Farrow [2016]. This is more than just a hypothetical system; it is fully operational, and run by the Carnegie Mellon DELPHI group to provide real-time nowcasts of flu incidence every week, in all US states, plus select regions, cities, and territories. (See https://delphi.midas.cs.cmu.edu). In addition to the surveillance sensors described above (search volume for flu terms, site traffic metrics for flu pages, etc.), the measurement vector in this nowcasting system also uses a sensor that is trained to make predictions of wILI using a seasonal autoregression with 3 lags (SAR3). From the KF-SF equivalence established in Section 2.2, we can think of this SAR3 sensor as serving the role of something like a process model, in the underlying dynamical system.

While wILI itself is available at the US state level, the data source used to train each sensor may only be available at coarser geographic resolution. Thus, importantly, each sensor outputs a prediction at a different geographic resolution (which reflects the resolution of its corresponding data source). As an example, the number of visits to flu-related CDC pages are available for each US state separately; so for each US state, we train a separate sensor to predict wILI from CDC site traffic. However, counts for Wikipedia page visits are only available nationally; so we train just one sensor to predict national wILI from Wikipedia page visits.

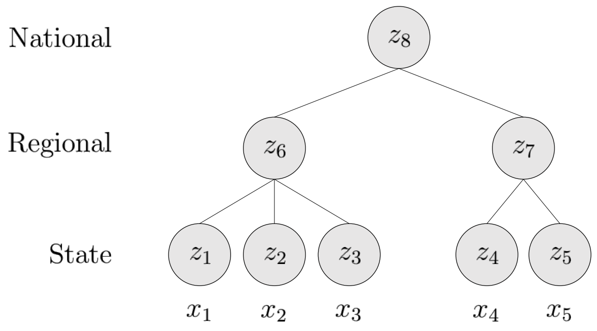


Figure 2.1: *Simplified version of the flu nowcasting problem, with $k = 5$ states and $d = 8$ sensors. We have a 3-level hierarchy, where x_1, x_2, x_3 are part of the first region and x_4, x_5 are part of the second. The national level is at the root. As for the sensors, we have one at each state, one at each region, and one at the national level. Assuming all states have equal populations, the sensor map H is*

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ {}^1/_3 & {}^1/_3 & {}^1/_3 & 0 & 0 \\ 0 & 0 & 0 & {}^1/_2 & {}^1/_2 \\ {}^1/_5 & {}^1/_5 & {}^1/_5 & {}^1/_5 & {}^1/_5 \end{bmatrix}.$$

Assuming unbiasedness of all the sensors, we construct the map H in (2.2) so that its rows reflect the geography of the sensors. For example, if a sensor is trained on data that is available at the ith US state, then its associated row in H is

$$(0, \dots \underset{\overset{\uparrow}{i}}{1}, \dots 0);$$

and if a sensor is trained on data from the aggregate of the first 3 US states, then its associated row is

$$(w_1, w_2, w_3, 0, \dots 0),$$

for weights $w_1, w_2, w_3 > 0$ such that $w_1 + w_2 + w_3 = 1$, based on relative state populations; and so on. Figure 2.1 illustrates the setup in a simple example.

2.5 Interpreting the constraints

At a high-level, the constraints in (2.14) encode information about the measurement model (2.2). They also provide some kind of implicit regularization. Interestingly, as we will see in the experiment discussed shortly, this can still be useful when used in addition to more typical (explicit) regularization.

How can we interpret these constraints? We give three interpretations, the first one specific to the flu forecasting setting, and the next two general.

Flu interpretation

In the flu nowcasting problem, recall, the map H has rows that sum to 1, and they reflect the geographic level at which the corresponding sensors were trained (see Section 2.4). The constraints $H^T b_j = e_j$, $j = 1, \dots, k$ can be seen in this case as a mechanism that accounts for the geographical hierachy underlying the sensors. As a concrete example, consider the simplified setup in Figure 2.1, and $j = 3$. The constraint $H^T b_3 = e_3$ reads:

$$\begin{aligned} b_{31} + {}^1\!/_3\, b_{36} + {}^1\!/_5\, b_{38} &= 0, \\ b_{32} + {}^1\!/_3\, b_{36} + {}^1\!/_5\, b_{38} &= 0, \\ b_{33} + {}^1\!/_3\, b_{36} + {}^1\!/_5\, b_{38} &= 1, \\ b_{34} + {}^1\!/_3\, b_{37} + {}^1\!/_5\, b_{38} &= 0, \\ b_{35} + {}^1\!/_3\, b_{37} + {}^1\!/_5\, b_{38} &= 0. \end{aligned}$$

The third line can be interpreted as follows: an increase of 1 unit in sensor z_3, $1/3$ units in z_6, and $1/5$ units in z_8, holding all other sensors fixed, should lead to an increase in 1 unit of our prediction for x_3. This is a natural consequence of the hierarchy in the sensor model (2.2), visualized in Figure 2.1. The first line can be read as: an increase of 1 unit in sensor z_1, $1/3$ units in z_6, and $1/5$ in z_8, with all others fixed, should not change our prediction for x_3. This is also natural, following from the hierachy (i.e., such a change must have been propogated by x_1). The other lines are similar.

Invariance interpretation

The SF prediction (at time $t + 1$) is $\hat{x}_{t+1} = \hat{B}^T z_{t+1}$. To denoise (i.e., estimate the mean of) the measurement z_{t+1}, based on the model (2.2), we could use $\hat{z}_{t+1} = H\hat{x}_{t+1}$. Given the denoised $\hat{z}_{t+1}$, we could then refit our state prediction via $\tilde{x}_{t+1} = \hat{B}^T \hat{z}_{t+1}$. But due to the constraint $H^T \hat{B} = I$ (a compact way of expressing $H^T \hat{b}_j = e_j$, for $j = 1, \dots, k$), it holds that $\tilde{x}_{t+1} = \hat{B}^T H \hat{x}_{t+1} = \hat{x}_{t+1}$. This is a kind of *invariance* property. In other words, we can go from estimating states, to refitting measurements, to refitting states, etc., and in this process, our state estimates will not change.

Generative interpretation

Assume $t \geq k$, and fix an arbitrary $j = 1, \dots, k$ as well as $b_j \in \mathbb{R}^k$. The constraint $H^T b_j = e_j$ implies, by taking an inner product on both sides with x_i, $i = 1, \dots, k$,

$$(Hx_i)^T b_j = x_{ij}, \quad i = 1, \dots, k.$$

If we assume x_i, $i = 1, \dots, k$ are linearly independent, then the above linear equalities are not only implied by $H^T b_j = e_j$, they are actually equivalent to it. Invoking the model (2.2), we may rewrite the constraint $H^T b_j = e_j$ as

$$\mathbb{E}(b_j^T z_i | x_i) = x_{ij}, \quad i = 1, \dots, k. \tag{2.15}$$

In the context of problem (2.14), this is a statement about a *generative* model for the data (as $z_i|x_i$ describes the distribution of the covariates conditional on the response). The representation in (2.15) shows that (2.14) constrains the regression estimator to have the correct conditional predictions, on average, on the data we have already seen (x_i, z_i), $i = 1, \dots, k$. (Note here we did not have to use the first k time points; any past k time points would suffice.)

2.6 Influenza nowcasting experiments

2.6.1 Setup

We examine the performance of our methods for nowcasting (one-week-ahead prediction of) wILI across 5 flu seasons, from 2013 to 2018 (total of 140 weeks). As described earlier, we have $k = 51$ states and $d = 308$ measurements. At week $t+1$, we derive an estimate $\hat{x}_{t+1}$ of the current wILI in the 51 US states, based on sensors z_{t+1} (each sensor being the output of an algorithm trained to predict wILI at a different geographic resolution from a given data source), and past wILI and sensor data.

We consider 7 methods for computing the nowcast $\hat{x}_{t+1}$: (i) SF, or equivalently, constrained regression (2.14); (ii) SF as in (2.14), but with an additional ridge (squared ℓ_2) penalty (equivalently, SF with covariance shrinkage); (iii) SF as in (2.14), but with an additional lasso (ℓ_1) penalty; (iv/v) regression as in (2.14), but without constraints, and using a ridge/lasso penalty; (vi) random forests (RF) [Breiman, 2001], trained on all of the sensors; (vii) RF, but trained on all of the underlying data sources used to fit the sensors.

At prediction week $t + 1$, we use the last 3 years (weeks $t - 155$ through t) as the training set for all 7 methods. We do not implement unpenalized regression (as in (2.14), but without constraints), as it is not well-defined (156 observations and 308 covariates). We point out that SF is still well-defined, due of the constraint in (2.14): a nonunique solution only occurs when the (random) null space of the covariate matrix has a nontrivial intersection with the null space of H^T, which essentially never happens. All ridge and lasso tuning parameters are chosen by optimizing one-week-ahead prediction error over the latest 10 weeks of data (akin to cross-validation, but for a time series context like ours).

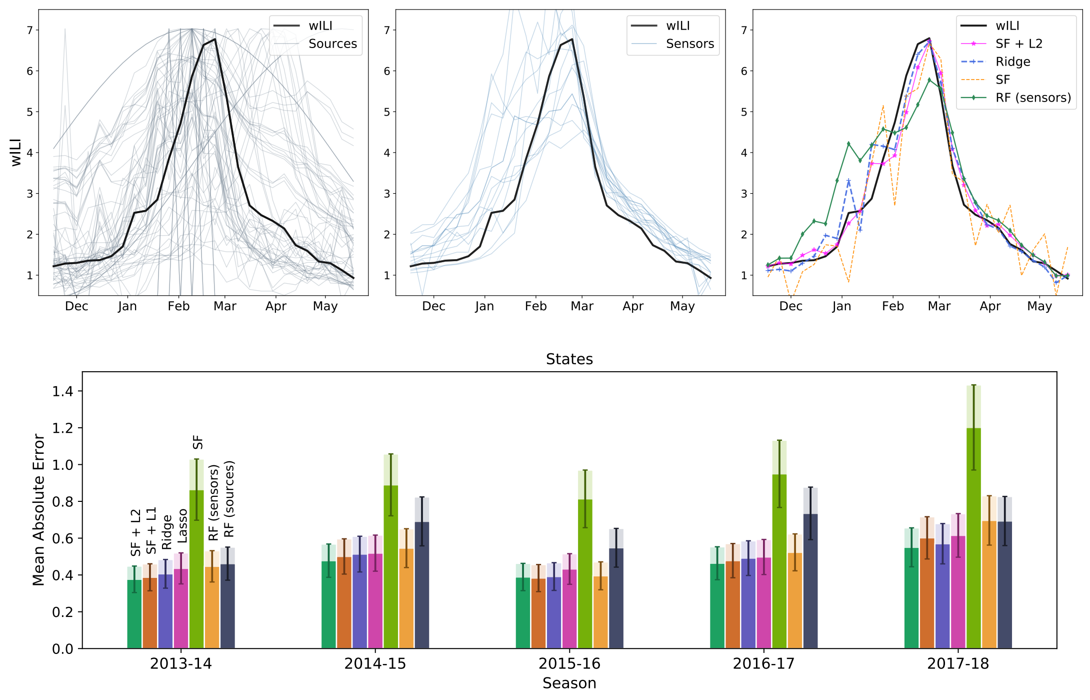


Figure 2.2: *Top row, from left to right: data sources, sensors, and nowcasts are compared to the underlying wILI values for Pennsylvania during flu season 2017-18. For visualization purposes, the sources are scaled to fit the range of wILI. On the rightmost plot, we display nowcasts using select methods. Bottom row: MAEs (full colors) and MADs (light colors) of nowcasts over 5 flu seasons from 2013-14 to 2017-18.*

Real-data missingness

Unfortunately, sensors are observed at not only varying geographic resolutions, but also varying temporal resolutions (since their underlying data sources are), and missing values occur. In our experiments, we choose to compute predictions using the regression perspective, and apply a simple mean imputation approach (using only past sensor data), before fitting all models.

2.6.2 Results and interpretations

The bottom row of Figure 2.2 displays the mean absolute errors (MAEs) from one-week-ahead predictions by the 7 methods considered, averaged over the 51 US states, for each of the 5 seasons. Also displayed are the mean absolute deviations (MADs), in light colors. We see that SF with ridge regularization is generally the most accurate over the 5 seasons, SF with lasso regularization is a close second, and SF without any regularization is the worst. Thus, clearly, explicit regularization helps. Importantly, we also see that the constraints in the regression problem (2.14) (which come from its connection to SF) play a key role: in each season, SF with ridge regularization outperforms ridge regression, and SF with lasso regularization outperforms the lasso. Therefore, the constraints provide additional (beneficial) implicit regularization.

RF trained on sensors performs somewhat competitively. RF trained on sources is more variable (in some seasons, much worse than RF on sensors). This observation indicates that training the sensors is an important step for nowcasting accuracy, as this can be seen as a form of denoising, and suggests a view of all the methods we consider here (except RF on sources) as prediction assimilators (rather than data assimilators). Finally, the top row Figure 2.2 visualizes the nowcasts for Pennsylvania in the 2017-18 season. We can see that SF, RF (on sensors), and even ridge regression are noticeably more volatile than SF with ridge regularization.

2.7 Discussion

In this work, we studied connections between the Kalman filter, sensor fusion, and regression. We derived equivalences between the first two and latter two, and discussed various interpretations and implications of our results. We studied the application of our work to nowcasting the weekly influenza levels in the US. We remark that the equivalences described in this chapter are deterministic, in that they do not require the modeling assumptions (2.1), (2.2), or any modeling assumptions whatsoever. Furthermore, even though their proofs are elementary (they are purely linear algebraic) and the setting is a classical one (linear dynamical systems), these equivalences are—as far as we can tell—new results. They may have implications beyond what is explored in this work.

For example, the regression formulation of SF may still be a useful perspective for problems in which past states are fully unobserved (this being the case in most KF applications). In such problems, we may consider using *smoothed* estimates of past states, obtained by running a backward version of the KF forward recursions (2.3)–(2.7) (see, e.g., Chapter 7 of Anderson and Moore [1979]), for the purposes of the regression formulation.

As another example, the SF view of the KF may be a useful formulation for the purposes of estimating the covariances R, Q, or the maps F, H, or all of them; in this paper, we assume that F, H, R, Q are known (except for in the regression formulation of SF, in which R is unknown but past states are available); in general, there are well-developed methods for estimating F, H, R, Q such as *subspace identification* algorithms (see, e.g., Overshee and Moor [1996]), and it may be interesting to see if the SF perspective offers any advantages here.

We provide several modifications of the basic SF formulation (leaving details to A.4) and show that these also equivalences in the regression perspective: namely, shrinking the empirical covariance in (2.13) towards the identity is equivalent to adding a ridge (squared ℓ_2) penalty to the criterion in (2.14); and also, adding a null sensor at each state (one that always outputs 0) is equivalent to removing the constraints in (2.14). The latter equivalence here provides indirect but fairly compelling evidence that the constraints in the regression formulation (2.14) play an important role (under the model (2.2)): it says that removing them is equivalent to including meaningless null sensors, which intuitively should worsen its predictions.

Future directions of this work are given in Section 5.2 of our Discussion chapter, and there we introduce various extensions of the regression formulation that do not have clear equivalences in the KF perspective. Namely, we detail modifications of the regression problem to allow for distributional nowcasting, and gradient boosting approaches to simultaneously improve the sensors and assimilate them.

Chapter 3

Nowcasting Convolved Signals

This work was done in collaboration with Andrew Chin and Ryan Tibshirani, and contains content that appears in:

> Maria Jahja, Andrew Chin, and Ryan J. Tibshirani. Real-Time Estimation of COVID-19 Infections: Deconvolution and Sensor Fusion. To appear: *Statistical Science*, 2022.

R and Python code for this work is available at: https://github.com/cmu-delphi/stat-sci-nowcast.

3.1 Introduction

In this chapter, we study nowcasting in a setting where we have only access to the convolved outcomes $y_t \in \mathbb{R}^k$ of latent (and never observed) states $x_t \in \mathbb{R}^k$, $t = 1, 2, 3, \ldots$. As before, $z_t \in \mathbb{R}^d$, $t = 1, 2, 3, \ldots$ denote the observed sensors measuring the latent state.

As previously mentioned, KF is often applied in problems where the state is always unobserved (indeed, it was originally devised for this setting), and the process model (2.1) is applied to evolve estimates of the past state (and subsequently corrected via (2.2)). Using an extension of our sensor fusion (SF) approach where we estimate past states (say, the Kalman smoothing approach outlined in Section 2.7), we could simply designate the outcomes y_t as yet another sensor (with a lagged relationship), and append it to the measurement vector z_t. However, doing so would overlook the informative structure of the convolutional relationship between states x_t and outcomes y_t. In our work, we choose to distinguish y_t from z_t, and instead directly employ deconvolution to first recover the entire sequence of past states $\hat{x}_t$, $t = 1, 2, 3, \ldots$.

The advantages are two-fold. First, we are able to easily incorporate specialized techniques to improve the volatility of estimates on the right boundary (corresponding to the most recent state values). This volatility is due to a data truncation issue, often called *right truncation*, which occurs in real-time nowcasting. Right truncation (which we describe in detail shortly) poses a critical challenge in producing stable nowcasts, and many of our contributions in this work are motivated by it. The second advantage is that we now have the past state estimates in hand, without introducing additional state modeling, and can immediately apply the SF framework as before without modification.

In what follows, we will ground our proposed methods around the goal of estimating newly onset COVID-19 infections. Infection incidence, to be clear, is a completely latent time series, emitting only observed outcomes captured in published case reports. As for our sensors, they are nearly identical to the surveillance sensors described previously in Section 2.4, but are indicators of COVID-19 disease activity rather than influenza. In addition to the extra complications of the latent state, we also focus on estimating *daily* infections in real-time, and lay out a framework for an operational nowcasting system that is forced to cope with all the challenges of disease tracking using provisional data that can be heavily revised, or occasionally delayed.

Performing real-time nowcasting not only affects the way we carry out our experiments (both model training and evaluation), it also leads us to develop novel methodology to deal with the issue of right truncation (highlighted in Figure 3.1 by the blue region) mentioned earlier. For example, in order to estimate the delay distribution that convolves infection symptom onset to case report, we develop a Kaplan-Meier-like procedure to deal with a form of right censoring that occurs in our real-time data stream. We also develop new regularization methods to stabilize the most recent estimates at nowcast time for the optimization problem that we solve in real-time deconvolution.

We organize the rest of this chapter as follows. In the following Section 3.2, we provide background and motivation for our work towards COVID-19 nowcasting. We give an overview of related work in Section 3.3. In Section 3.4, we cover the available data sources and infrastructure, and other preliminary details about the problem setup. Formalization of the convolutional model and retrospective construction of the convolution delay distribution are described in Section 3.5. In Section 3.6, we will cover the core issues in real-time estimation, and detail the substantive portion of our methodological contributions. The sensor fusion layer is covered in Section 3.7. Section 3.8 contains extensive evaluations—comparing nowcasts made in real-time to those made retrospectively (using "finalized" data that would have only been available much later), and the correlation of these nowcasts to COVID-19 hospitalization rates; in this section we also introduce a simple post hoc smoothing method. We conclude in Section 3.9 with a discussion and outline a few directions for future work.

3.2 COVID-19 background and motivation

Accurate, real-time estimates of incident infections play a critical role in informing the public health response to the spread of a disease through a population. However, official metrics on disease activity published by traditional public health surveillance systems in the United States do not in fact reflect activity in real-time, as they suffer from some degree of latency due to the way their reporting pipelines are set up and implemented.

With addressing the latency in traditional public health reporting a part of the motivation, the last decade has seen a rise in the development of *digital surveillance* streams in public health. Search and social media trends have constituted much of the focus [e.g., Brownstein et al., 2009, Ginsberg et al., 2009, Kass-Hout and Alhinnawi, 2013, Paul and Dredze, 2017, Salathé et al., 2012]. More broadly, *auxiliary surveillance* streams that operate outside of traditional public health surveillance, like online surveys, medical device logs, or electronic medical records, have also received significant attention [e.g., Ackley et al., 2020, Carlson et al., 2013, Charu et al., 2017, Kass-Hout and Zhang, 2011, Leuba et al., 2020, Radin et al., 2020, Santillana et al., 2016, Smolinski et al., 2015, Viboud et al., 2014, Yang et al., 2019].

Auxiliary surveillance can improve not only on the timeliness but also on the accuracy and robustness of traditional public health reporting. Auxiliary data streams have therefore become an integral part of modern systems for disease *nowcasting* [e.g., Brooks, 2020, Farrow, 2016, Jahja et al., 2019, McIver and Brownstein, 2014, Santillana et al., 2015, Yang et al., 2015], which, put broadly, are used to estimate the contemporaneous value of a signal that will only be fully observed at a later date, using partial or noisy data.

3.2.1 Surveillance during the pandemic

During the COVID-19 pandemic, public health surveillance has produced, on one hand, some of the most detailed public health data that the U.S. has ever seen, such as daily, county-level data on reported COVID-19 cases and deaths. It has also, on the other hand, painted an imperfect picture of situational awareness, which created a number of downstream challenges for the public health response. See, e.g., Rosenfeld and Tibshirani [2021] and references therein for an overview of the issues. In this work, we identify a few issues surrounding COVID-19 case reporting in particular, propose methodology to address them, and implement and evaluate this proposal over eight months of pandemic data.

To give some background, in the early days of the pandemic, a handful of non-gonvermental groups such as JHU CSSE [Dong et al., 2020] (and also the COVID

Tracking Project, the New York Times, and USAFacts) became known as the most trustworthy sources for aggregate public health reporting data on COVID-19 in the U.S. They were founded around the idea of scraping COVID-19 data published daily on dashboards that are run by local public health authorities (such as state and county departments of public health), which, at the time, provided more accurate and timely data than federal health authorities (probably due to unrecoverable failures at one or more points along the reporting pipeline). In fact, not only in the early days of the pandemic, but throughout, the data published by these groups has been invaluable for decision-makers, modelers, journalists, and the general public; for example, data from JHU CSSE remains the gold standard for COVID-19 case and death forecast evaluation in the COVID-19 Forecast Hub [Reich Lab, 2020], a community-driven repository of forecasts that serves as the official source for forecasting communications by the U.S. CDC.

Turning our focus now to case reporting, JHU scrapes cumulative case numbers that are published daily on local health authority dashboards, and subsequently derives a notion of case incidence based on day-to-day differences in cumulative counts. Note that, by construction, this definition of incidence reflects the number of new COVID-19 cases that are *reported* (to the public) on any given day. Of course, this is not the same as the number of new cases by date tested, specimen collection date, or symptom onset date. Any of the latter options would be more informative (increasingly so) as a definition of incidence; revamping our surveillance systems so that they can directly provide these and other aggregates of interest to the public health response is a critical task for future public health crises.

The reality of the current pandemic: alignment by report date is the only option available, given the data published broadly on local health authority dashboards, hence collected and aggregated by data scrapers. JHU publishes the number of new COVID-19 case reports per U.S. county, daily, at a 1-day lag. However, since report dates can lag behind symptom onset dates by many days (a typical lag is around 5-10, but lags can be up to 30 days or more; see Figure 3.3), this is actually giving us a glimpse into COVID activity in the recent past, rather than the present.

Importantly, the CDC publishes a de-identified patient-level data set ("line list") on COVID-19 infections [Centers for Disease Control and Prevention, COVID-19 Response, 2020a], which provides a symptom onset date column. In principle, this should allow us to construct a notion of case incidence that is aligned by symptom onset date, but this is not possible in practice, due to two barriers. First, the CDC only publishes updates to the line list monthly (due to the complexity of managing this data set). Second, and more problematically, this line list is fraught with missingness, extending well beyond missingness in the symptom onset column: the *total* number of COVID-19 cases according to this line list (whether the symptom onset date is observed or not) is far less than the total number of cases from JHU (e.g., in early September 2021, the CDC line list reports about 30 million total versus about 40 million from JHU), and some states (such as Texas) appear to missing nearly all of their cases in the line list altogether (see Figure 3.2).

3.2.2 Confounding

Estimates of COVID infections obtained by deconvolving reported cases will generally underestimate the true number of infections, because many infections are undetected or untested, and as such, do not appear later on in case reports. If we wanted to estimate the true number of symptomatic infections from case reports, then we would need to have some sense of the fraction of symptomatic infections that go untested. Of course, this only gets more complicated if we extend our consideration to both symptomatic and asymptomatic infections.

Other authors, e.g., Chitwood et al. [2021], have taken the ambitious step of proposing and implementing frameworks with parameters that account for such confounding. However, adjustments for case ascertainment and asymptomatic infections generally rely, at least to some nontrivial extent, on model assumptions (typically, mechanistic ones) that are difficult to substantiate.

We take a different perspective and pose the problem as one of real-time deconvolution only. We seek to answer the question:

> *Can we estimate—in real-time—the number of new symptomatic COVID-19 infections that will eventually appear in case reports?*

Hence, by construction, confounding is not a problem that we even attempt to reconcile (because the target we track, infections that eventually show up in case reports, simply inherits any confounding that would be present in the case reporting stream in the first place).

Our approach can be seen as one that runs in parallel (rather than in contradiction) to an approach that explicitly models and removes the effects of confounding in case reporting. We focus on addressing the deconvolution problem as carefully as possible, with a concern for real-time estimation, and an eye toward using auxiliary signals to improve accuracy and robustness. Estimates of parameters that account for confounding (that comes from other work focused on these aspects) could certainly be applied to our deconvolution estimates post hoc in order to adjust them appropriately; we revisit this idea in the discussion.

Lastly, under an assumption that the confounding acts as a multiplicative bias that changes slowly over time, our real-time infection rate estimates—themselves subject to confounding, as explained above—can be post-processed to derive real-time *approximately unconfounded* estimates of R_t, the instantaneous reproductive number, an oft-used epidemic parameter.

3.3 Related work

In the computational epidemiology literature, the term "nowcasting" has been applied to a variety of related but distinct estimation problems. Broadly speaking, what these problems have in common is that they are about real-time estimation of some quantity, based on partial or noisy data. They differ in *what* is being estimated, and whether this quantity will eventually be fully observed (after enough time has passed) or whether it is latent. Examples in the former non-latent setting, which span applications in influenza, dengue, and COVID-19, include Brooks [2020], Farrow [2016], Hawryluk et al. [2021], Jahja et al. [2019], McGough et al. [2020], Yang et al. [2015].

The latent setting exhibits another degree of diversity within itself. In our work, we target symptomatic COVID-19 infections, which, to be perfectly clear, is a latent time series. Another example along similar lines is Goldstein et al. [2009], who estimate influenza infection incidence via Bayesian deconvolution of mortality data. Meanwhile, other authors might view inferring latent infections as just a stepping stone toward ultimately estimating the instantaneous reproductive number R_t. Important contributions to the methodology on real-time estimation of R_t include: Bettencourt and Ribeiro [2008], who use a local approximation to the SIR model, and Cori et al. [2013], Thompson et al. [2019], who use a discretization of the renewal equation within a Bayesian framework. For a thorough review and comparison of these methods, see Gostic et al. [2020]. The latter paper also discusses in some detail the importance of properly modeling the delay between infection onset and case report, and the issue of right truncation, which, as we will see, are central issues in our work as well.

The aforementioned methods have been applied and extended to build systems for real-time R_t nowcasting during the COVID-19 pandemic by Abbott et al. [2020], Chitwood et al. [2021], Systrom et al. [2020]. A key difference between these approaches and ours is that they infer infections through forward-filling: loosely speaking, they convolve forward a candidate estimate of infections, obtain feedback by comparing the result to measured cases, and iterate to refine estimates. This can be effective given accurate prior knowledge, but of course it can be hard to judge the accuracy of prior knowledge in practice. We take a more flexible approach and estimate infections via direct deconvolution. Our approach is nonparametric, but is still fairly simple and computationally efficient. We also focus on fusing in auxiliary sources of information in order to improve real-time accuracy and robustness. We remark that, if estimates of R_t were desired, then these could certainly be inferred as a by-product of our infection nowcasts.

Finally, deconvolution has been extensively studied for many years in many fields, notably signal and image processing, where deconvolution is sometimes called deblurring. As an inverse problem, deconvolution is ill-posed in settings in which the convolution operator is not known exactly or observations are made with noise [Oppenheim and Verghese, 2017]. Approaches to overcome this traditionally involve regularization, as in the classical Wiener deconvolution [Wiener, 1964], which stabilizes the inversion using an estimated signal-to-noise ratio. Alternative approaches employ familiar regularization techniques such as ℓ_1 and ℓ_2 penalities [Debeye and Van Riel, 1990, Taylor et al., 1979]. Most related to our work is deconvolution using total variation regularization, first proposed by Rudin and Osher [1994], and now a central tool in signal and image processing.

3.4 Preliminaries

In what follows, we develop a framework for estimating the daily symptomatic COVID-19 infection rate (where by "rate" we mean a count per 100,000 people, the standard units in epidemiology), concentrating on infections that will eventually result in a reported COVID-19 case. To be clear on nomenclature: for convenience, we will often abbreviate "symptomatic infection" by "infection" (and so, terms like "infection onset" and "infection rate" should be implicitly interpreted as symptomatic). To estimate infection rates, we deconvolve reported case rates with an estimated symptom-onset-to-case-report delay distribution. In the following experiments, we use the case data from JHU CSSE [Dong et al., 2020], and to infer the delay distribution, we use a de-identified line list on patient-level infections from the CDC [Centers for Disease Control and Prevention, COVID-19 Response, 2020a].

3.4.1 Auxiliary indicators

After deconvolution, we improve our infection rate estimates by incorporating a number of contemporaneous signals that track COVID activity—we will also refer to these as *indicators*—which are publicly available through Delphi's COVIDcast API [Reinhart et al., 2021]. The five indicators that we consider, described below, provide auxiliary information on COVID-19 outside of traditional public heath reporting. Here and throughout, we abbreviate COVID-like illness by CLI.

1. Change Healthcare COVID (CHNG-COVID): The percentage of outpatient visits that have confirmed COVID-19 diagnostic codes, based on de-identified Change Healthcare medical claims data.
2. Change Healthcare CLI (CHNG-CLI): The percentage of outpatient visits that

have COVID-like diagnostic codes, based on the same data.

3. Doctor Visits CLI (DV-CLI): The same definition as CHNG-CLI, but applied to de-identified medical claims data from other health systems partners.
4. COVID Trends and Impact Survey CLI in the community (CTIS-CLIIC): The estimated percentage of people reporting illness in their household or local community, based on Delphi's COVID Trends and Impact Survey (CTIS), in partnership with Facebook.
5. Google searches for anosmia and ageusia (Google-AA): A measure of volume for Google queries related to anosmia or ageusia (loss of smell or taste), from Google's COVID-19 Search Trends data set.

Roughly speaking, we study these particular indicators (ordered roughly from "late" to "early") because conceptually they reflect data measurements that would be made at some period of time in between infection onset and case report to a public health authority, and therefore would be relevant in inferring latent infection rates. More information on these indicators and their underlying data sources is given in Reinhart et al. [2021]. For more information on CTIS in particular, see Salomon et al. [2021]; and for a study of how these and similar indicators can improve COVID-19 forecasting, see McDonald et al. [2021].

Sensor fusion layer

For each of the auxiliary indicators described above, we train a model to estimate latent infection rates from indicator values, using historical data (described in Section 3.7.1). At each nowcast date, we then use such a model to estimate the latent infection rate from the current indicator value, which gives a total of five estimates (one from each of the five models), along with a sixth estimate coming from an autoregressive model trained on historical estimated infection rates. We will refer these six contemporaneous estimates as *sensors*.

In this chapter, we consider (as described in Section 3.7.3) various methods for combining these estimates into a single estimate of the infection rate, among which contain the Kalman filter-based SF described in the Chapter 2. We will call this class of methods *sensor fusion* methods, as broadly speaking, sensor fusion is a form of ensembling, which is ubiquitous in in predictive modeling in statistics and machine learning, as it can often help improve both accuracy and robustness. In our particular application, the sensors themselves are constructed from data streams operating outside of traditional public health reporting, which itself contributes an additional important angle in terms of robustness.

An illustration is given in Figure 3.1, where sensor fusion improves accuracy and

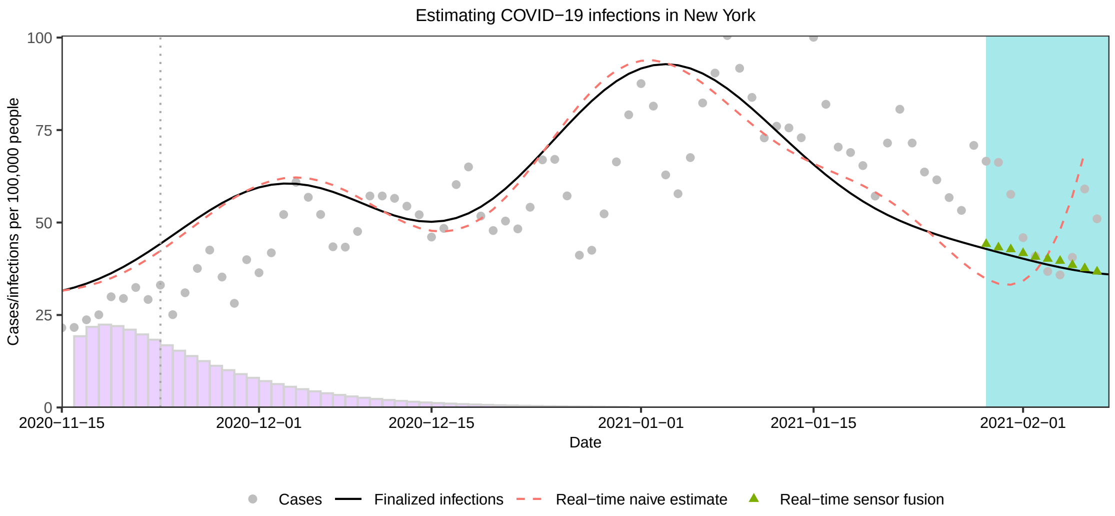


Figure 3.1: *Illustration of estimating latent infections from reported cases. The dashed red line displays infection rates estimated "naively" in real-time, by directly deconvolving case data up through early February 2021, while the solid black line display infection rates estimated using finalized data from roughly four months afterwards. The blue region on the right-hand side highlights a period in which the real-time estimate deviates substantially from the finalized one, due to the fact that we are lacking sufficient (future) case observations needed to perform a "full" deconvolution. The green triangles represent real-time nowcasts made by sensor fusion, which reduces the volatility of the real-time estimate and tracks the finalized estimate nicely. Lastly, the (scaled) reporting delay distribution estimated at the midpoint of November 2020 is drawn in purple, with the median reporting delay (8 days) marked as a dotted gray line.*

robustness of our estimates of new infections for the most recent 10 days (where deconvolution is particularly challenging).

3.4.2 Problem setup

Estimation period

For every day t in between October 1, 2020 and June 1, 2021 inclusive (243 days in total), we estimate the symptomatic infection rate at day $t - k$, using only data that would have been as of time t, which in this context we call the *nowcast date*. Estimation of the latent infection rate at time $t - k$ (for positive k) is technically a backcast, though we will not be careful to distinguish this notationally from nowcasting, and will generally refer to this as nowcasting at lag k. We produce estimates for each $k = 1, \dots, 10$, a total of 10 targets per nowcast date t.

When we say above that nowcasts are made using data that would have been available *as of* a given nowcast date t, we mean that we adhere not to only the real-time availability (latency) of signals at t, but also the *version* of the data published at t—simply put, imagine that we "rewind" the clock to time t and query the API to receive the data that would have been returned then. This is possible becausse the COVIDcast API records and provides access to all historical versions of data, as described in Reinhart et al. [2021]. As epidemic data is often subject to revision, if we train and evaluate models on "finalized" data (that would have been available only at a much later time point) then this can lead to inaccurate conclusions about real-time model performance; see, e.g., McDonald et al. [2021].

Further, it is worth noting that reported case data from JHU is available at a 1-day lag, and we assume that there is at least another 1-day lag between symptom onset and case report (explained in Section 3.5.2). Hence through real-time deconvolution alone we would be able to make nowcasts at a 2-day lag at the earliest. Making nowcasts at a 1-day lag is possible with sensor fusion, using auxiliary signals with 1-day latency (explained in Section 3.7). In this sense, sensor fusion is able to improve not only accuracy, but also latency, and buys us 1 extra day.

Geographic scope

We produce nowcasts at the county resolution, but for computational purposes, we restrict our attention to the 200 U.S. counties with the highest population. We additionally produce estimates for each of the 50 U.S. states. (Some of the methodology that we use for sensor fusion requires a geographical hierarchy, thus using the remaining $\approx$ 3000 U.S. counties we aggregate these within each state to create "rest-of-state" jurisdictions, and make estimates for these as well, for the purposes or maintaining such a hierachy.)

Evaluation period

We evaluate all nowcasts made in between October 1, 2020 and June 1, 2021 inclusive (243 days in total) and at each of the 250 locations in consideration (50 states and the 200 largest counties) against latent infection rate estimates obtained by deconvolving the case rate data available as of August 30, 2021. We will refer to the latter as *finalized* infection rate estimates (as opposed to real-time ones); details are given in Section 3.5.3.

3.5 Retrospective deconvolution

In this section, we study and fit a convolutional model between infections and reported cases. We adopt a *retrospective* angle here and do not concern ourselves with data availability or versioning issues; this is covered in the next section.

3.5.1 Convolutional model

For simplicity, we introduce the convolutional model in just a single location. We denote by y_t the number of new cases that are reported at time t, and by x_t the number of new infections that have onset at time t. Our jumping-off point is the following model:

$$\mathbb{E}[y_t \,|\, x_s, s \leq t] = \sum_{s=1}^{t} \pi_t(s)\, x_s, \tag{3.1}$$

where for each $s \leq t$,

$$\pi_t(s) = \mathbb{P}\big(\text{case report at } t \,|\, \text{infection onset at } s\big). \tag{3.2}$$

We refer to the probabilities above as *delay probabilities* at time t, and the entire sequence $(\pi_t(s) : s \leq t)$, as the *delay distribution* at time t.

The justification for (3.1), (3.2) is elementary: to count y_t, we enumerate all infections that ever occurred in the past:

$$y_t = \sum_{s=1}^{t} \sum_{i=1}^{x_s} 1\{\text{the } i^{\text{th}} \text{ infection at } s \text{ gets reported at } t\}.$$

Taking a conditional expectation on both sides above, and using linearity, delivers (3.1), (3.2).

In the next subsections, we will describe how to estimate the probabilities $\pi_t(s)$ in (3.2), and how to use this alongside the observed case reports y_t in order to estimate the latent infections in (3.1).

3.5.2 Estimating the delay distribution

At the outset, we place the following assumptions on the delay distribution in order to make its estimation (using the CDC line list data, to be described shortly) more tractable.

Assumption 1. Infections are always reported within $d = 45$ days; that is, $\pi_t(s) = 0$ whenever $s < t - d$.

Assumption 2. The probability of zero delay is zero; that is, $\pi_t(t) = 0$.

Assumption 3. The delay distribution is geographically invariant (it is the same for any location).

Assumption 1 is innocuous. The vast majority of pairs of recorded infection dates and report dates in the CDC line list data fall within $d = 45$ days of one another. Assumption 2 is perhaps less innocuous but still fairly minor, and it is a consequence of the fact that a delay of zero (infection date equal to report date) has been used inconsistently in the CDC line list: this could mean a true delay of zero, or it could be a code for missingness.

Assumption 3 is the most noteworthy and troublesome. We do *not* believe it to be true that different locations actually have identical patterns of delay between infections and case reports; conversely, we expect there to be a considerable amount of variability between locations in this regard. While we do allow the delay distribution to change over time (see Figure 3.3 for evidence for the importance of this), we consider Assumption 3 to be a weakness of our work. However, the *data is simply not there* in the CDC line list to warrant location-specific estimation of the delay distribution (see Figure 3.2), thus we resort to estimating a nation-wide delay distribution.

Meanwhile, it is worth pointing out that better (location-specific) estimates of the delay distribution could be simply plugged into our deconvolution methodology (detailed in Section 3.5.3) to yield better estimates of latent infections. This would carry over to all of the real-time methodology for deconvolution and sensor fusion (in Section 3.6) as well. In other words, a strength of our methodology is that it can treat the delay distribution as an input, and a user (say, a local health official) can replace the default nation-wide delay distribution with a more-informed local one in order to get more-informed local estimates.

In light of Assumptions 1 and 2, we change our notation henceforth, and rewrite (3.1), (3.2) as:

$$\mathbb{E}[y_t \,|\, x_s, s \le t] = \sum_{k=1}^{d} p_t(k)\, x_{t-k}, \tag{3.3}$$

where for $k = 1, \dots, d$,

$$p_t(k) = \mathbb{P}\big(\text{case report at } t \,|\, \text{onset at } t-k\big). \tag{3.4}$$

CDC line list

The CDC provides de-identified patient-level surveillance data on COVID-19 in both public and restricted forms [Centers for Disease Control and Prevention, COVID-19 Response, 2020a,b]. The restricted one is made available under a data use agreement. The public line list contains the same patient-level records as the restricted one, but it has geographic details withheld. (There is another publicly available that contains geographic details, but withholds temporal details). We use the public data set[1] for estimating the delay distribution, since missingness compels us to make nation-wide (rather than location-specific) estimates.

It is worth noting that the line list is itself provisional and subject to revision. Furthermore, the CDC only publishes updates to the line list monthly. In our experiments, for simplicity, we use a single version of the CDC line list—released on September 9, 2021—to construct all delay distributions. Nonetheless, in our real-time nowcasting experiments, we restrict our access to data in this line list that would have been available at each nowcast date t (rows whose report date to the CDC is at most t) to construct delay distribution estimates at t. This is highly nontrivial, due to bias induced by truncation of data after t (see Section 3.6.2).

Missing values

The CDC line list (both public and restricted data sets) is subject to a high degree of missingness. Such missingness manifests itself in a variety of ways. For the public line list published on September 9, 2021:

- it has 29,851,450 rows, compared to 39,365,080 cumulative cases reported by JHU CSSE on September 9, 2021;
- 8.64% of rows are missing the case report date (the $\mathrm{cdc\_report\_dt}$ column);
- 53.6% of rows are missing the symptom onset date (the $\mathrm{onset\_dt}$ column);
- of all rows in which symptom onset date is present, the case report date is also present, but when a report date is missing in practice it sometimes gets filled in with the onset date, clouding the interpretation of a zero delay.[2]

Due to the last point, we exclude zero in the construction of all delay distribution estimates, in what follows.

The restricted line list is no better with respect to such missingness, exhibiting nearly exactly the same patterns as those described above. It does additionally provide geographic details, which allows us to examine how missingness is dispersed across different locations. Figure 3.2 displays results to this end, using the restricted line list released on October 12, 2021. The top panel shows that there is a high degree of missingness in complete case counts (those with both onset date and report date observed) in most states, often well over 50%, and moreover, missingness is far from uniform at random: e.g., Texas has barely any of its cases present in the line list. The latter observation is why we resort to estimating nation-wide delay distributions, in

[1]The CDC does not take responsibility for the scientific validity or accuracy of methodology, results, statistical analyses, or conclusions presented.

[2]Confirmed by personal communication with the CDC.

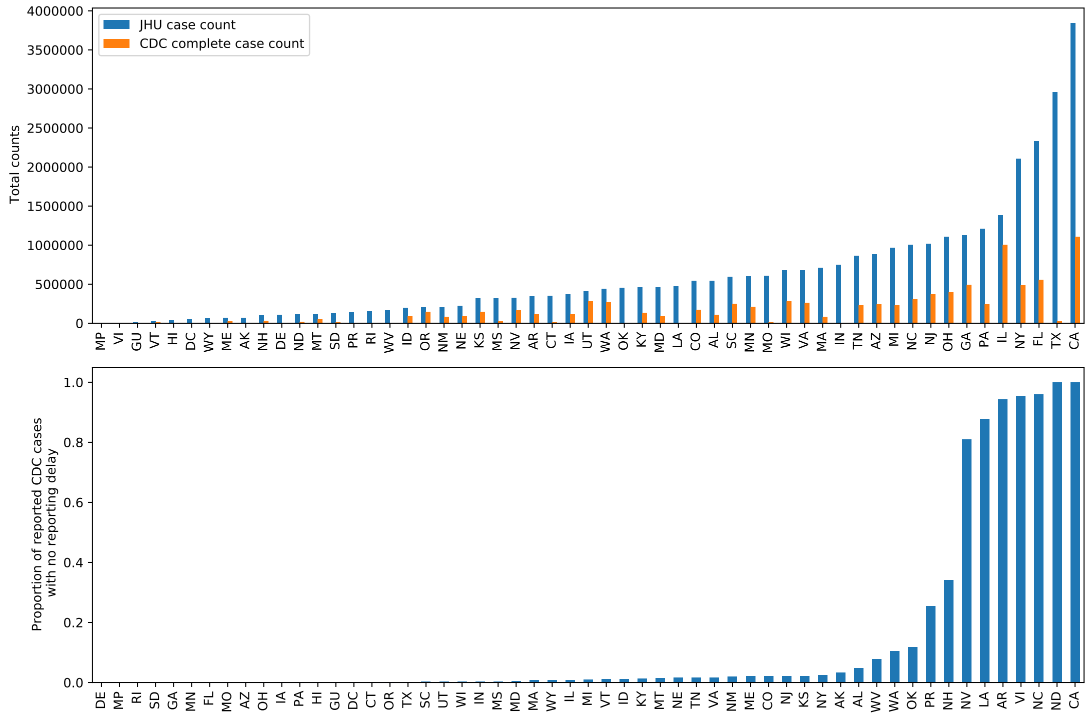


Figure 3.2: *Top row: cumulative case count per state on June 1, 2021, as reported by JHU CSSE, compared to the complete case count (where both onset date and report date are observed) per state up through the same date, in the CDC restricted line list. Most states have less than 50% of the cases appear in complete form in the line list, and some (e.g., Texas) have almost none at all. Bottom row: proportion of complete cases with zero delay per state in the same line list data. There is very wide variation between these proportions.*

what follows.

The bottom panel in the figure shows that there is also a high degree of heterogeneity in the fraction of complete cases with zero delay (between onset date and report date) across states. Some states (e.g., California) have zero delays for nearly all of their complete cases, while others (e.g., Delaware) have zero delays for none of their complete cases, suggesting that the practice of setting a missing report date equal to the associated onset date is highly inconsistent between states. This only further corroborates the decision to exclude zero delays from the data set when estimating the delay distribution.

Algorithm 3.1: Delay distribution estimation, retrospective

```
Input: Time $t$, support size $d$, window size $w = 2d$, line list $\mathcal{D}$ with onset
        dates $a_i$ and report dates $b_i$.
Output: Estimated delay probabilities $\hat{p}_t(1), \dots, \hat{p}_t(d)$.
Find all pairs in $\mathcal{D}$ with onset dates within a recent time window:
  $I_t = \{i : a_i \in (t - w, t]\}$.
Compute the empirical distribution of lags $1, \dots, d$ among these pairs:
        $\bar{p}_t(k) = \frac{|\{i \in I_t : b_i - a_i = k\}|}{\sum_{\ell=1}^d |\{i \in I_t : b_i - a_i = \ell\}|}, \quad k = 1, \dots, d.$
Fit a gamma density to $\bar{p}_t(1), \dots, \bar{p}_t(d)$ using the method of moments
  (matching the mean and variance).
Discretize this gamma density to the support set $\{1, \dots, d\}$, call the result
  $\hat{p}_t(1), \dots, \hat{p}_t(d)$, and return these probabilities.
```

Delay distribution estimation

From the public line list, we estimate the delay distribution at each time t, namely the probabilities in (3.4) for $k = 1, \dots, d$, using the empirical distribution of all lags, excluding zero, between complete onset and report dates, for all onset dates falling in $[t - 2d + 1, t]$. Then, we fit a gamma density to the empirical distribution by the method of moments, and discretize the resulting density over the support $\{1, \dots, d\}$. For concreteness, this procedure is described in Algorithm 3.1.

We use only "recent" pairs of onset and report dates at time t (whose onset date lies in $[t - 2d + 1, t]$) in order to adapt to the nonstationarity in reporting delays over time. The top panel in Figure 3.3 plots quantiles of the estimated delay distribution from Algorithm 3.1, as t ranges from June 1, 2020 to June 1, 2021. We see sharp drops in all quantiles during the first half of this period, and then a more gradual decline over time. The bottom panel in the figure gives a qualitative sense of how the delay distribution estimates change in shape over time.

3.5.3 Defining ground truth

Given the estimated delay distributions over time from the previous subsection, we now describe how to estimate latent infections in the model (3.3). In short, we will solve one large optimization problem to perform deconvolution. To define the best possible retrospective estimates of latent infections over the period October 1, 2020 to June 1, 2021, which we will treat as *ground truth* in what follows (in the sense that they will be the point of comparison for all of our real-time estimates), we will

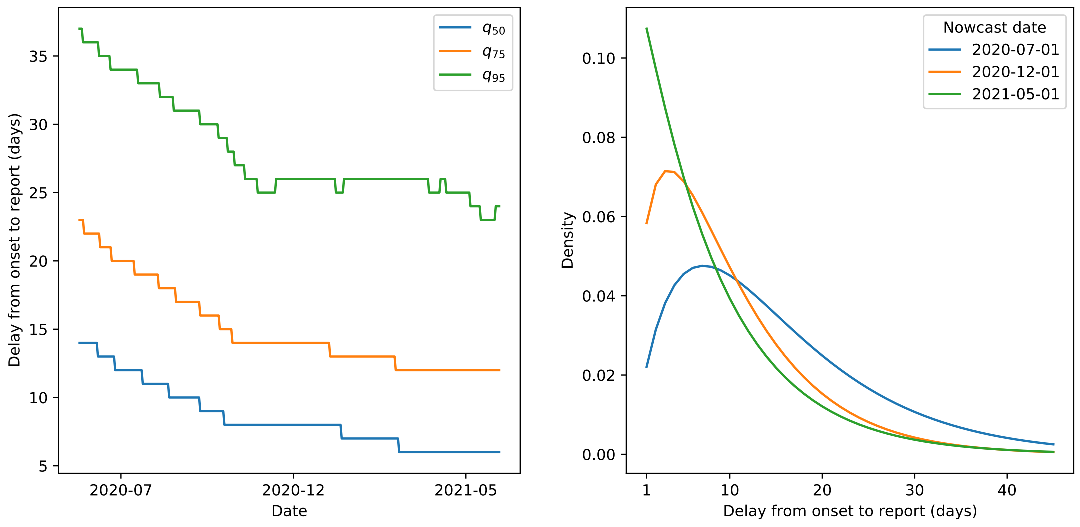


Figure 3.3: *Left: quantiles of the estimated delay distribution returned by Algorithm 3.1 at the levels 50%, 75%, and 95%, as t varies from June 1, 2020 to June 1, 2021. Right: estimated delay distributions overlaid for three nowcast dates within the same time interval.*

perform deconvolution over a wider time period than the previously specified one in order to avoid any bias issues at the boundaries (where there is insufficient data for accurate deconvolution; more details are provided in the next section): our retrospective deconvolution runs from May 1, 2020 to August 28, 2021, a period we denote by $\mathcal{T}$, and uses case data published on August 30, 2021.

For location ℓ, denote by $y_{\ell,t}$ and $x_{\ell,t}$ the number of new cases reported and number of new infections that onset at time t, respectively, per 100,000 people. Note that $y_{\ell,t}, x_{\ell,t}$ obey (3.3), (3.4), because we have just rescaled the underlying counts here by a constant (in order to put them on the scale of rates), and recall, we assume that all locations have the same delay distribution (Assumption 3).

Given the delay distribution estimates from Algorithm 3.1, $\hat{p}_t = (\hat{p}_t(1), \dots, \hat{p}_t(d))$ for $t \in \mathcal{T}$, we estimate the full vector $x_\ell = (x_{\ell,t})_{t\in\mathcal{T}}$ of latent infection rates across time, separately for each location ℓ, by solving the problem:

$$\underset{x_\ell}{\text{minimize}} \sum_{t\in\mathcal{T}} \left(y_{\ell,t} - \sum_{k=1}^{d} \hat{p}_t(k)\, x_{\ell,t-k} \right)^2 + \lambda \|D^{(4)} x_\ell\|_1, \tag{3.5}$$

where $D^{(4)}$ is a matrix such that $D^{(4)}v$ gives all 4th-order differences of a vector v, and $\|\cdot\|_1$ is the ℓ_1 norm. Problem (3.5) could be called a trend-filtering-regularized least squares deconvolution problem. We solve it (as well as all related optimization problems in this work) numerically with an adaption of the ADMM algorithm of Ramdas and Tibshirani [2016], detailed in Appendix B.1.

The solution $\hat{x}_\ell$ in problem (3.5) takes the form of a cubic piecewise polynomial (discrete spline) with adaptively chosen knots [Tibshirani, 2014, 2020]. The tuning parameter $\lambda \geq 0$ controls its complexity, and we choose it using 3-fold cross-validation: we hold out every third value from training, and impute it by the average of the neighboring trained estimates; to compute the validation error, we reconvolve the full vector of imputed infections and measure against observed cases.

3.6 Real-time deconvolution

Real-time deconvolution refers to the the task of deconvolving case reports observed up until time t to estimate latent infections up until t, repeatedly, as t marches over the period of interest. We are particularly focused on estimating recent latent infections—nowcasting at a k-day lag, which means estimating at t the latent infection rate at time $t - k$. (To clarify notation, k denotes the number of days prior to our nowcast time, not the dimension of the state x_t; as introduced in (3.1), the state corresponds to a single location, and its value at any point in time is a scalar quantity.)

Compared to retrospective deconvolution, real-time deconvolution differs in two important ways. The first is that we are forced to work with provisional case data, subject to revision at times in the future, as discussed earlier in Section 3.4.2. All of our experiments in what follows use properly-versioned data that would have been available as of the nowcast date. We use the notation $y^{(t)}_{\ell,s}$ to reflect the reported case rate in location ℓ at time s as of time t. Reported case data from JHU is available at a 1-day lag and therefore, as of time t, we only observe $y^{(t)}_{\ell,s}$ up through $s = t - 1$ (we use analogous superscript notation for all auxiliary signals and estimates). This means we can only produce deconvolution estimates $\hat{x}^{(t)}_{\ell,s}$ up through $s = t - 2$ (recall we exclude zero delays, in Assumption 2).

The second issue of note, in real-time deconvolution, is *right truncation*: in nowcasting at lag k, where k is small (compared to d), we are only able to carry out a "partial" deconvolution, as much of the needed information would come from case reports occurring in the future, past time the nowcast date t. Figure 3.4 gives an illustration. Thus, if we simply performed real-time deconvolution by solving the problem analogous to (3.5), using data that would have been available at time t,

$$\underset{x^{(t)}_\ell}{\text{minimize}} \sum_{s<t} \left(y^{(t)}_{\ell,s} - \sum_{k=1}^{d} \hat{p}^{(t)}_s(k)\, x^{(t)}_{\ell,s-k} \right)^2 + \lambda \big\| D^{(4)} x^{(t)}_\ell \big\|_1, \tag{3.6}$$

then we would find that the solution $\hat{x}^{(t)}_\ell = (\hat{x}^{(t)}_{\ell,s} : s < t)$ has highly volatile compo-

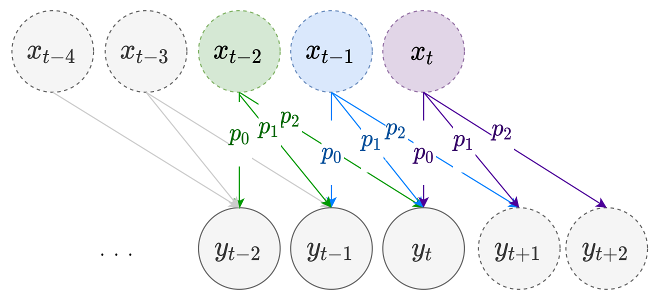


Figure 3.4: *Illustration of right truncation with a delay distribution of length 3 (which is taken to be stationary for simplicity). At the nowcast time t, some "part" of the latent signal x_t will appear in y_{t+1}, y_{t+2}; likewise, some "part" of x_{t-1} will appear in y_{t+1}.*

nents for s close to t.

The problem does not stop there; the truncation of data after the nowcast time t also affects estimation of the delay distribution itself. Most rows in the line list with an onset date of $s = t - k$, for small k, will only have a report date (and thus not appear in the line list) until after time t. This means that the estimate $\hat{p}_s^{(t)}$ of p_s given by the empirical distribution of all available line list data, with report date less than t, will be biased toward smaller lag values (i.e., it will place too little weight on larger lag values).

In the next two subsections, we work through each of these truncation issues in turn, by incorporating extra regularization around the right boundary into the criterion in (3.6), and estimating the delay distribution from truncated data using a Kaplan-Meier-like approach.

3.6.1 Incorporating extra regularization

We consider two forms of extra regularization to dampen the variability of trend filtering estimates toward the right boundary.

Natural trend filtering

A natural cubic spline places additional regularity on top of the cubic spline, by maintaining that the function be linear beyond the left and right boundary points of the underlying domain. Natural trend filtering proceeds in a similar vein, but operating in the space of discrete splines; see Tibshirani [2020]. Transporting this idea over to our real-time deconvolution problem (3.6), and applying it to the right boundary only,

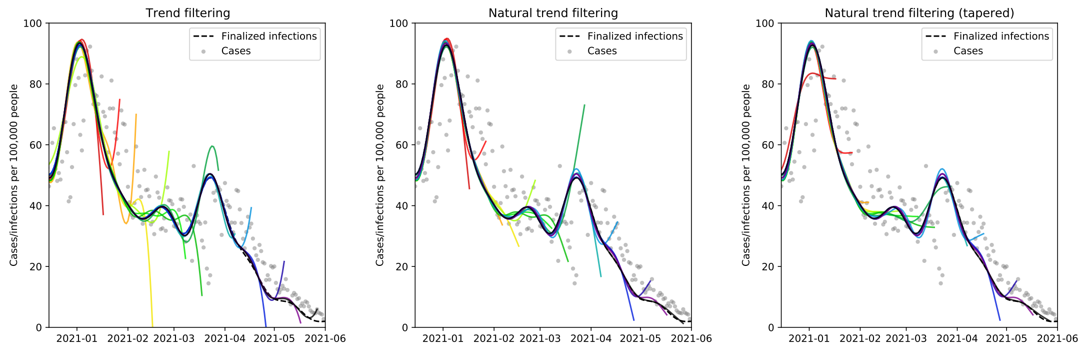


Figure 3.5: *Comparison of boundary behavior for real-time deconvolution in New York, displayed for a sample of different nowcast dates (where each colored curve traces out the deconvolution estimates for a different nowcast date). The black dashed line indicates finalized infections, estimated roughly three months after June 1, 2021.*

gives:

$$\begin{aligned} &\underset{x_\ell^{(t)}}{\text{minimize}} \; \sum_{s<t} \left(y_{\ell,s}^{(t)} - \sum_{k=1}^{d} \hat{p}_s^{(t)}(k)\, x_{\ell,s-k}^{(t)} \right)^2 + \lambda \big\| D^{(4)} x_\ell^{(t)} \big\|_1 \\ &\text{subject to} \;\; x_t^{(\ell)} - 2x_{t-1}^{(\ell)} + x_{t-2}^{(\ell)} = 0. \end{aligned} \tag{3.7}$$

The left and middle panels of Figure 3.5 demonstrate the improvement that the additional constraints in (3.7) can have on the boundary estimates, particularly during periods of dynamic change in the underlying case trajectories.

Tapered smoothing The right truncation phenomenon is not a binary one and there is increasingly less and less information available for deconvolution as we move the time index s up toward the nowcast date t. Therefore, we design a second penalty to add to the criterion in (3.7) to gradually increase the amount of regularization accordingly:

$$\begin{aligned} &\underset{x_\ell^{(t)}}{\text{minimize}} \; \sum_{s<t} \left(y_{\ell,s}^{(t)} - \sum_{k=1}^{d} \hat{p}_s^{(t)}(k)\, x_{\ell,s-k}^{(t)} \right)^2 + \lambda \big\| D^{(4)} x_\ell^{(t)} \big\|_1 + \gamma \big\| W^{(t)} D^{(1)} x_\ell^{(t)} \big\|_2^2 \\ &\text{subject to} \;\; x_t^{(\ell)} - 2x_{t-1}^{(\ell)} + x_{t-2}^{(\ell)} = 0, \end{aligned} \tag{3.8}$$

where $D^{(1)}v$ gives the first-order differences of a vector v, and $W^{(t)}$ is a diagonal matrix that is supported on the last d diagonal entries, these being (in reverse order, starting with the last entry):

$$\frac{1}{\sqrt{\hat{F}_{t-1}^{(t)}(k)}}, \quad k = 1, \ldots, d,$$

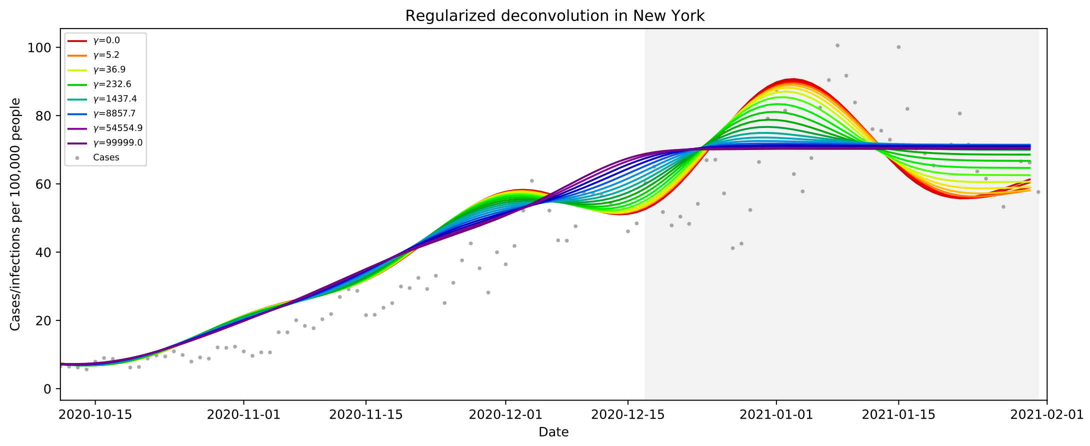


Figure 3.6: *Effect of the tapered smoothing penalty, as we vary the corresponding tuning parameter γ, for a single real-time deconvolution example with on nowcast date February 1, 2021. The gray region highlights the components on which the tapered smoothing penalty acts.*

where $\hat{F}^{(t)}_{t-1}$ is the cumulative distribution function (CDF) corresponding to the estimated delay distribution $\hat{p}^{(t)}_{t-1}$ at the most recent time $t-1$. The parameter $\gamma \geq 0$ controls the strength of the additional "tapered" penalty in (3.8), and we tune λ, γ with a two-stage cross-validation procedure:

1. fix $\gamma = 0$, and tune λ using 3-fold cross-validation, as before;
2. fix λ at the value in Step 1, and tune γ using 7-fold forward-validation: for $s = t-2, \ldots, t-8$, we solve the deconvolution problem with a working nowcast date of s, linearly extrapolate to impute an estimate at $s+1$, and then we reconvolve the solution vector along with this imputed point and measure error against observed cases at time $s+1$; the validation error is obtained by averaging these errors over the iterations $s = t-2, \ldots, t-8$.

Figure 3.6 displays the effect of varying γ on the solution in (3.8), for a particular deconvolution example, to give a qualitative sense of the role of the tapered penalty. Furthermore, the right panel in Figure 3.5 demonstrates the benefit this penalty can provide in nowcasting.

Lastly, and importantly, Figure 3.7 quantifies the improvement offered by the additional regularization mechanisms, in terms of mean absolute error (MAE) measured against finalized infections in nowcasting at a k-day lag, for each $k = 2, \ldots, 10$. This is averaged over all locations and every 10th nowcast date in the evaluation set. We see a considerable improvement in both the natural trend filtering and tapered smoothing modifications, with the biggest improvement occurring when the two are combined as

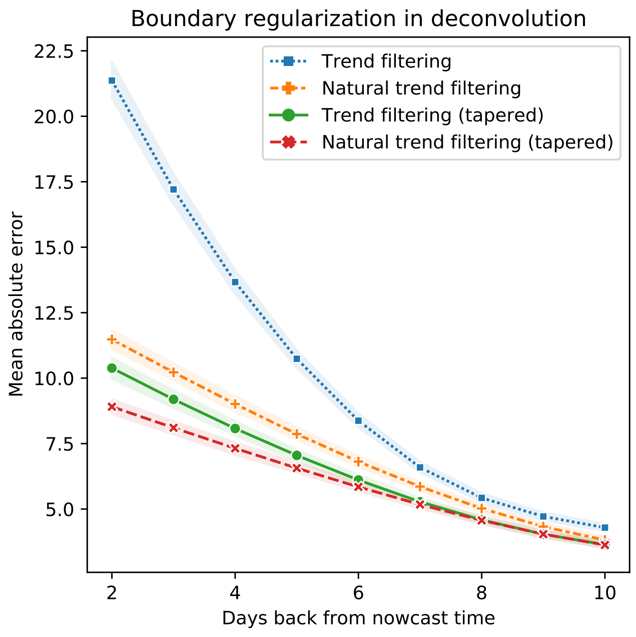


Figure 3.7: *Comparing regularization approaches by MAE for nowcasting (the shaded bands here and henceforth, in all MAE figures, correspond to 95% bootstrap confidence intervals.) Both approaches for additional regularization give a huge improvement on trend filtering. The biggest improvement comes from combining the two approaches.*

in (3.8), and hence we stick with this framework in what follows.

3.6.2 Adjusting the delay distribution for truncation

Now we propose an iterative adjustment to the empirical distribution of truncated line list data in order to overcome the truncation bias. To develop intuition, we first describe the problem using a simple abstraction, formulate a general solution, and then we translate this back over to our particular setting.

KM-adjustment under truncation

Suppose p is a distribution that is supported on $\{1, \dots, d\}$, and we observe independent random draws that we can partition into two sets: $\mathcal{D}_1$ and $\mathcal{D}_2$, where $\mathcal{D}_2$ contains draws from p and $\mathcal{D}_1$ contains draws from p conditional on the random variable lying in $[1, v_1]$, for a fixed $v_1 \in \{1, \dots, d\}$. Denote by $\hat{p}_{\mathcal{D}}$ the empirical distribution based on a data set $\mathcal{D}$. Clearly $\hat{p}_{\mathcal{D}_2}$ is unbiased for p, but $\hat{p}_{\mathcal{D}_1}$ is generally biased (it always places zero mass above v_1), and thus the pooled estimate $\hat{p}_{\mathcal{D}_1 \cup \mathcal{D}_2}$ would be biased as well.

To build a more informed estimate based on the pooled sample, the intuition is as follows. First, observe that the only way we can estimate $p(k)$ for $k > v_1$ is by using $\mathcal{D}_2$. Then, this gives an estimate of $S(v_1) = \sum_{k>v_1} p(k)$, the survival function of p at v_1, and we can estimate $p(k)$ for $k \leq v_1$, denoting $V \sim p$, by observing that

$$p(k) = \mathbb{P}(V = k \,|\, V \leq v_1)(1 - S(v_1)).$$

where we estimate $\mathbb{P}(Z = k | V \leq v_1)$ using the empirical distribution over the set $\mathcal{D}_1 \cup \mathcal{D}_2 \cap [1, v_1]$. In other words, we construct our distribution estimate $\bar{p}$ using two steps:

1. define $\bar{p}(k) = \hat{p}_{\mathcal{D}_2}(k)$ for $k > v_1$, and also $\bar{S}(v_1) = \sum_{k>v_1} \bar{p}(k)$;
2. define $\bar{p}(k) = \hat{p}_{\mathcal{D}_0}(k)(1 - \bar{S}(v_1))$ for $k \leq v_1$, where we let $\mathcal{D}_0 = \mathcal{D}_1 \cup \mathcal{D}_2 \cap [1, v_1]$.

We can readily generalize the above to a setting in which we observe N data sets, with varying levels of truncation:

$$\mathcal{D}_i \text{ contains draws } V \sim p \,| V \leq v_i,\, i = 1, \ldots, N, \tag{3.9}$$

where $1 \leq v_1 < \cdots < v_N = d$, and we set $v_0 = 0$ for notational simplicity. To construct an estimate of p based on all the samples, we proceed iteratively as before: first we estimate $p(k)$ for $k > v_{N-1}$ based on the data in $\mathcal{D}_N$, then we estimate $p(k)$ for $k \in (v_{N-2}, v_{N-1}]$ based on data in $\mathcal{D}_{N-1} \cup \mathcal{D}_2 \cap [1, v_{N-1}]$, and so on. Algorithm 3.2 spells out the procedure in full.

The algorithm just derived may be seen as Kaplan-Meier-like, in the sense that it is motivated by the decomposition

$$p(k) = \mathbb{P}(V = k \,|\, V \leq v_i)(1 - S(v_i)), \quad k \in (v_{i-1}, v_i].$$

We use an unbiased plug-in estimate for each term in the product above based on the appropriate data. The Kaplan-Meier estimator has a similar plug-in foundation [Kaplan and Meier, 1958], so we refer to our approach as the *KM-adjusted estimator* of the distribution under truncation.

Application to CDC line list

Porting the last idea over to the CDC line list, we can use it to estimate the delay distribution at time s using the line list as of time t. Note that if $s < t - d$ then we can still use Algorithm 3.1, as there is no truncation issue whatsoever. However, if $s \geq t - d$, then we would need to apply the KM-adjusted estimator, because we would be using the rows in the line list whose onset date is at or shortly before s, but are only able to see those whose report date is at most $t - 1$ (thus would have been available at

Algorithm 3.2: Distribution estimation under sequential truncation

```
Input: Data sets and truncation limits $\mathcal{D}_i$ and $v_i$, for $1,\dots,N$, as in (3.9).
Output: Estimated probabilities $\bar{p}(1),\dots,\bar{p}(d)$.
Initialize $\bar{S}(d) = 0$.
for $i = N,\dots,1$ do
    Set $\mathcal{D}_0 = \bigcup_{j=i}^N D_j \cap [1, v_i]$.
    Compute $\bar{p}(k)$, for $k \in (v_{i-1}, v_i]$ based on the empirical distribution of data
      in $\mathcal{D}_0$ and an estimate of the survival function at $v_i$:
        $\bar{p}(k) = \hat{p}_{\mathcal{D}_0}(k)(1 - \bar{S}(v_i)), \quad k \in (v_{i-1}, v_i]$.
    Compute an estimate of the survival function at $v_{i-1}$:
        $\bar{S}(v_{i-1}) = \bar{S}(v_i) + \sum_{k \in (v_{i-1}, v_i]} \bar{p}(k)$.
end
Return $\bar{p}(1),\dots,\bar{p}(d)$.
```

time t). After making this adjustment to the empirical distribution, we apply gamma smoothing as before. This is detailed in Algorithm 3.3.

Figure 3.8 compares the KM-adjusted and naive estimates of the delay distribution, Algorithm 3.3 versus Algorithm 3.1 applied directly to $\mathcal{D}^{(t)}$, the line list available at each nowcast date t. In terms of ℓ_1 distance, measured to the finalized delay distribution estimate computed retrospectively (based on the full untruncated line list), and averaged over all nowcast dates in the evaluation period, we see that the KM-adjustment greatly improves the accuracy at all lags $k = 2,\dots,10$ (where $k = t - s$, the difference between the nowcast and working onset dates).

3.6.3 Shortening the deconvolution window

Lastly, we investigate shortening the window used in the regularized deconvolution problem (3.8) so that we use only a window length of w days before t:

$$\begin{aligned} \underset{x_\ell^{(t)}}{\text{minimize}} \quad & \sum_{s \in [t-w, t)} \left(y_{\ell,s}^{(t)} - \sum_{k=1}^{d} \hat{p}_s^{(t)}(k)\, x_{\ell,s-k}^{(t)} \right)^2 + \lambda \left\| D^{(4)} x_\ell^{(t)} \right\|_1 + \gamma \left\| W^{(t)} D^{(1)} x_\ell^{(t)} \right\|_2^2 \\ \text{subject to} \quad & x_t^{(\ell)} - 2x_{t-1}^{(\ell)} + x_{t-2}^{(\ell)} = 0, \end{aligned} \tag{3.10}$$

Algorithm 3.3: Delay distribution estimation in real-time

```
Input: Nowcast time $t$, working onset time $s$, support size $d$, window size
       $w = 2d$, truncated line list $\mathcal{D}^{(t)}$ with onset dates $a_i$ and report dates $b_i$
       such that $b_i < t$.
Output: Estimated delay probabilities $\hat{p}_s^{(t)}(1), \dots, \hat{p}_s^{(t)}(d)$.
if $s < t - d$ then
    Return probability estimates from Algorithm 3.1 (setting $t = s$ and
      $\mathcal{D} = \mathcal{D}^{(t)}$ in the notation of that algorithm).
end
Set $N = d - (t - s) + 2$.
for $i = 1, \dots, N - 1$ do
    Define
        $\mathcal{D}_i = \{b_i - a_i : a_i = s - i + 1\}$
        $v_i = t - s + i - 2.$
end
Define $\mathcal{D}_N = \{b_i - a_i : a_i \in (s - w, t - d)\}$ and $v_N = d$.
Use Algorithm 3.2 (applied to $\mathcal{D}_i$, $v_i$, $i = 1, \dots, N$) to compute probability
  estimates $\bar{p}_t(1), \dots, \bar{p}_t(d)$.
Fit a gamma density to $\bar{p}_t(1), \dots, \bar{p}_t(d)$ using the method of moments
  (matching the mean and variance).
Discretize this gamma density to the support set $\{1, \dots, d\}$, call the result
  $\hat{p}_t(1), \dots, \hat{p}_t(d)$, and return these probabilities.
```

As we are mainly interested in the components of the solution $\hat{x}_s^{(t)}$ for s close to t, shortening the training window is computationally advantageous and should not change the behavior of the solution very much for s close to t.

Figure 3.9 compares (3.10) with $w = 2d$, $w = 4d$, and "all-past", which is the original problem (3.8), in terms of mean absolute error (MAE) measured against finalized infections in nowcasting at a k-day lag, for each $k = 2, \dots, 10$. This is averaged over all locations and every 10th nowcasting date in the evaluation set. The performance is basically identical for window lengths $2d$ and $4d$, and though all-past may appear to have the slightest advantage, this does not warrant the extra computation, hence in what follows we stick to (3.10) with a window length $w = 2d$ as our real-time deconvolution estimator.

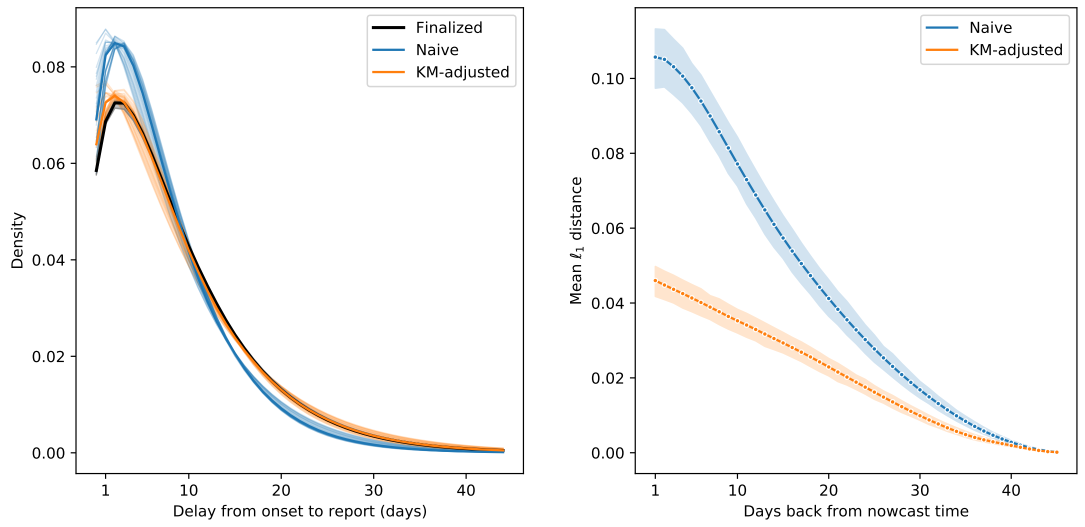


Figure 3.8: *Left: estimated delay distributions overlaid for all nowcast dates in the month of November 2020, when $s = t - 1$ (working onset date one day before the nowcast date). Right: mean ℓ_1 distance to finalized estimate of the delay distribution, as a function of the lag $k = t - s$.*

3.7 Leveraging auxiliary signals

The indicators enumerated in Section 3.4 have displayed impressive correlations to reported COVID-19 cases [Reinhart et al., 2021], and moreover, demonstrated an ability to improve the accuracy of case forecasting and hotspot prediction models [McDonald et al., 2021]. In this section, we describe how to use each indicator to build a real-time *sensor* that estimates the latent infection rate, and how to fuse such estimates together into a single nowcast.

3.7.1 Sensor models

At each prediction time t, for each location ℓ, and for each of the five indicators (abbreviated CHNG-COVID, CHNG-CLI, DV-CLI, CTIS-CLIIC, and Google-AA), we will train a model to predict in real-time latent infections from indicator values. Let $\hat{x}^{(t)}_{\ell,s}$ denote the solution at time s in problem (3.10), which represents our best estimate of the latent infection rate at time s as of time t from deconvolution of case rates alone.

We use $u^{i,(t)}_{\ell,s}$ to denote the value of indicator i at time s and location ℓ, as of time t. We fit a simple linear model to predict latent infections from indicator values by

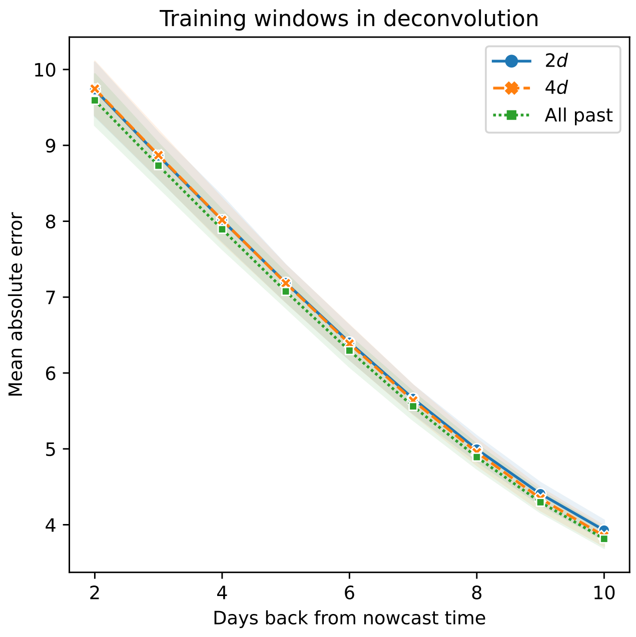


Figure 3.9: *Comparing window lengths used in regularized deconvolution by MAE for nowcasting. The performance is very similar throughout.*

solving

$$\underset{\beta_0,\beta_1}{\text{minimize}} \sum_{s=t-d}^{t-\tilde{k}_i} w_s^{(t)} \left(\hat{x}_{\ell,s}^{(t)} - \beta_0 - \beta_1 u_{\ell,s}^{i,(t)}\right)^2, \tag{3.11}$$

which is a weighted linear regression over the time period $[t-d, t-\tilde{k}_i]$, where $\tilde{k}_i = \max\{k_i, 2\}$ and k_i denotes the lag at which indicator i is available. This is:

- $k_i = 1$ for CTIS-CLIIC and Google-AA[3]; and
- $k_i = 4$ for the claims-based indicators, due to heavy revision or "backfill" over the first several days in the underlying claims data after an outpatient visit date [Reinhart et al., 2021].

Notice that, as defined, $\tilde{k}_i$ is the lag at which *both* the deconvolution estimate of infection rate and auxiliary signal i are available, which is the data we need to fit the linear sensor model (response and covariate data, respectively).

[3]Our treatment of Google-AA is different from the rest. Google's team did not start publishing this signal until September 2020, and the historical latency of this signal was sporadic, but was often longer than 1 week. However, unlike (say) the claims-based signals, revisions are never made after initial publication, and the latency of the signal is not an unavoidable property of the data type, and therefore we use finalized signal values, with a 1-day lag, in our analysis.

The observation weights in (3.11) are given by

$$w_{t-k}^{(t)} = \hat{S}_{t-1}^{(t)}(k-1), \quad k = 1, \dots, d.$$

Here $\hat{S}_{t-1}^{(t)}$ is the survival function of $\hat{p}_{t-1}^{(t)}$, the estimated delay distribution from the most recent time point $t-1$. We define $\hat{S}_{t-1}^{(t)}(1) = 1$, corresponding to the exclusion of 0-day delays. This scheme upweights the more recent estimates (responses in the regression) of latent infections as they contain more timely information for nowcasting (assuming that the right-truncation bias has been effectively mitigated in the deconvolution step).

Given the solution $\hat{\beta}_{\ell,0}^{i,(t)}, \hat{\beta}_{\ell,1}^{i,(t)}$ in (3.11), we then define a sensor—to reiterate, a sensor is just a prediction from the fitted linear model—based on indicator i, for time s and location ℓ, as of time t, as:

$$z_{\ell,s}^{i,(t)} = \hat{\beta}_{\ell,0}^{i,(t)} + \hat{\beta}_{\ell,1}^{i,(t)}\, u_{\ell,s}^{i,(t)}. \tag{3.12}$$

This sensor is available up until $s = t - k_i$. For the CTIS-CLIIC and Google-AA sensors, the lag is $k_i = 1$, smaller than the inherent lag of 2 in the deconvolution estimate.

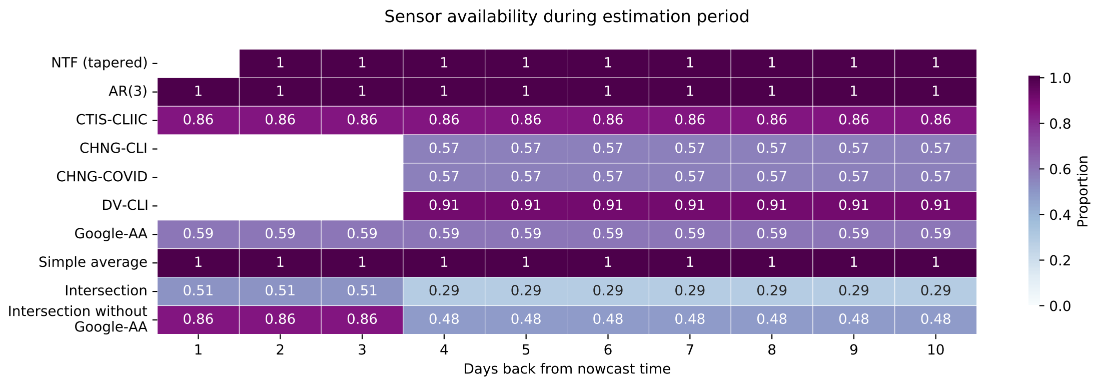
In brief, each sensor model takes a certain indicator and transforms it—using a location-specific and time-varying mapping—to the scale of local infection rates. While this mapping is simple (based on linear regression), it is also highly nontrivial, as it inherently accounts for geographic biases and nonstationarity.

Finally, in addition to defining sensors based on (3.11), (3.12) for each of the five auxiliary sensors, we also define a sixth sensor based on a 3rd order autoregressive model trained on $\hat{x}_{\ell}^{(t)} = (\hat{x}_{\ell,s}^{(t)} : s < t)$. It is constructed exactly as in (3.11), (3.12) (same weights and same training window). Henceforth we abbreviate it AR(3).

3.7.2 Sensor missingness

To be clear (3.11), (3.12) are to be implicitly understood as performed over observed (non-missing) indicator values. If an indicator value is missing at a particular location and time, then we drop it from the training set in (3.11), and do not produce a corresponding sensor value in (3.12). For a summary of missingness in the sensors, see Figure 3.10.

In general, an indicator will be missing when there is insufficient underlying data (from surveys, medical claims, etc.) to form a reliable signal value at a given location and time. However, the situation is different for the Google-AA indicator: here missingness occurs because the COVID-19 search trends data set is released after using a differential privacy layer [Bavadekar et al., 2020], and a missing value means that the level of noise added for privacy protection is high compared to the search count. Therefore we impute missing Google-AA signal values by zeros in our analysis; we do this unless the



	1	2	3	4	5	6	7	8	9	10
NTF (tapered)		1	1	1	1	1	1	1	1	1
AR(3)	1	1	1	1	1	1	1	1	1	1
CTIS-CLIIC	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86	0.86
CHNG-CLI				0.57	0.57	0.57	0.57	0.57	0.57	0.57
CHNG-COVID				0.57	0.57	0.57	0.57	0.57	0.57	0.57
DV-CLI				0.91	0.91	0.91	0.91	0.91	0.91	0.91
Google-AA	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59	0.59
Simple average	1	1	1	1	1	1	1	1	1	1
Intersection	0.51	0.51	0.51	0.29	0.29	0.29	0.29	0.29	0.29	0.29
Intersection without Google-AA	0.86	0.86	0.86	0.48	0.48	0.48	0.48	0.48	0.48	0.48

Figure 3.10: *Proportion of observed (non-missing) sensor values over the evaluation period from October 1, 2020 to June 1, 2021, and over all locations, as a function of lag* $k = 1, \ldots, 10$. *(NTF refers to the real-time deconvolution estimator, and simple average refers to the sensor fusion method that averages all available sensors.) The bottom two rows reflect the intersection of location-time pairs for which all data—deconvolution estimates and sensors—are available for that given lag, with and without including the Google-AA sensor, since this sensor has a large amount of individual missingness. Each intersection at each given lag* k *is restricted to data whose latency is not greater than* k. *For example, the bottom leftmost cell computes the porportions of locations and dates at which AR(3), CTIS-CLIIC, and the simple average are concurrently available.*

Google-AA signal was missing for a particular location and *all* times in the evaluation period, in which case we leave it as missing for this location entirely.

3.7.3 Sensor fusion

Sensor fusion, in its broader definition, refers to the process of assimilating data sources, each of which ideally contains complementary information, in order to produce more accurate estimates or predictions. Sensor fusion falls into the general class of ensemble methods, and the sensors constructed in the previous section can be thought of as base learners, to be subsequently combined.

We consider the following five ensemble methods. In each case, we describe how to form the estimate at time s and location ℓ as of time t. Though not explicitly stated, it is to be implicitly understood that all sensor values are as of time t as well.

1. Simple average: the average of available sensors at time s and location ℓ.
2. Simple regression: the prediction from a linear regression model at time s and location ℓ, fit to available sensors over the training period at location ℓ.
3. Ridge: the prediction from a ridge regression model at time s and location ℓ, fit

to available sensors over the training period and over locations j such that j, ℓ lie in the same U.S. state (including the state sensor itself).

4. Lasso: same as in the last item, but using the lasso instead of ridge regression.
5. KF-SF: the Kalman-filter-inspired method for sensor fusion from Farrow [2016], Jahja et al. [2019] and detailed in Section 2.1.2, with covariance shrinkage, and operating on the geographical hierarchy within each U.S. state.

Methods 2–5 are trained on the most recent $2d$ time points, and 3–5 are tuned using 7-fold forward validation, where we allow them to choose a lag-specific tuning parameter. Methods 1–2 are "simple" in the sense that for nowcasting at a location ℓ they use sensors from ℓ only. Methods 3–5 are more sophisticated in that they pool information across locations within the same state.

The KF-SF method requires a proper geographical hierarchy and thus we create "rest-of-state" jurisdictions by aggregating the remaining counties (outside of the top 200 counties nationally) within each state, and to run KF-SF, we create an AR(3) sensor at these rest-of-state locations (since one sensor at each location is sufficient). It is worth noting that, as shown in Jahja et al. [2019], KF-SF bears a close connection to ridge in Model 4: it is in fact equivalent to a modified ridge optimization problem that imposes additional linear constraints.

3.8 Evaluation

We now evaluate nowcasting performance over all locations and all but every 10th nowcasting date in our evaluation period from October 1, 2020 to June 1, 2021. (We do this because it gives us a "pure" test set, since every 10th nowcasting date was already used to choose the real-time deconvolution methodology in Section 3.6.) As before, we compare to finalized estimates of infection rates computed via retrospective deconvolution, as in Section 3.5.

For the purposes of making fair comparisons, in every analysis (figure) that we present, we only aggregate over the intersection of nowcasts dates and locations at which the particular estimates under consideration—coming from real-time deconvolution, individual sensor models, or sensor fusion—are all available. Abiding by this rule leads us to examine several different ways of stratifying results, as the full intersection is fairly sparse (see the second-to-last row in Figure 3.10). In particular, we consider the following two dimensions used to define strata:

- inclusion of Google-AA or not;
- inclusion of all claims-based sensors (CHNG-CLI, CHNG-COVID, and DV-CLI) or not.

To be explicit, when we say we do not “include” certain sensors, it means both that we ignore results from their individual sensor models (in computing the common intersection of available nowcast dates and locations), and *also* that we exclude them in running the sensor fusion methods.

In the following subsections, we first examine the performance of individual sensor models and a certain sensor fusion method (the simple average) compared to real-time deconvolution, and then examine the relative performance of the different sensor fusion methods.

3.8.1 Performance of sensors and sensor fusion

We begin by comparing the MAE of nowcasts from natural trend filtering (NTF) using tapered smoothing, as in (3.10) (the real-time deconvolution estimator chosen based on the analysis in Section 3.6) to those from individual sensor models and the simple average sensor fusion method. Despite its simplicity, the simple average appears to be the best-performing sensor fusion method overall (details in the next subsection), and so we stick with it as the de facto sensor fusion method in this subsection. The results here do not include Google-AA; results including Google-AA are shown in Appendix B.2.

Figure 3.11 displays the MAE from various methods as a function of lag k. The left and right panels do not and do include the claims-based sensors, respectively. In either case, we see that up until lag 6, all sensors outperform the real-time deconvolution estimate from NTF. The simple average of all sensors improves accuracy even further, and achieves the best MAE for all lags up through lag 6. We recall that NTF (with tapered smoothing) itself already provides a huge increase in accuracy over the more naive method for real-time deconvolution given by applying trend filtering without extra boundary regularization (Figure 3.7). At lag 7, the NTF estimate catches up to about equal accuracy, and then surpasses sensor fusion and all sensors in accuracy at lag 8 and onward. An interpretation for this: right truncation ceases to be a significant problem past lag 7, and thus we are better off performing deconvolution directly in order to estimate infections more than a week into the past.

Figure 3.12 displays the empirical distributions of ranks of nowcast errors coming from each method, computed with respect to each other, over common nowcast tasks (defined by a location-date-lag triplet). For example, in a particular nowcast task, we assign a rank of 1 to the method with the smallest absolute error for that nowcast task. The left panel again excludes claims-based signals, and the right panel includes them. The striking feature in either panel, particularly the bottom panel, is that the simple average has a highly distinctive distribution of ranks—it is rarely the best method, but never the worst. While this is not particularly surprising (averaging random

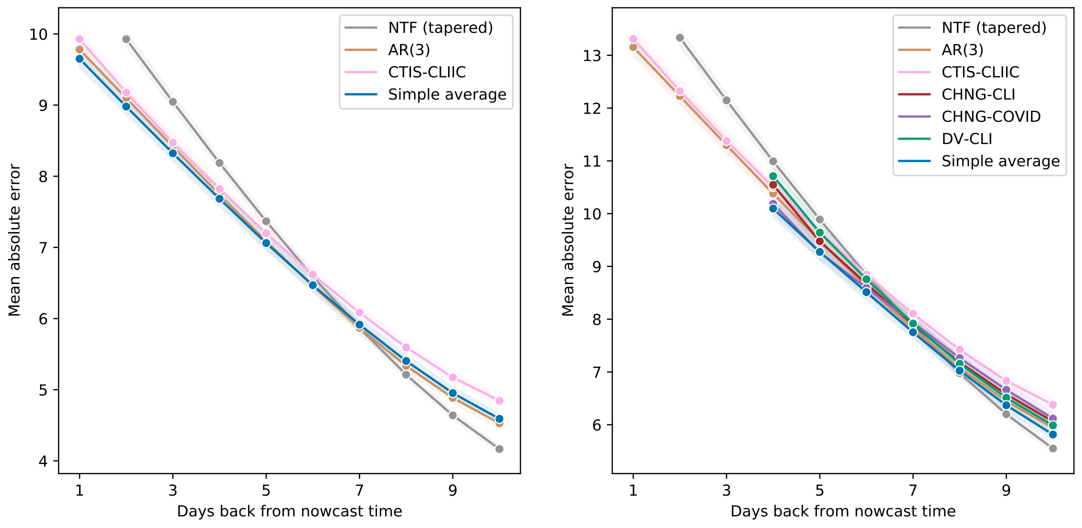


Figure 3.11: *Comparing NTF to individual sensor models and the simple average sensor fusion method by MAE for nowcasting. The left panel excludes the claims-based sensors, whereas the right includes them. For lags smaller than 7, all methods improve upon NTF (with tapered smoothing), with simple average being the best among them.*

variables tends to be variance-reducing, as long as the variables are not too correlated), it also points to a key property of sensor fusion—a certain kind of robustness, beyond accuracy.

3.8.2 Relative performance of sensor fusion methods

We now compare the various sensor fusion methods to each other. The results here do not include claims-based signals; results including claims-based signals are shown in Appendix B.2. Figure 3.13 displays the MAE of the various sensor fusion estimates, but divided up into three panels, defined by averaging over small, medium, and large states (the figure caption provides more details). Recall that for the lasso, ridge, and KF-SF approaches, a model in a particular county is fit using the sensors from other counties in the same state. Larger states have more pooling of information across locations and present a greater potential for gains in accuracy. We see that the simple average method is typically the best sensor fusion method at each lag, but for medium and large states, KF-SF catches up with it and is just about as accurate.

Figure 3.14 displays the relative ranking of sensor fusion methods. The simple average and KF-SF methods appear the most favorable (often the best, and less so the worst), followed by lasso, then ridge, and lastly simple regression (most often the worst).

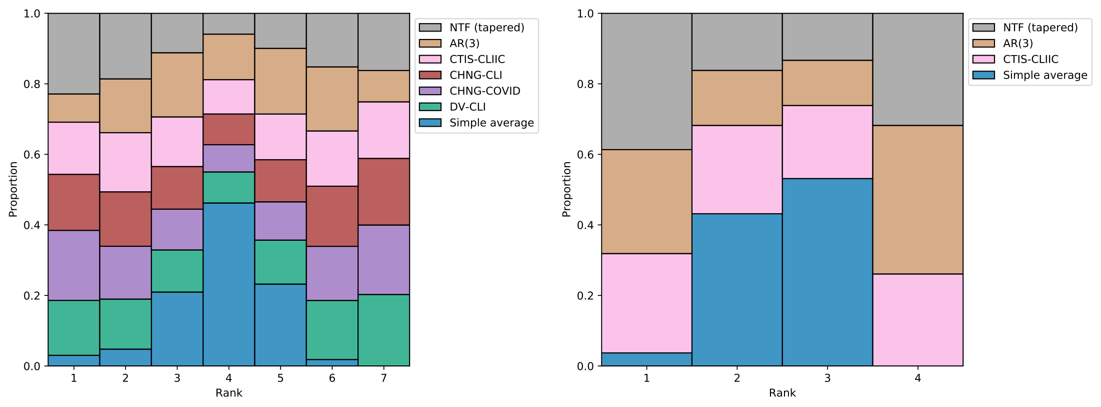


Figure 3.12: *Comparing NTF to individual sensor models and the simple average sensor fusion method by relative ranks over common nowcast tasks. The left panel excludes all claims-based sensors and considers lags 1–5, whereas the right panel includes them and considers lags 4–9 (the first 5 lags at which all methods are available, in either case). The simple average exhibits striking consistency: it is rarely the best, but also never the worst.*

3.8.3 Post hoc smoothing

As we saw, sensor fusion provides a real-time improvement on pure deconvolution up until about a 7-day lag, and past that point, the deconvolution estimates appear stable enough that sensor fusion becomes unnecessary. While the quantative benefit of sensor fusion for small lags is clear, sensor fusion is also lacking in the following qualitative aspect: its estimates do not always appear visually smooth across time (this is because the sensors themselves need not be smooth over time, and furthermore, sensor fusion may end up using a different subset of sensors at each lag, creating additional jaggedness).

We employ a simple technique to simultaneously impose smoothness on the final sensor fusion trajectory (by trajectory, we mean the sequence of estimates over time) and incorporate the stable NTF estimates at larger lags. There are two steps, as follows:

1. Concatenate together the sensor fusion estimates and NTF estimates, where we take the sensor fusion values for lags smaller than 6, and NTF estimates for lags greater than 8. For the three values at $k = 6, 7, 8$, we take the average estimate from both methods.
2. Perform smoothing on the concatenated trajectory; we apply a simple quadratic trend filtering layer [Tibshirani, 2014], where the weights at "joining" points $k = 6, 7, 8$ are doubled. To be clear, the trend filering problem here simply performs local smoothing on the signal, and not regularized deconvolution. We fix the value of the smoothing parameter to a small value relative to the

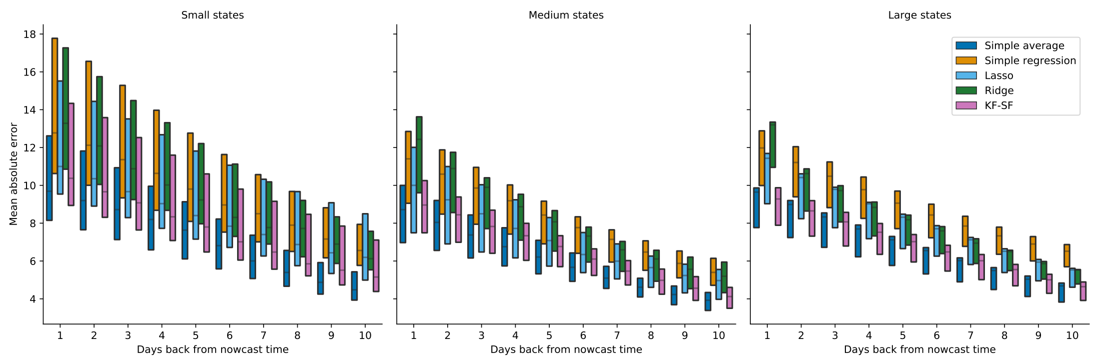


Figure 3.13: *Comparing sensor fusion methods by boxenplots of nowcasting errors (each box conveys the level 25%, 50%, and 75% quantiles of the absolute error distribution.) The three panels average over small (containing less than 5 locations), medium (between 5 and 14 locations), and large (more than 15 locations) states. Simple average performs generally the best throughout, but KF-SF catches up for medium and large states.*

magnitude of the infection rate.

We refer to this method as *smoothed sensor fusion*, as it retains all benefits from sensor fusion (namely improved accuracy at the right boundary and better latency) while maintaining the stability of the NTF approach as right truncation diminishes over time. Figure 3.15 visualizes the resulting curve, in green, where the red squares correspond to the three joining points. The jagged blue curve corresponds to the original sensor fusion estimate, which are visually difficult to reconcile. For completeness, we evaluate the mean absolute error of the smoothed sensor fusion estimates (see Figure B.5 in the appendix), and show that post hoc smoothing does not cost anything in terms of accuracy: it achieves the best performance over NTF and the original sensor fusion method. In later experiments, we use the smoothed sensor fusion trajectory as a replacement for our final estimates at time t.

3.8.4 Correlation to hospitalizations

The epidemiological modeling community considers hospitalization forecasting as one of its more important problems, as hospitalizations contribute a direct, quantifiable stress on health care systems. Accurate forecasts aid these systems in anticipating and preparing for surges and declines in admissions (e.g., through proactive resource allocation). A growing criticism of case reporting signals is that they are not *leading* indicators of hospital admissions; that is, the current number of reported cases are not necessarily indicative of future hospitalizations [Reich, 2021].

We can observe this in Figure 3.16, which plots the correlation of real-time case rates

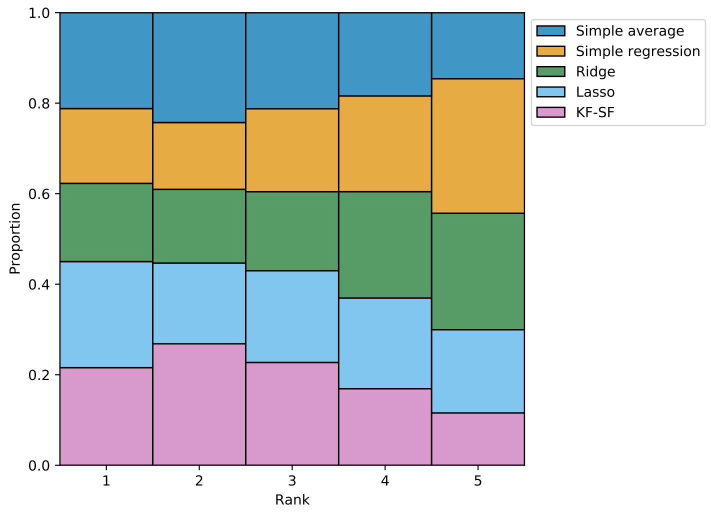


Figure 3.14: *Comparing sensor fusion methods by relative ranks over common nowcast tasks, and considering only lags 1–5. The simple average and KF-SF methods consistently perform in the top half, while simple regression is most often the worst.*

to (eventually observed) hospitalizations over the 50 US states across an evaluation period of October 1, 2020 to May 1, 2021 (212 days). Using the COVIDcast API [Reinhart et al., 2021], on each day, and for each location, we compute the correlation of finalized hospitalization incidence rates (count per 100,000 people) over the past 45 days to both the case reporting rate and estimated infection rate.

The vertical orange line marks the highest correlation attained with the case reporting rate, which is at lag 0 (where cases neither lead nor lag hospitalizations). In this sense, we can view case reporting rates as a signal that is almost *contemporaneous* to hospitalizations.

Notably, we see in the same figure that our infection estimates (corresponding to the blue line), *are* a leading indicator of hospitalization, and can attain nearly as good correlation at its peak 11 days lagged. This is a striking result, as we have taken case reports and meaningfully recovered an infection signal that is a useful predictor of future hospitalizations. To give another interpretation of Figure 3.16, we see that our infection estimates can provide information just as indicative to hospitalizations as case reports 6 to 7 days earlier (as the case correlation lagged 5 days is close to the highest infection correlation).

We turn now to comparing smoothed sensor fusion and NTF infection estimates. To determine if the addition of auxiliary signals is helpful, we take the estimates at each nowcast time at a fixed k over our evaluation period. We then compute a

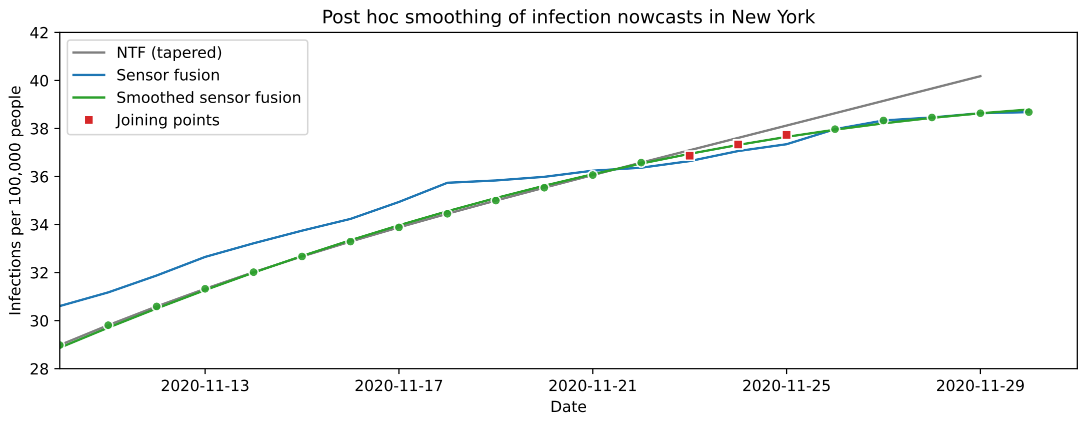


Figure 3.15: *Illustration of post hoc smoothing on infection nowcasts for New York as of December 1, 2020. The green points align closely with the NTF (tapered) estimates for large lags $k > 8$, and the sensor fusion estimates for small lags $k < 6$. The "joining" points at lags $k = 6, 7, 8$ are plotted by red squares, and fall in the middle of the NTF and sensor fusion curves.*

single correlation to finalized daily hospitalization incidence in that same period, and compute the average over the 50 US states. The result is shown in Figure 3.17, where each panel corresponds to k (the latency of the nowcast estimate). In the first row, we see that at $k = 2$ where both SF and NTF produce nowcasts, that SF has much better correlation over SF throughout. This is excellent news towards the use of sensors in improving the most recent nowcasts. As k increases, we notice the two infection curves becoming similar, and that at $k = 10$ the curves are essentially identical. This is unsurprising, as we saw previously that sensors provided the most information at small k, and decreased in utility for further away nowcasts. Furthermore, we note that the SF estimates after $k > 8$ were replaced with estimates from NTF, and thus we expect no meaningful difference at these lags.

3.9 Discussion

In this work, we proposed, implemented, and evaluated a framework for real-time estimation of new symptomatic COVID-19 infections from case reports. At time t, in order to nowcast the infection rate at time $t - k$ (for small values of k, such as $k = 1, 2, \ldots$), the main steps are to:

1. estimate a symptom-onset-to-case-report delay distribution using the most recent data available in a line list provided by the CDC;
2. perform regularized deconvolution on the most recent case data available from

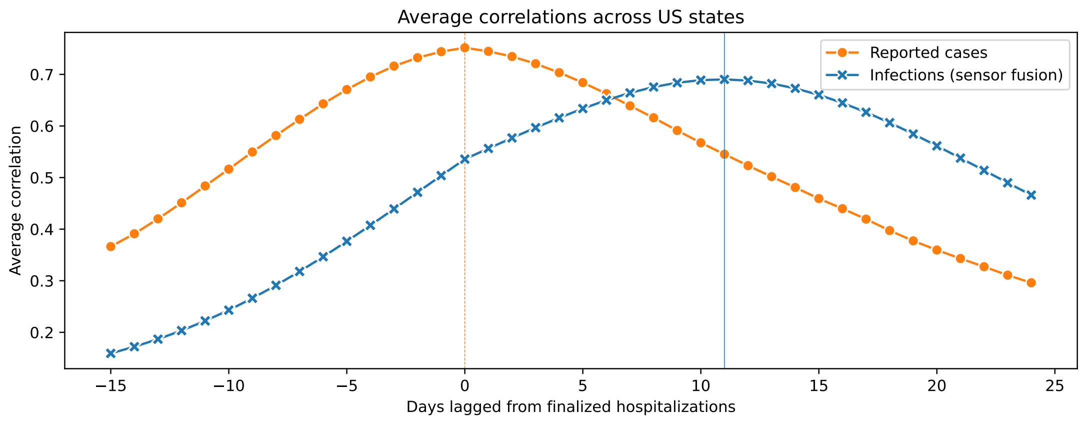


Figure 3.16: *Lagged and leading correlations of observed case reports and infection nowcasts (from smoothed sensor fusion), both available in real-time, to finalized hospitalization averaged over October 1, 2020 to May 1, 2021. The correlation is taken over a rolling window of* $d = 45$ *days, and compared to confirmed hospitalization admission rates pulled as of February 1, 2022. Both case and hospitalization signals are averaged over the past 7 days, which remove any spurious day-of-week artifacts. The highest correlations are marked by the dashed vertical lines, which indicate that case reports are best correlated with hospitalizations at 0 lag, whereas infections obtain the best correlation when lagged 11 days.*

JHU CSSE;

3. update models to track recent infection rates from various auxiliary signals (based on COVID-related data from medical insurance claims, online surveys, and Google searches), and fuse together the predictions from these models in order to stabilize recent estimates of infection rates.

In each step, we proposed methodological advances that improved the accuracy of our nowcasts, when measured against finalized infection rate estimates obtained by retrospective deconvolution (using data that would have only been available months later). While using auxiliary signals (step 3) did help in terms of accuracy and robustness, the additional regularization devices that we incorporated into real-time deconvolution (step 2) ended up providing the biggest benefit to accuracy.

To reiterate, we purposely defined our target of estimation to be symptomatic infections that would eventually show up in public health reports, allowing us to focus on developing and testing tools for real-time deconvolution and sensor fusion, with minimal assumptions (e.g., without a mechanistic model for disease spread). Estimating the number of true symptomatic infections at any point in time—whether or not they will appear in case reports—is of course a much harder problem. However,

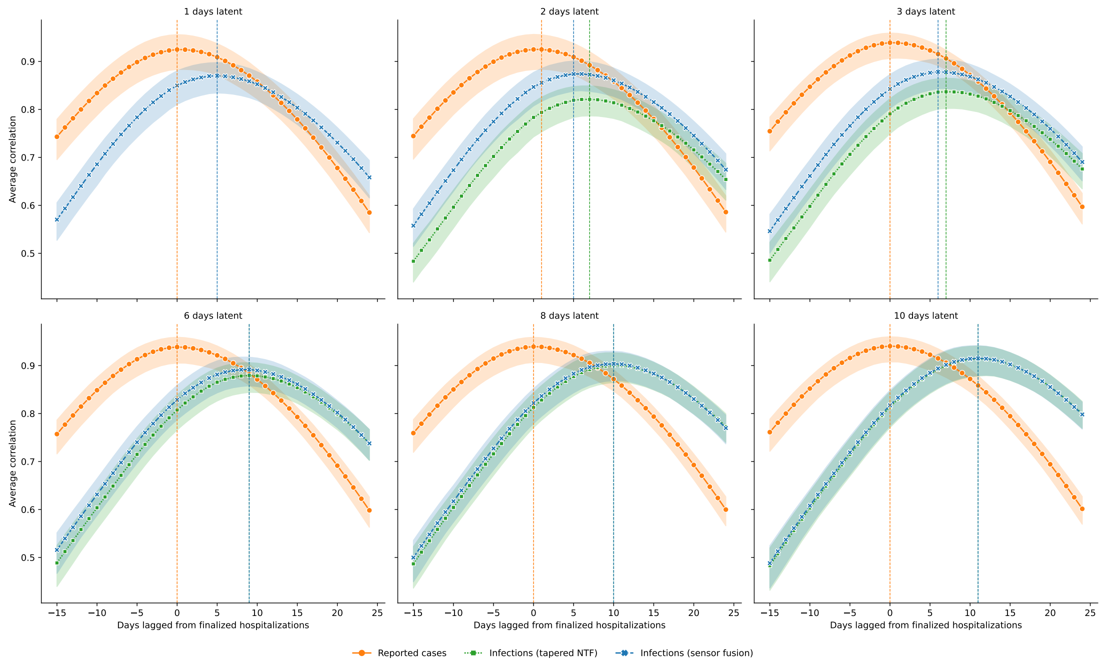


Figure 3.17: *Lagged and leading correlations of observed case reports and infection nowcasts to finalized hospitalization over October 1, 2020 to May 1, 2021. Each panel computes a single correlation for each of the 50 US states, using the estimate produced k days before each nowcast date in the evaluation period (a vector of length 212). The case report and finalized hospitalization signals are as described in Figure 3.16.*

our methodology may be seen as a contribution toward solving this larger problem in real-time; moreover, some simple post hoc corrections could be applied to our real-time estimates in order to adjust for confounding. For example, if $a_{\ell,t}$ is the fraction of untested symptomatic infections in location ℓ at at time t, which (say) is estimated from external data sources, then we could just multiply each element $\hat{p}^{(t)}_{\ell,s}$ of the delay distribution used in (3.10) by $b_{\ell,t} = 1/(1 - a_{\ell,t})$ in order to estimate *all* symptomatic infections from case reports. Due to the way we have set up the deconvolution problem (cross-validating over optimal choices of tuning parameters), this would be essentially equivalent to post-multiplying the nowcast $\hat{x}^{(t)}_{\ell,s}$ we already produce by $b_{\ell,t}$.

An important avenue for evaluating our methodology (beyond evaluating against finalized infection rate estimates or finalized hospitalization rates, as we do in this work), would be to reconvolve our real-time nowcasts of infection rates forward in time in order to predict future case rates, and evaluate these predictions against finalized case reporting data. Making and evaluating point predictions would be relatively straightforward, however, distributional forecasts are currently the standard in epidemiological forecasting (and also in COVID-19 forecasting), and adding a distributional layer to our nowcasts (and propagating this through the convolution operator) requires substantial new developments. To this end, we investigate a reconvolution framework for distributional evaluation in the next chapter.

Chapter 4

A Reconvolution Approach for Evaluation

> This work was done in collaboration with Daniel McDonald, James Sharpnack, and Ryan Tibshirani.

4.1 Introduction

In this chapter, we turn our attention to a reconvolution framework to evaluate nowcasts made through deconvolution. The following work comes as a natural sequel to Chapter 3, and we accordingly lay out our approach in the context of nowcasting the rate of newly onset COVID-19 infections.

We emphasize again that our motivation behind infection nowcasting is to produce accurate, real-time estimates of disease spread in order to inform the public health response during an ongoing epidemic. The trustworthiness of our estimates then, is paramount, and we seek to answer the question:

How well can we trust our infection estimates made in real-time?

Though this question underlies the entirety of our work in this chapter, it is broadly stated, and one can imagine numerous avenues to an answer. We dedicate this chapter to a reconvolution approach, where the latent state estimates are propagated onto the scale of an observable signal, namely, the finalized case incidence rates.

Given estimates of newly onset infection rates and reporting delay probabilities, we reconvolve (and to be clear, by "reconvolve" we simply mean to convolve the estimates—originally found by deconvolution—forward) infections to form predictions of the eventually-observed case rates. At a nowcast time t, we can easily recover point nowcasts of case rates up till t, as we have in hand all the necessary information from past infections that will eventually turn up in case reports. We say that these estimates come through a *full reconvolution*, where "full" refers to the completeness of information used to calculate the outcome. Calculating error (using, say, mean absolute error) to the true observed case rate is akin to examining the fitted residuals of a model; as we have trained on the case reports themselves, examining these errors can be seen as a measure of the goodness of fit found by our deconvolution method.

But are these infection nowcasts informative? To answer this question we turn to out-of-sample errors, and view evaluation as a *forecasting* task: how well can the infection estimates predict future cases? If our infection estimates are accurate, then by the convolutional relationship (3.1), each infection will eventually be reflected in a case report published later in the future. In following sections, we lay out *partial reconvolution*, a technique to propagate forward the infection estimates without imposing any parametric structure or assumptions apart from a locally constant delay distribution. As a counterpart to "full" reconvolution, the term "partial" indicates that only some fraction of the observed information is available to carry out the convolution. Partial reconvolution bypasses additional modeling done on either the infection or case reporting curve, and allows us to focus exclusively on understanding the amount of information captured in our nowcast solution.

In this work, we also move away from point predictions and propose sampling-based methods to generate distributional forecasts. Distributional forecasts are currently the standard in epidemiological forecasting, and aside from being useful when assessing the quality of our nowcasts, contain strictly more information (over point estimates) that is helpful for nuanced decision-making. Quantifying the uncertainty around the nowcasts is critical for guiding local health officials, who necessarily weigh many factors before enacting policy action. To this end, we lay out residual-based approaches to introduce stochasticity in both the deconvolution solution and simulated case forecast, which leads us to a better understanding of the stability and uncertainty surrounding our infection nowcasts. This, in turn, allows us to better answer our question of interest.

This chapter is organized as follows. In Section 4.2, we discuss our dimensions of evaluation, and cover relevant material needed to ground future sections. Section 4.3 provides an overview of related work. In Section 4.4, we describe a Monte Carlo approach to produce samples of perturbed infection solutions, and introduce partial reconvolution. A residual sampling approach for distributional forecasting around the reconvolved estimates is given in Section 4.5. Section 4.6 is the experimental section, which covers various evaluations and interpretations of our deconvolution estimates. We conclude in Section 4.7 with a discussion.

4.2 Preliminaries

In this preliminary section, we give an overview of the dimensions we will evaluate the nowcasts over; these dimensions align with the *partiality* of information (i.e. the amount of observed components) used in the calculation of an infection or case estimate. We note that, as in previous work, we will often write "symptomatic infections" as just "infections", where "infections" should also be be taken as shorthand for the symptomatic infection rate, for covenience (the issues surrounding asymptomatic infections is described in Section 3.2.2). Following this, we give a short recapitulation of the relevant deconvolution material, and outline the problem setup that motivates our proposed framework.

4.2.1 Dimensions of evaluation

In this work, we will stratify evaluation by two dimensions:

1. as a function of the lag k, where $k = t - s$ is the number of days away an estimate for time s is from the nowcast time t;
2. as a function of the forecasting ahead i, where $i = 1, 2, 3, \dots$ is the number of days ahead of an estimate for time s.

To motivate the first dimension, recall that in real-time nowcasting, we are subject to *right truncation*, the issue where outcomes from recent infections will only be revealed in the not-yet-observed case reports. Right truncation forces us to perform a "partial" deconvolution to estimate infections towards the right boundary, and infer the future contribution from case reports (see Figure 3.1 for a visualization, where the blue region highlights the issue). This implies that our most recent estimates, which are the most important for decision-makers, are generally the least reliable. To observe this trend, and to identify which models better mitigate this effect, we will bucket our errors over the lag k, which indexes the degree of right truncation for an estimate. In this notation, a nowcast with large k is largely unaffected (as it has observed a good amount of information), whereas a nowcast at small k, has inferred a great amount of its value.

The second dimension arises from the forecasting problem needed to perform evaluation, and has a sort of symmetry to the former dimension. Rather than missing case information, a reconvolved estimate is missing infection values needed to perform "full" reconvolution. To characterize this dimension, we group errors over the ahead i, where a case estimate for time $s + i$ (regardless of the nowcast time t, $t > s$) only has access to the infection estimates up until time s. Clearly, forecasts made at large aheads i must extrapolate further, and can be less reliable. To illustrate the most challenging case, given a maximum reporting delay of d days, the reconvolved estimate for the ahead $s + d$ has observed only one infection value—the infections that onset at time s, which have the potential to be reported d days later.

To summarize: we wish to understand how much we can trust nowcasts, and—as shown by our previous experiments—nowcasts made further back in the past are more stable and accurate to finalized infections estimates. This forms the basis for evaluating over the first dimension indexed by k. Any fixed k-day back infection nowcast will result in a case outcome in the future. We index this future path with the ahead i, our second dimension. To simplify our evaluations, we will pay attention to nowcasts with lag $k = 10$ or smaller (corresponding to the 10 most recent estimates), and calculate the sum of their error over all possible aheads $i = 1, 2, 3, \ldots$. We then provide analyses that fix the lag k, and show error as a function of i, which provides an alternative view of our results.

4.2.2 Review of COVID-19 setup

We now review the infection nowcasting setup, and solidify notation for our framework described shortly. For notational simplicity, we concern ourselves only for a single location. For a day t, let x_t denote the rate of newly onset infections and y_t the rate of newly reported cases, where for completeness we define "rate" as the count per 100,000 people (henceforth, we will simplify our writing by referring to x_t as "infections" and y_t as "cases", with all the qualifiers described previously).

We begin by reintroducing the following convolutional model:

$$\mathbb{E}[y_t \,|\, x_1, \ldots, x_s, s \leq t] = \sum_{s=1}^{t} \pi_t(s) x_s, \tag{4.1}$$

where $\pi_t(s)$ represents the probability of a case reported at t given infection onset at s. For each nowcast time t, we obtain the case rate reports at times $s < t$ from JHU CSSE [Dong, Du, and Gardner, 2020].

Recall that we estimate the distribution of the delay probabilities $\pi_t = (\pi_t(s) \,:\, s \leq t)$ using data from a de-identified patient-level surveillance line list dataset provided by the CDC [Centers for Disease Control and Prevention, COVID-19 Response, 2020a]. Given the realities of limited data availability (discussed in Section 3.5.2), we make three assumptions when estimating the delay distribution, reproduced here. First, we assume that an infection, once onset, is always reported within the following $d = 45$ days. Second, we assume that the infection will never be reported at a zero day delay; taken in combination with the first assumption, we can shorten and denote the sequence of delay probability estimates as $p_t(j), j = 1, \ldots, d$, where

$$p_t(j) = \mathbb{P}\big(\text{case report at } t \,|\, \text{onset at } t - j\big). \tag{4.2}$$

Our third assumption is that the delay probabilities are geographically-invariant, and in our experiments we use a nation-wide estimate across locations. This is a concerning assumption that arises out of data limitations, where there simply is not sufficient data to construct geographic-specific estimates. While we provide further reasoning and details regarding these assumptions in Section 3.5.2, we wish to reiterate that we do not think the third assumption to be a weakness of our methodology, as any candidate delay distribution (say, for a particular location where more data is available) can simply be plugged into the framework. Finally, we remind the reader that we employ a Kaplan-Meier-like algorithm (see Alg. 3.2) to generate the delay distribution estimates, which is one of our contributions to alleviate data censoring effects in real-time nowcasting.

As the previous chapter covered deconvolution in depth, we only mention here that we produce the initial infection estimates $\hat{x}_1, \dots, \hat{x}_{t-2}$ by solving a regularized deconvolution problem (3.10) applied on case reports $y_1, \dots, y_{t-1}$. In our experiments, we use *versioned* data, meaning that at time t, we only use data that was available at time t (this is also why we address data censoring in delay distribution estimation). Case reports from JHU CSSE are published on a 1-day lag, meaning the latest case observation available is y_{t-1}. However, by virtue of our third delay assumption (no infections are reported on the same day as onset), our latest deconvolution estimate has lag $k = 2$.

Our final major contribution towards deconvolution is sensor fusion, wherein we combine together contemporaneous auxiliary information to improve upon our deconvolution estimates. In the previous chapter, we describe the background and modeling of these various data sensors (also referred to as *indicators*, as they indicate COVID-19 activity) in great detail (see, in particular, Sections 3.4 and 3.7 for a review). We highlight that sensor fusion allows us to gain an additional day's estimate and produce nowcasts at only a 1 day latency in total (to be clear, our latest estimate is $\hat{x}_{t-1}$). This additional estimate is produced by fusing data sensors available at a 1-day lag; Figure 3.10 visualizes the availability of these signals as a function of latency, and we incorporate the CTIS-CLIIC and AR(3) signals for lag $k = 1$. As a final step, we apply post hoc smoothing as in Section 3.8.3, to produce our final infection nowcasts $\hat{x}_s$, $s < t$.

Notation

In following sections we will simplify our equations by introducing notation for a convolution matrix P, which has rows corresponding to the delay distribution p_s,

$s < t$, such that for a vector x

$$(Px)_s = \sum_{j=1}^{d} p_s(j) x_{s-j}.$$

Here, we leave the dimensions of P and x ambiguous, but it should always be clear from the problem context. This convolution formulation allows us to keep consistent notation across problems with different underlying dimensions.

In the previous chapter, we used superscripts to denote the *version* of an estimate or observation, for example, $\hat{x}_s^{(t)}$ denotes an estimate of incident infection rates on date s generated as of time t. In this section, and in following sections, we will omit these superscripts for notational simplicity (again providing clarification in text, if needed), but will occasionally emphasize versioned data through superscripts when appropriate.

4.2.3 Defining ground truth

In past evaluations we assessed our nowcasts to a "finalized" infection value, calculated by solving the deconvolution problem (3.5) with data issued several months after our final nowcast date. This waiting period allowed us to produce estimates that were not subject to the two critical issues present in real-time nowcasting (discussed in Section 3.6): provisional releases, where data is subject to revision over future time, and right truncation, described previously.

As we now turn to assessing nowcasts by cases, we again find two issues of note. First, while right truncation no longer poses a problem (at time t, we have estimated all infections needed to produce a case prediction at t), we must of course wait the span of the forecast horizon before our desired case data is available. Moreover, this data is still provisional, and may undergo revisions as delayed reports arrive, or are corrected. Therefore, we are still constrained to wait some period of time before making our evaluations.

The second issue is of a different nature; the current case signal we have been using (incidence rate of newly onset case reports) contains "day-of-week" effects, which refers to the phenomena where a signal exhibits systematic differences as a function of the day of the week. As an example in our context, case report collection may be delayed on weekends, and only published on the subsequent Monday, causing an artifical spike in incidence. Adjusting for day-of-week effects effects is a common problem in time series modeling (and in epidemiological settings where one models reporting streams, e.g., Reinhart et al. [2021], Rumack [2020]), and can require sophisticated modeling techniques. To make the issue even worse, day-of-week effects appear to vary by

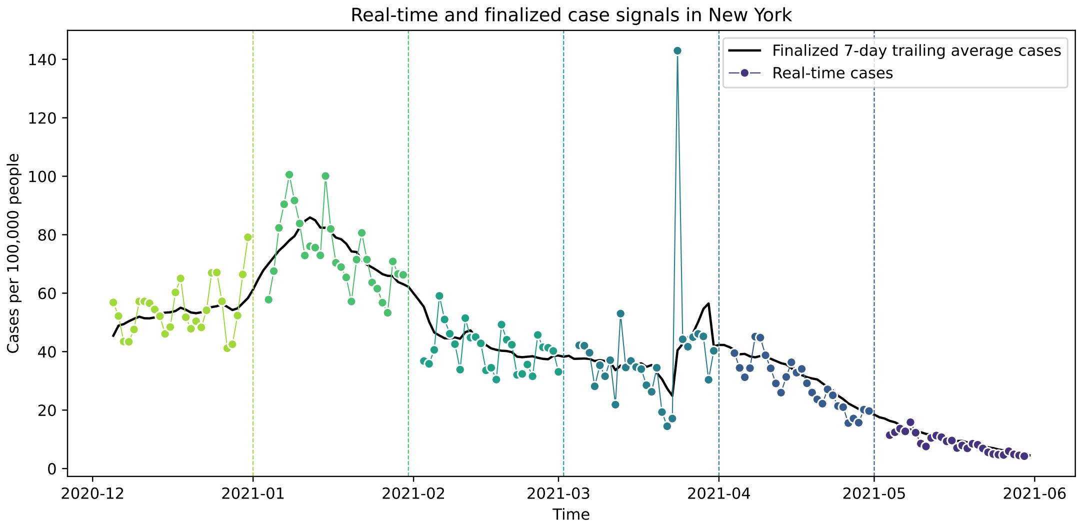


Figure 4.1: *Comparison of real-time case incidence rates to finalized 7-day average case incidence rates in New York, over a sample of 6 nowcast times. Each vertical dashed line indicates one of the nowcast times t, and corresponds to the case data pulled as of t, illustrated by the preceding colored line with markers. The solid black line indicates finalized 7-day trailing average case incidence rates used as ground truth, queried roughly three months after June 1, 2021.*

location in our setting, and solving this problem entails individual tuning for each geography. We consider using a simpler approach, at the cost of introducing a small amount of latency.

We define as our ground truth the reported 7-day trailing average case incidence rate $\bar{y}_t$, queried at a time much later than t when data revisions are exceedingly rare (In practice, we use the version of data issued on August 30, 2021, three months after our final nowcast date). To be clear, $\bar{y}_t$ averages over the daily case incidence rates $y_{t-6} \ldots, y_t$ also issued in the far future. Aside from addressing the issues just described, using the trailing average has the advantage of mitigating any remaining data artifacts that are permanently recorded (for example, the arrival of sudden batch of reports stemming from a change in case definition); such artifacts are in essence "averaged out". For simplicity, and for consistent notation, we will consider the averaged case observation $\bar{y}_s$ as a (day-of-week adjusted) realization of the true case rate y_s.

Figure 4.1 overlays both the original and averaged case signal for a sample of nowcast times in New York. Clearly, the real-time unaveraged signal exhibits cyclical day-of-week patterns, which is especially noticeable in the earlier nowcast times. We also notice that towards the end of March 2021 there is an anomalous jump in reported

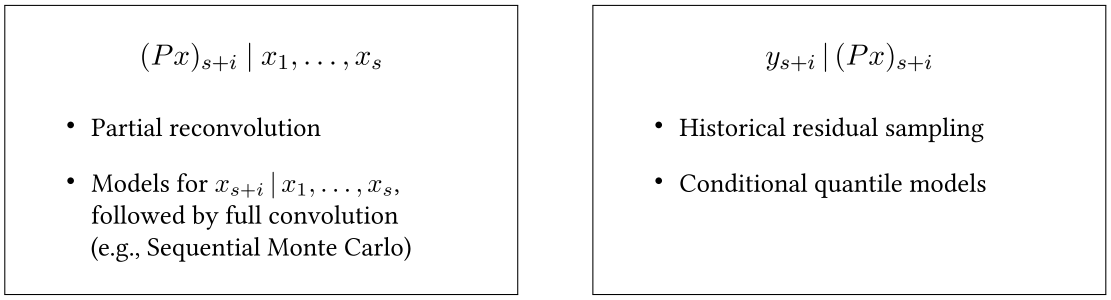

$(Px)_{s+i} \mid x_1, \dots, x_s$
Partial reconvolution
Models for $x_{s+i} \mid x_1, \dots, x_s$, followed by full convolution (e.g., Sequential Monte Carlo)
$y_{s+i} \mid (Px)_{s+i}$
Historical residual sampling
Conditional quantile models

Figure 4.2: *Overview of considered approaches for the reconvolution problem.*

cases; the effect of this spike is softened in the averaged signal.

4.2.4 Problem setup

Ultimately, we would like to estimate the probability of observing the true case rate y_{s+i} from a forecast density f constructed from the infection estimates $\hat{x}_1, \dots, \hat{x}_s$. To be clear about our various indices, we vary s across a range of recent lags $k = t - s$, and forecasting aheads $i = 1, 2, \dots, d$, where (given our first delay distribution assumption) $s + d$ is the final day that an infection onset at time s could be reported.

We organize our framework into three steps. At time t, using infections up until time s, and for an ahead i, we will:

1. generate draws of $x_1, \dots, x_s$ (which we henceforth refer to as the solution distribution), by a Monte Carlo procedure on the deconvolution method;
2. apply partial reconvolution on each draw to propagate the sampled infections to cases, denoted by the (point) trajectory $(Px)_{s+i}$;
3. add appropriate residual noise to each trajectory to simulate y_{s+i}, and estimate the conditional density $f(y_{s+i} \mid (Px)_{s+i})$.

To give a brief overview, step 1 introduces stochasticity into our infection estimates, which otherwise would be treated as fixed, by perturbing the case inputs and resolving the deconvolution problem. To be straightforward, this procedure helps us understand the *stability* of our deconvolution solution, but does not necessarily capture the *uncertainty* around our estimates. Nonetheless, following experiments show that introducing stochasticity here can be helpful in adding flexibility to resulting case forecasts, leading to a richer evaluation.

In step 2, we take the resulting infection draws and apply partial reconvolution to push infections forwards to cases. We remark that any point prediction method can be applied here, e.g., fitting an autoregressive model on an infection curve, iteratively forecasting forward, then performing a full convolution. We take an alternative avenue, and push forward the infection estimates by introducing additional rows to the convolution matrix P (corresponding to future time points), and renormalizing the delay probabilities in the appended rows such that the full probability mass is carried forward. We explain the details in Section 4.4.2.

Lastly, in step 3 we define a distribution at each forecast time by repeatedly adding test residual samples to the case forecasts. Assuming that the current nowcast task will have similar error as past nowcast tasks, we draw the samples from the empirical distribution of historical reconvolution residuals. In following experiments, the collection of residuals is stored over all available past, but future improvements could be made by various schemes to curate a bank of relevant residuals. One simple approach is to upweight the most recent residuals in time, which assumes that the most recent past better matches the short-term future; another more sophisticated approach could upweight on residuals where the historical case (or infection) curve exhibits a similar trend as the current case (infection) curve. We call the latter idea *curve alignment* (as opposed to the first approach of *time alignment*), and revisit this idea in Section 4.5.

In Figure 4.2, we provide an simple overview of considered approaches, which roughly groups together approaches for steps 1 and 2 on the left panel, and step 3 on the right. There are several possibilities to mix-and-match approaches between the panels; in what follows, we implement partial reconvolution followed by historical residual sampling, as just described. In the discussion section, we provide some details for the alternative ideas, which we leave as an open direction.

Before moving on to methodological details, we give a brief summary of our experimental setup for evaluation below.

Experimental setup

Using infection nowcasts made over October 1, 2020 to June 1, 2021, inclusive (a total period of 244 days), we run our framework, described shortly, over the evaluation period of January 1, 2021 to June 1, 2021, inclusive (a period of 151 days). We use the duration between October 1, 2020 to December 31, 2020 as a "burn-in" period to construct the empirical residual bank, used in Section 4.5. In all our experiments, we average the nowcast scores over all locations and every 5th nowcasting date in our evaluation period. (For the same reason that we evaluated our deconvolution methodology over every 10th nowcasting date, we do this to avoid overfitting to a particular methodology, and create a "pure" test set for our final run-through.) We will evaluate nowcasts produced at the state resolution, noting that our methods can be applied directly to the county (or any desired) resolution. In the interest of computational efficiency, we restrict ourselves to the 50 U.S. states.

Scoring metric

The forecast score is calculated by the estimated log probability

$$\log f(y_{s+i} \mid x_1, \dots x_s),$$

where y_{s+i} is the true (7-day averaged) case value, and f the estimated forecast density for time $s + i$. We will refer to this as the *log score* of the estimate, as is common in forecasting literature. Recalling our two dimensions of evaluation, we produce two types of analyses:

1. When evaluating along the lag $k = 1, \dots, 10$, we sum the log score across all forecasting aheads $i = 1, \dots, d$.
2. When evaluating as a function of the ahead i, we simply fix a single value of k, and produce independent results for several values of k.

The log score returns negative values for inputs smaller than 1; as we work with probabilities, the best score is 0, and larger values imply better nowcasting performance.

4.3 Related work

In predictive epidemiology, distributional forecasting and evaluation has become standard practice, even instituting common sets of metrics (e.g., binned log scores or ranked probability scores) when developing methodology [Bracher et al., 2021]. Generally, these metrics are applied in non-latent settings, such as influenza or COVID-19 case prediction, where the target signal is eventually observed (we note here that the literature here is vast, and point to several examples in the context of the aforementioned diseases: Brooks [2020], Brooks et al. [2018b], Cramer et al. [2021], Farrow [2016], McDonald et al. [2021]).

In large part, related epidemiological models that infer latent infections do so in tandem with the instantaneous reproductive number R_t, a key epidemic parameter, and typically perform model evaluation on estimates of R_t [e.g., Abbott et al. [2020], Abry et al. [2020], Bettencourt and Ribeiro [2008], Chitwood et al. [2021], Cori et al. [2013], Gostic et al. [2020], Pascal et al. [2021], Thompson et al. [2019]]. As R_t itself is unobserved, evaluations tend to rely on simulation studies where the exact data generating process is known. Compared to infections, when applied to real data R_t can be easier to visually align over plots of case incidence. In this form of evaluation the cooccurrence of trends is used to provide evidence towards model performance.

Work that performs deconvolution to directly infer infections, as we do, subsequently reconstruct the reproductive number, and draw comparisons to cases [e.g., Abry et al. [2020], Goldstein et al. [2009]]. In this approach, simulated data is also applied to better understand model performance. To the best of our knowledge, a distributional framework directly propagating infections to case reports has not been proposed, and our contributions that follow are novel.

4.4 Reconvolution to observables, Px

4.4.1 Monte Carlo deconvolution

We begin by introducing a simple Monte Carlo approach to produce samples of the infection estimates, where each sample can be thought of as a draw from the solution distribution of $\hat{x}_s$, $s < t$. To create stochasticity, we will repeatedly resolve perturbed versions of the deconvolution problem (3.10), where the perturbations are found through the residuals of our infection estimates. To be precise, at time t, we calculate and store the residuals

$$r_s = y_s - (\hat{P}\hat{x})_s, \tag{4.3}$$

$s \leq t$, where y_s are the latest issued case rate for time s, and $\hat{P}\hat{x}$ the full vector of reconvolved infections using the estimated convolution matrix $\hat{P}$ and infection nowcasts $\hat{x}$, all at the latest version issued as of t. Here, we leave the choice of the range of s, which determines how far back to calculate residuals, open to the modeler, as it is problem and time dependent and could be tuned in operationalization. (In practice, we take the starting point of s to be $t - 2d$, which corresponds to the length of the deconvolution window discussed in Section 3.6.3.) We store these residuals in an unordered and symmetric set, which we refer to as a residual "bank".

As our aim is construct samples around our original deconvolution estimate $\hat{x}_s$, we center our residual distribution to have mean zero through symmeterization. To this end, we conceptually consider these residuals as *unsigned*, and in practical implementation we simply attach a duplicate residual value with the opposite sign. Now, with our residual bank and initial deconvolution estimates $\hat{x}_s$, $s < t$ in hand, we outline the simulation of a single draw:

1. Perform full reconvolution on $\hat{x}_s$, $s < t$, to obtain $(\hat{P}\hat{x})_s$, $s \leq t$. Notice we are able to calculate $(P\hat{x})_t$ (as implied in (4.3)) as we have estimated all possible infections that would be reported at time t. This is a consequence of our third delay distribution assumption: no infection onset on day t will be reported on day t. This extra point is necessary to recover $\hat{x}_{t-1}$ later on in step 3.
2. For each available $s \leq t$, draw a residual $\tilde{r}$ at random from the residual bank, with replacement, and perturb the reconvolution estimate:

$$\tilde{y}_s = (P\hat{x})_s + \tilde{r}. \tag{4.4}$$

The sequence of $\tilde{y}_s$ acts as our perturbed case observation over time.

3. Re-solve the deconvolution problem (3.10) on $\tilde{y}_s$, $s \leq t$, to produce a simulated infection curve $\tilde{x}_s$, $s < t$. For computational purposes, we solve (3.10) using the best values of the regularization parameters λ, γ found by cross-validation in the initial deconvolution. We note that varying the amount of smoothness chosen by regularization can produce more diverse samples, and can be done in situations where the solving cost is cheap.

To generate more draws, we can simply repeat steps 2–3. as many times as desired using the same residual bank and output of step 1. This whole procedure is concisely outlined in Algorithm 4.1. Before we provide an illustration, we introduce one final tuning parameter in our method.

Scaling the residual bank

Aside from varying the parameters of the deconvolution method itself (say, over various regularization strengths), we can control the amount of diversity in our solution distribution by rescaling the residuals. We control this using a constant multiplier $\nu \geq 0$ on the residual values, where, at large values ν pushes the sampled estimates further away from the original solution. On the other hand, if $\nu = 0$ then our resulting solution distribution is a point mass at each $s < t$.

The effect of ν, along with the entire procedure notated in Algorithm 4.1, is illustrated in Figure 4.3. In the top panel, we plot quantile bands (at the 10% and 90% level) of the resulting $\tilde{y}$ samples; this corresponds to line 5 in Algorithm 4.1. We observe that at $\nu = 2$, the quantile bands just cover the overlaid case points, which were the original input used to produce the initial infection nowcasts. At $\nu < 2$, the resulting band is tighter than the original case observations, and at $\nu > 2$, the bands show a wider distribution. In the bottom panel, we plot the corresponding quantiles (again at 10% and 90% levels) of the re-solved infection solutions using the perturbed cases. At no added noise $\nu = 0$, the single line is the unperturbed infection solution.

Notice that the quantile bands around the perturbed case estimates are a constant width across time; this is by construction, as we sample uniformly from a symmetrical residual bank. However, the quantiles bands around the perturbed infection solutions are not uniform, where towards the right boundary we observe a gradual widening. As mentioned previously, it is important to note that this is *not* a reflection of the uncertainty around the right boundary, but rather the instability of the solution.

We discuss the choice of ν later on in our evaluation, as this decision can be informed by the historical forecasting error, for which we need two other components: a method to propagate the infection samples forward into the future, and a method to

Algorithm 4.1: Monte Carlo deconvolution

```
Input: Time $t$, deconvolution algorithm $A$, observed cases $y_s$, $s < t$, delay
       probabilities $\hat{p}_s = (\hat{p}_s(1), \dots \hat{p}_s(d))$, $s \le t$, residual noise multiplier
       $\nu > 0$, number of draws $B > 0$.
Output: Simulated draws of deconvolution solutions $\tilde{x}_s^{(b)}$, $s < t$, $b = 1, \dots, B$.
1 Solve $A$ using delay probabilties $\hat{p}$ and $y_s$ to obtain the initial solution $\hat{x}_s$, $s < t$.
2 Reconvolve infections to obtain $(\hat{P}\hat{x})_s = \sum_{j=1}^d \hat{p}_s(j)\hat{x}_{s-j}$, $s \le t$.
3 Compute symmetric residual bank as in (4.3).
4 for $b = 1, \dots, B$ do
5     Sample a residual $\tilde{r}$ from the residual bank, with replacement, and add to
        the reconvolution estimate:
              $\tilde{y}_s^{(b)} = (\hat{P}\hat{x})_s + \nu\tilde{r}$
        for $s \le t$.
6     Re-solve $A$ with $\hat{p}$ on $\tilde{y}_s^{(b)}$, $s \le t$ to obtain the perturbed solution $\tilde{x}_s^{(b)}$, $s < t$.
7 end
8 Return samples $\tilde{x}_s^{(b)}$, $s < t$, $b = 1, \dots, B$.
```

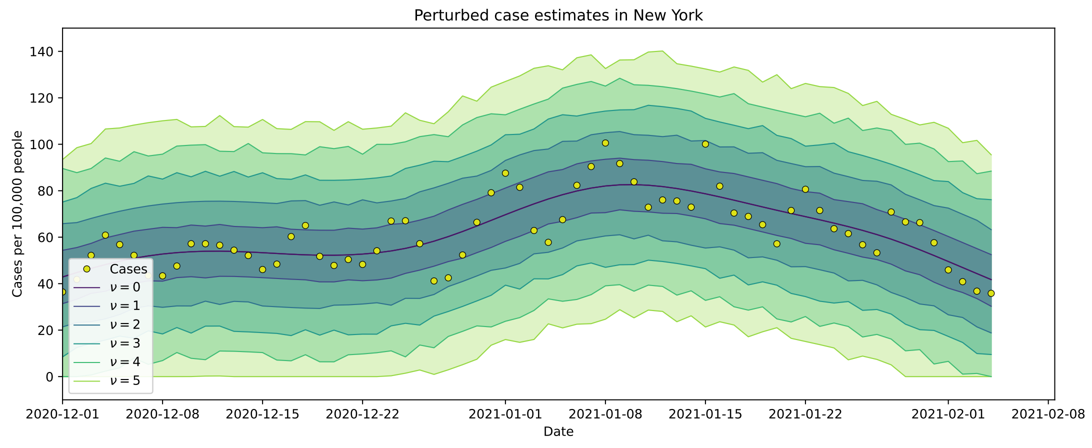
add uncertainty about the forecasted estimates. We now describe an avenue for the first component.

4.4.2 Partial reconvolution

In this section, we discuss *partial reconvolution*, our approach to push forward infection estimates into an estimate of future cases. To provide some motivation, consider the convolution model for a case report i days ahead of of a time $s < t$:

$$\mathbb{E}[y_{s+i} \,|\, x_1, \dots, x_s] = \sum_{j=1}^{d} p(j) x_{s+i-j}, \tag{4.5}$$

where $p(j)$ denotes the (assumed) time-invariant delay probabilities for the future report date. Here, for simplicity, we introduce one final assumption on the delay distribution: beginning at time t, the true delay probabilities $\pi_t(j)$ are stationary for the foreseeable future. Hence, in practice we set $p(j) = \hat{p}_t^{(t)}(j)$, $j = 1, \dots, d$ corresponding to the latest estimated delay distribution, for any report date after t. (In this work, we focus on providing an evaluation framework without introducing side models for the delay probabilities. While we do not believe the delay probabilities to be stationary over the future, we point out that this extra assumption is not a



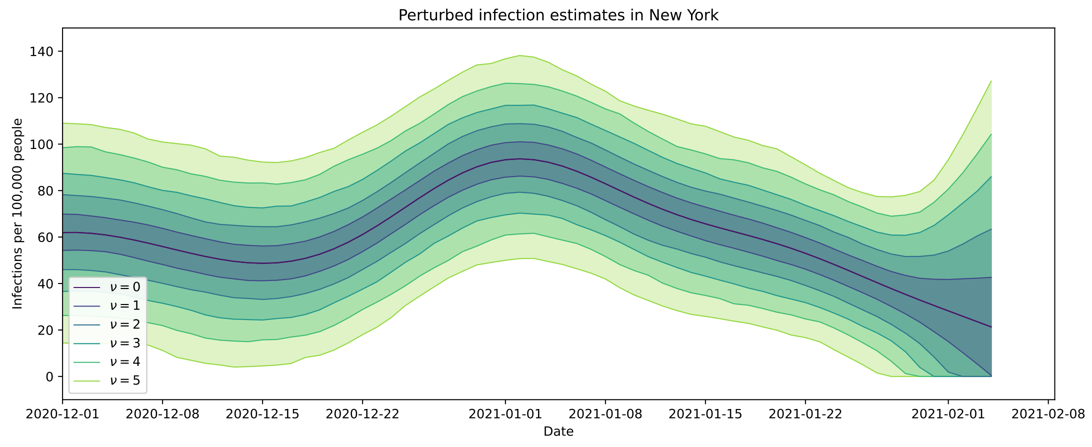


Figure 4.3: *Illustration of the effect of scaling the residual bank over values of the noise multiplier ν as in Algorithm 4.1. Top row: Perturbed case inputs to the deconvolution problem* (3.10)*, where the yellow points represent the initial case rates available in real-time. The colored bands capture the 10% and 90% level quantiles corresponding to varying values of ν. Bottom row: Quantiles at the 10% and 90% level of the corresponding perturbed estimates found by deconvolution on the perturbed case samples from the top panel. Both panels are generated using the latest data for New York as of February 5, 2021.*

weak point of our framework: a supplementary model can be proposed to evolve the delay distribution estimates forward, and these estimates can be simply plugged in the following methodology.)

Returning to our convolutional model (4.5), we note that, given infections up until time s, we are missing the estimates x_{s+i-j} for all $i > j$. To be clear, we are missing estimates of infections onset *after* time s that would have been reported at the future time $s + i$; this implies we are missing exactly $i - 1$ future estimates necessary to calculate $\hat{y}_{s+i}$.

Can we carry out reconvolution by summing over only the infection estimates already at hand? We take the following approach: consider compensating for the missing case mass (the product of a missing infection estimate and its corresponding delay probability $p(j)$) by renormalizing the remaining delay probabilities to sum to 1. Then, applying convolution using the adjusted delay weights and summing only over the observed infection estimates, we can estimate the future case rate as:

$$(P\hat{x})_{s+i} = \frac{\sum_{j=i}^{d} p(j)\hat{x}_{s+i-k}}{\sum_{j=i}^{d} p(j)}, \tag{4.6}$$

We refer to the solution of (4.6) as the partial reconvolution estimate. (As an implementational note, the convolution matrix P can be simply be appended with an additional row, corresponding to the renormalized delay probabilities for $s + i$.) From another perspective, partial reconvolution can be viewed as a variation on zero-padded convolution, where, after adding zeros to the boundary, we renormalize the convolution filter (our delay probabilities) at each step to ensure that the total mass is moved across. This additional step prevents underestimation for the extrapolated estimates.

4.5 Introducing uncertainty in $y \mid Px$

We now turn to the second half of our framework: incorporating uncertainty into our case forecast. As discussed previously, we will investigate and evaluate a residual based method, where the residual distribution comes from the errors of historical nowcasts.

As we saw in Section 4.4.2, we can apply partial reconvolution to produce a point forecast of estimated case rates, and by applying partial reconvolution to each simulated infection solution found by Algorithm 4.1, we have in hand a collection of forecasted case rate curves. Importantly, we point out that this collection of curves has introduced stochasticity into infections x_s, $s < t$ but we have not yet incorporated stochasticity into cases y_{s+i}, $i = 1, 2, \ldots, d$, which we refer to as the *forecast path*. In this section, we now introduce uncertainty into the forecast path, to produce a distribution for each $\hat{y}_{s+i}$, $i = 1, 2, \ldots, d$, via residual sampling.

At time t, for evaluating a nowcast at lag k, we calculate and store the residuals at all previous nowcast times $t' < t$, given by

$$r_s^{(t',t)} = \bar{y}_s^{(t)} - (Px^{(t')})_s, \tag{4.7}$$

$s \le t'$, where we reintroduce superscripts to denote the date the data or estimate was issued; we emphasize here that we always calculate the residual error to the most recent issue of case reports.

Given this residual "bank", and the point samples of reconvolved infections $(P\tilde{x})_s$, our final estimate is obtained by drawing a residual $\hat{r}$ at random from the residual bank, which increments the final reconvolution estimate

$$\hat{y}_{s+i} = (P\tilde{x})_{s+i} + \hat{r}. \tag{4.8}$$

We repeat this procedure over all infection samples (where sampling is done with replacement), and over all aheads $i = 1, \dots, d$. We provide several remarks on this approach below.

Remark 1. Differently to the residual calculation in (4.3), (4.7) calculates error to $\bar{y}_s^{(t)}$, the 7-day trailing average cases for time s as described in our discussion of ground truth. In Monte Carlo deconvolution, our goal was to generate a perturbed case sample similar to our *original* case input, $y_s^{(t)}$. Here, we wish to construct a residual distribution aligned with our ground truth signal, $\bar{y}_s^{(t)}$. One consequence of this change (as implicated by Figure 4.1) is that the distribution of these residuals will be narrower, and hence, produce tighter forecast bands (which are more meaningful for evaluation). Ideally, this residual distribution will have bias-correcting properties, which we remark on next.

Remark 2. We consider these residuals as *signed*, and hence we do *not* perform symmeterization on the residual distribution. Our reasoning follows along a similar vein as the previous remark: our goal is to form a density around our estimate of the truth, rather than around the infection solution as before. This distribution can have helpful bias, for example, if the residual bank was positively biased (meaning we underestimate the true case rate the majority of the historical past), then by repeated resampling, our forecasted distribution will be pushed upwards in a corrective manner. Of course, this approach assumes that the near future will behave similarly to the past, which is certainly not true near the peaks and dips of an epidemic.

Remark 3. To accompany remark 2, we note that we can "tune" the range of time points over which to calculate the residual bank. This can be parametrized by the range of past times t' we search over; if t' is close to the present time t, then we have assumed that the recent past matches the recent future. There are numerous variations of this, e.g., choosing t' such that the distance between recent case trends is small. This can be represented as $d(t', t) = D([y_{t'-l}, y_{t'}], [y_{t-l}, y_t])$, where we use brackets to denote a sequence of cases with length l, and D represents an appropriate distance metric (some immediate choices are Euclidean distance or cosine similarity). One can then assign sampling weights proportional to the distance, for instance, $w_{t'} = \exp(-\gamma d(t', t))$, and tune over various levels of $\gamma > 0$. Moreover, we note that this scheme can be applied *across geographies* to increase the size of the search history, which can lead to a highly customized residual distribution. Lastly, we point out that this weighting scheme can mitigate the previous concern in Remark 2 by identifying sequences of history where epidemic change-points have occurred.

Remark 4. Our final remark is on a second refinement to the residual bank construction. Notice that we add residuals to the partial reconvolution estimate, where, depending on the ahead i, has varying levels of uncertainty. Rather than sampling from historical "full reconvolution" residuals, we could sample from the set of partial reconvolution residuals calculated over past nowcast tasks (so we have some observations of the finalized ground truth) with matching i. Similar to remark 3, this can be enacted more generally as a weighting scheme, where for the forecast $s+i$, weights are given by $w_{t',i} = |\sum_{j=i}^{d} p_{s'+i}(j) - \sum_{j=i}^{d} p(j)|$, where $s' = t' - k$. This metric measures the (absolute) difference between the observed probability mass used to produce the current forecast at $s+i$, and the observed probability mass for the historical forecast at $s' + i$.

Ultimately, our goal is to obtain residuals from the historical nowcast tasks most similar to the current task. The basic sampling framework is formalized in Algorithm 4.2, where more sophisticated approaches (such as those detailed in remarks 3 and 4), can be swapped in at the first line.

4.6 Evaluation

We apply our reconvolution framework to evaluate three deconvolution methods provided in Chapter 3:

- trend filtering deconvolution (3.6);
- natural trend filtering deconvolution with a tapered penalty (3.10);
- sensor fusion deconvolution, which averages digital surveillance sensors as explained in Section 3.7.3, with the additional smoothing step given in Section 3.8.3.

We split our evaluation in two subsections, where the first subsection revisits the initial step of Monte Carlo deconvolution, and examines its utility by studying how performance across values of the residual multiplier ν. Here, we evaluate over the

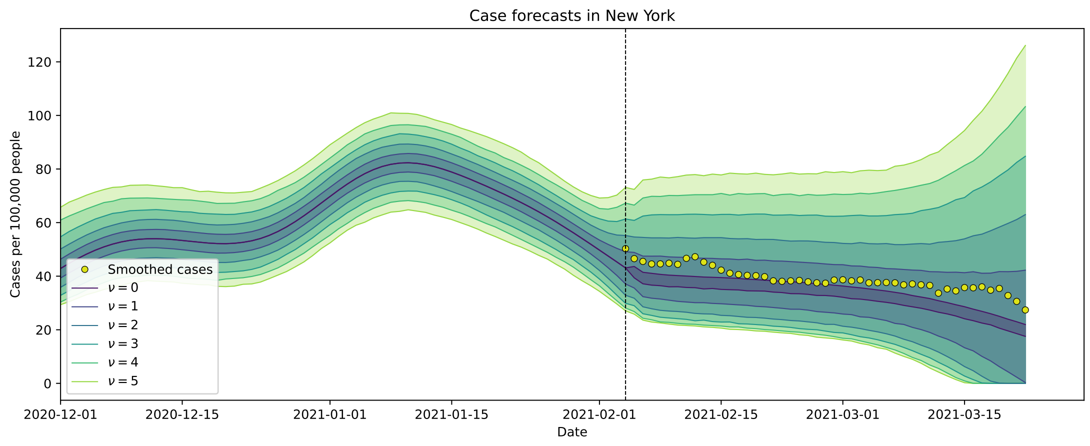


Figure 4.4: *Quantiles (level 10% and 90%) of the case forecast path across noise levels ν, using data as of February 5, 2021 (the nowcast date), at a $k = 1$ day lag. The black dashed line indicates the cut-off for observed data, which ends one day behind the nowcast time. The yellow points are the finalized 7-day trailing average case curve observed several months after the nowcast date. These forecasts are the simulated estimates from applying Algorithm 4.2 on the samples displayed in bottom panel of Figure 4.3.*

"training" evaluation period which runs on every 5th nowcasting date. This provides a cleaner "test" set for the following subsection, where we fix ν, and evaluate the deconvolution approaches using the larger test set, which runs over the full evaluation period, except for the aforementioned training dates.

4.6.1 Results of Monte Carlo deconvolution

To investigate the utility of introducing stochasticity at the level of infections, we evaluate the performance of the case forecasts made by deconvolution over six values of the scaling multiplier $\nu = 0, \ldots, 5$. Our baseline is at $\nu = 0$, where no stochasticity is introduced, and the infection estimate at any time has a point mass distribution at the original solution.

We first give the results for sensor fusion, which we consider as our "best" deconvolution method. Figure 4.5 plots the log score over the various levels of ν. The left panel plots the total summed log score as a function of the lag k (days back from the nowcast time t) summed over all forecast aheads $i = 1, \ldots, i$. The best performance (highest curve) is given by $\nu = 2$. This conclusion aligns with the panel on the right, which plots the average log score as a function of ν over all values of $(k,\ i)$. The worse performance is given at $\nu = 0$. These findings are intuitive; adding no noise essentially treats our infection solution as fixed and known, and is in this sense, overconfident.

Algorithm 4.2: Stochastic forecasts via residual sampling

```
Input: Time $t$, working onset time $s$, support size $d$, deconvolution samples
       $\tilde{x}_s^{(b)}$, $s < t$, $b = 1, \dots, B$, observed 7-day average cases $\bar{y}_s$, $s < t$, delay
       probabilities $\hat{p} = (\hat{p}_s(1), \dots, \hat{p}_s(d))$, $s \leq t$.
Output: Simulated draws of the forecasted $\hat{y}_{s+i}$, $i = 1, \dots, d$.
1 Compute residual bank as in (4.7), using $\bar{y}_s$ for $t' < t$.
2 for $b = 1, \dots, B$ do
3     Apply partial reconvolution as in (4.6) to $\tilde{x}_s^{(b)}$, $s < t$, to obtain $(P\tilde{x}^{(b)})_{s+i}$,
       $i = 1, \dots, d$.
4     Sample a residual $\tilde{r}$ from the residual bank, with replacement, and add to
       the partial reconvolution estimate:
           $\hat{y}_{s+i}^{(b)} = (P\tilde{x}^{(b)})_{s+1} + \tilde{r}$,
       for $i = 1, \dots, d$.
5 end
6 Return case forecast samples $\hat{y}_{s+i}^{(b)}$, $i = 1, \dots, d$, $b = 1, \dots, B$.
```

Adding a large amount of noise creates indecisive estimates, as the bands range over almost all feasible case values. This is visualized by Figure 4.4, where $\nu = 5$ (roughly) covers the interval $[0, 125)$.

Figure 4.6 displays the log score as a function of the forecast ahead i, for two fixed values of lag $k = \{1, 10\}$, corresponding to the 1-day-back nowcast and the 10-day-back nowcast. At short aheads $i < 4$, adding no noise $\nu = 0$ performs the best, but is quickly outstripped at larger aheads. We see that even at the furthest ahead $i = d$, using the largest noise level $\nu = 5$ does not outperform more moderate levels of noise. Both findings are not particularly surprising, as short term partial reconvolution is likely to perform well without introduced noise, but can quickly grow overconfident. On the other hand, using large levels of noise is unnecessary at for immediate forecasts, but is eventually helpful (up to a point).

We perform the same experiment on the other two considered deconvolution methods: trend filtering and natural trend filtering with tapered penalties. For the latter method, the result and interpretation is nearly identical to smoothed sensor fusion, with the best performance given at $\nu = 2$. However, for trend filtering we see that the best value is given at no added noise $\nu = 0$. This interesting result points to the noisiness and instability already present in the original trend filtering solution, as no level of added noise was helpful in improving forecasting performance. Analogous plots to Figures 4.5 and 4.6 for both these methods are provided in Appendix C.

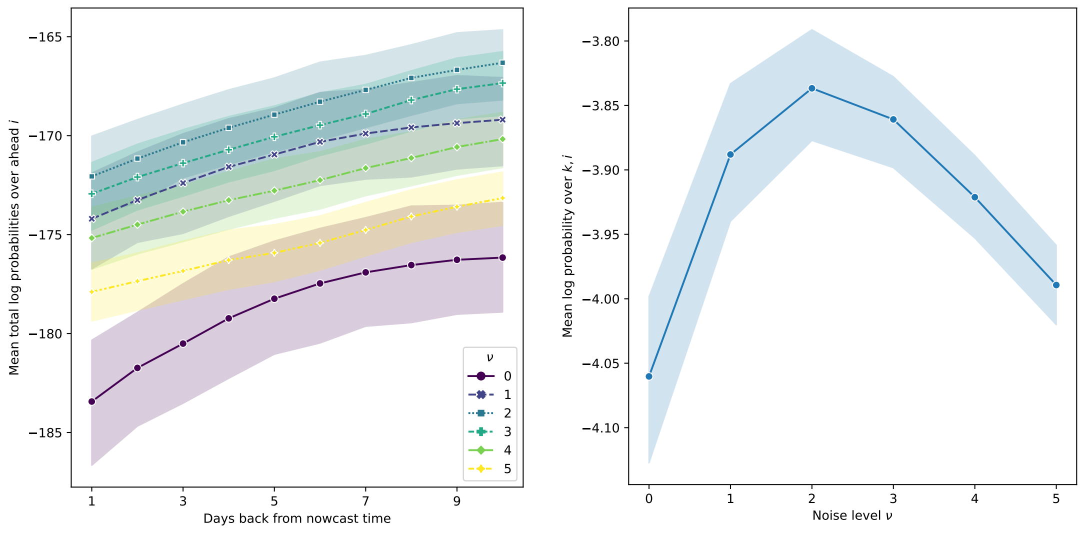


Figure 4.5: *Performance of smoothed sensor fusion over values of the residual multiplier ν. Left: Mean total log probability score summed over all aheads $i = 1, \ldots, d$, where the x-axis varies over values of the lag $k = t - s$. Larger values of lag k have better performance, as they have access to more complete information. Multiplying with noise level $\nu = 2$ has the best performance overall, matching with the results in Figure 4.5, while zero perturbation has the worst performance. Right: Mean log probability score as a function of the noise level ν. All positive values of ν improve on the pointwise approach $\nu = 0$, where no perturbation is applied. The best performance is found at $\nu = 2$. For both plots, and all subsequent figures in this chapter, the surrounding bands indicate 95% bootstrap confidence intervals.*

In general, the value of ν could (and should) be tuned over time, as historical nowcasts gradually observe the ground truth and can be evaluated. For the purposes of retrospective evaluation in the next subsection, we simply fix the value of $\nu = 2$ for the sensor fusion and natural trend filtering (tapered) methods, and $\nu = 0$ for the trend filtering method, which were the best performing for each method respectively.

4.6.2 Performance of deconvolution methods

We now evaluate the performance of the three deconvolution methods over every date in our evaluation period from January 1, 2021 to June 1, 2021, excluding every 5th nowcasting date which was used to fix ν.

Figure 4.7 shows the results as a function of the lag k, where we can immediately observe a huge improvement that natural trend filtering and sensor fusion make on the original trend filtering deconvolution method. This is a comforting result that

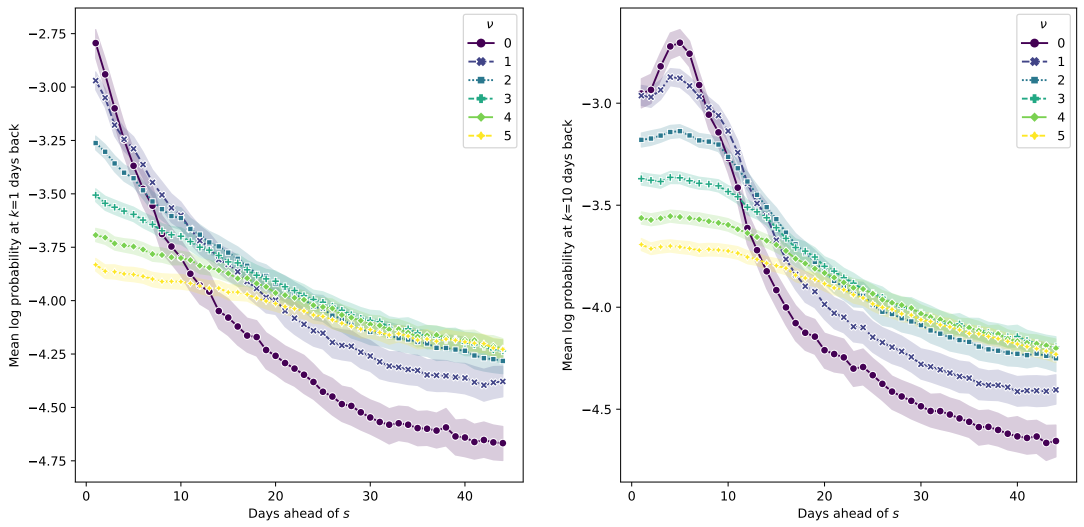


Figure 4.6: *Performance of smoothed sensor fusion over values of the residual noise multiplied ν. Left: Mean log probability score as a function of forecast aheads $i = 1, \ldots, d$, where we fix the lag $k = 1$, corresponding to the score for the most recent nowcast $\hat{x}_{t-1}$. Right: The same as the left, but at fixed lag $k = 10$. In both panels, applying zero perturbation with noise level $\nu = 0$ performs best for the immediate short-term forecasts. For longer aheads, approximately $i > 5$, introducing noise is beneficial.*

affirms many of our conclusions in the previous chapter. However, between the former two methods the difference is statistically negligible, and surprisingly, natural trend filtering appears to have slightly better scores. To a certain extent, the result may be explained by the post hoc smoothing applied on the sensor fusion estimates, which recall "joins" together the initial sensor fusion estimates at small values of k with the natural trend filtering estimates at large k. Hence, in large part the estimates across both methods are near identical, and the aggregate scoring metric (summed over all forecasting aheads $i = 1, \ldots, d$) shown here cannot capture any differences. Of course, the obvious exception is at $k = 1$, where sensor fusion is the only method that makes nowcasts with a 1-day lag; here, the difference in score between $k = 1$ and $k = 2$ appears to appropriately reflect the additional difficulty in producing a more recent nowcast.

We turn then to Figure 4.8, which compares the three methods over the forecast ahead i, at fixed lags $k = \{2, 10\}$. In both panels, we again observe the vast performance difference between trend filtering and the two improved methods throughout. Interestingly, we can observe that sensor fusion outperforms tapered natural trend filtering at small values of i. On the left panel, there is no overlap in the 95% bootstrapped confidence intervals, indicating that sensor fusion outperforms natural trend filtering up to roughly $i = 10$. For $i \in [11, 40]$, natural trend filtering produces better distributional forecasts over sensor fusion, which explains the result seen in the first Figure 4.7.

To better understand the performance as a function of i, we calculate the average difference in log score summed up over various "max ahead" i. To be clear, we calculate for each $i = 1, \dots, d$

$$\sum_{j=1}^{i} \log(\hat{p}^{\text{SF}}_{s+j}) - \log(\hat{p}^{\text{NTF}}_{s+j}),$$

where $\hat{p}_{s+j}$ is the estimated probability of observing the true case rate at $s + j$ from the forecast density created by each method. The log difference is averaged across all locations and evaluation times, at each lag k. The result is given in Figure 4.9, where each line corresponds to a value of i. We see that the mean log difference is positive until roughly $i = 20$, indicating that the more immediate forecasts from sensor fusion outperform natural trend filtering, whereas the reverse is true for long-horizon forecasts.

To be straightforward, estimates made after $k = 7$ are nearly identical for smoothed sensor fusion and natural trend filtering, yet we see clear performance differences at various levels of i. In particular, for very large i, partial reconvolution produces case forecasts that are nearly identical to the most recent infection estimates (in the most extreme case, the estimate for $\hat{y}_{s+d}$ is simply $\hat{x}_s$). This produces forecast densities that are not capturing the true case rate, and we consider investigation and improvements here as an immediate next step in future work.

4.7 Discussion

In this chapter, we proposed and implemented a reconvolution framework to evaluate three deconvolution methods described in Chapter 3. This distributional framework can be summarized by three components:

1. a Monte Carlo deconvolution procedure to produce draws of infection solutions;
2. a partial reconvolution technique to propagate sampled infection solutions forward to cases with minimal modeling assumptions;
3. a residual sampling approach to form distributional forecasts around each case estimate.

The results from applying our framework on COVID-19 infection nowcasts align well with our previous findings, and provides further evidence towards our methodological contributions developed in the deconvolution chapter. Alongside our proposed

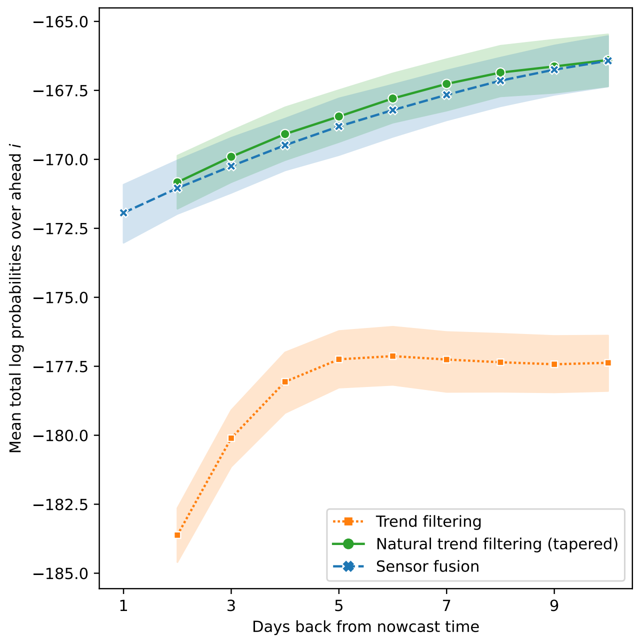


Figure 4.7: *Distributional evaluation of deconvolution estimates as a function of the lag* $k = t - s$.

methods, we described various extensions and open directions of our work, which can be explored to improve future evaluation. To finish, we revisit some of these open directions, and introduce others.

First, we note that our discussion of introducing stochasticity into our infection model (as in the first component) naturally lends itself to a discussion of sequential Monte Carlo and particle filter approaches [Doucet et al., 2001]. This class of nonlinear algorithms is popular and well-studied, and we can apply it to our setting to generate distributions across the entire infection trajectory. We further consider particle smoothers [Doucet and Johansen, 2009], which allow information to flow backwards from later observations to previous estimates, and can been seen as an analogue to fitting one deconvolution step across all observed history. The challenge here is in specifying good models for the transition and observation distributions, which we leave as an open direction.

Next, we point out that there are many avenues to propagate infections to cases beyond partial reconvolution (component 2). For a simple alternative, we could apply a time series model to predict infections in the future, and then apply reconvolution. A more complicated approach could be to construct an optimization problem similar in spirit to that which we proposed for deconvolution. Yet another approach could be

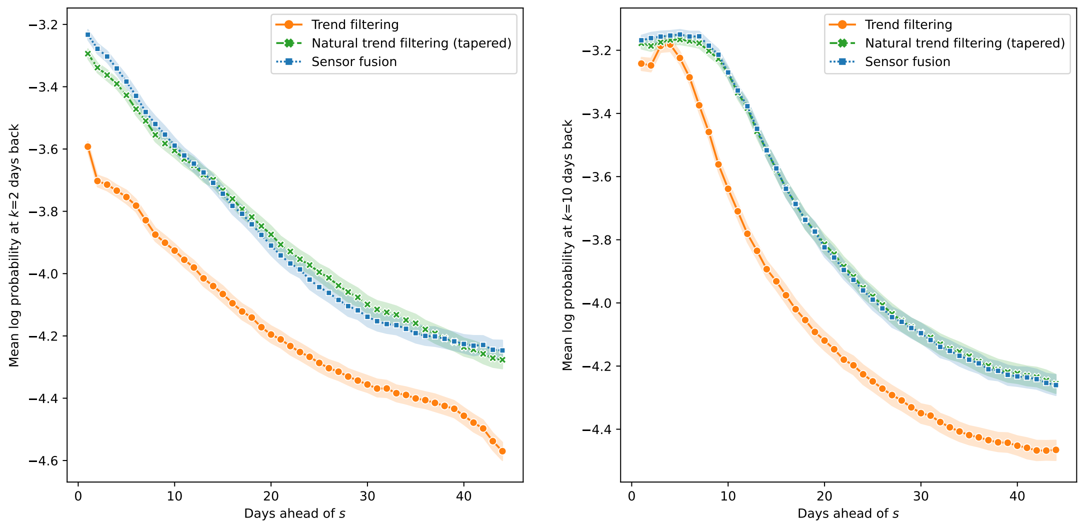


Figure 4.8: *Distributional evaluation of deconvolution estimates as a function of the forecast aheads* $i = 1, \ldots, d$. *The left panel gives the result for fixed lag* $k = 2$*, and the right panel for fixed lag* $k = 10$.

to implement Markov chain sampling that evolves infections forward. In any of these cases, such models must be implemented cautiously to avoid introducing any bias that could contaminate honest evaluation of the original nowcaster.

Finally, we remarked on various improvements to construct the residual bank used in the third component. We emphasize here that such improvements are the immediate next steps in improving the framework, as this component is critical to introducing proper measures of uncertainty to our estimates. Toward this end, we provided several concrete directions in Section 4.5.

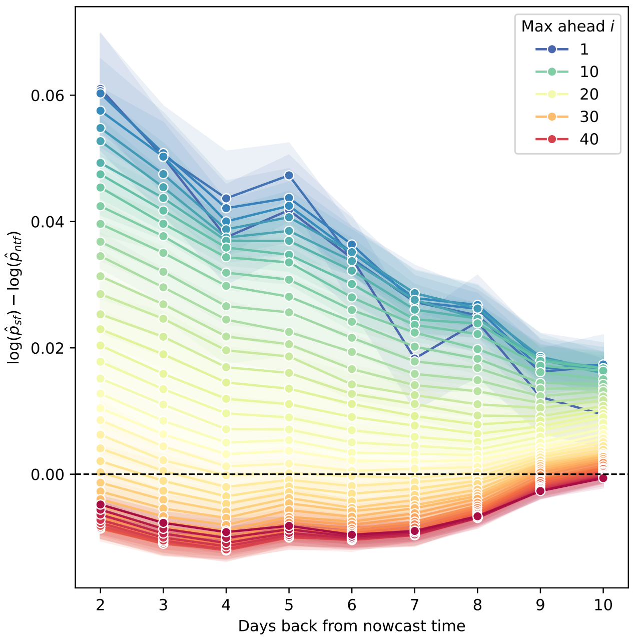


Figure 4.9: *Difference of log probability score between sensor fusion and tapered NTF, averaged over lags k and various aheads i.*

Chapter 5

Discussion

5.1 Summary and remarks

In this thesis, we described, implemented, and evaluated frameworks for nowcasting with sensor fusion. At each step, we demonstrated the utility of our contributions to produce timely predictions of disease spread, namely for tracking influenza and COVID-19. We group our contributions into two parts. In the first part, we built on an existing Kalman-filter-based sensor fusion framework in the influenza setting [Farrow, 2016], and described a mathematical equivalence of the estimator to the original Kalman filter and an unusual regression problem. This equivalence leads in a promising direction in adopting long-studied regression techniques (such as regularization) for dynamical system modeling (and possibly vice versa).

In the second part, we sought to estimate COVID-19 infections through deconvolving observed case reports. We described an optimization problem to perform deconvolution, and introduced various forms of regularization to mitigate the right truncation effect which affects the stability of the most recent nowcasts. In this work we also proposed an adjusted estimator (again to deal with real-time data limitations) to estimate the reporting delay distribution which convolves infections to cases. As our last methodological contribution, we proposed several ensembling approaches to perform sensor fusion, and showed that this additional layer contributes significantly towards the stability of our estimates.

We evaluated the utility of these algorithms in three ways: first, we made comparisons to finalized infection signal, which is not subject to any real-time estimation biases. Second, we performed a correlation analysis to demonstrate that the real-time infection estimates are an informative predictor of hospitalization rates, and can attain correlations comparable to case reporting signals, but with nearly a week's improvement in latency.

In our last evaluation step, we proposed a reconvolution framework that propagates infections forward into a case forecast, and in tandem, introduces a distributional layer. The resulting probabilistic forecasts were validated against finalized case reports, and we showed that the results are intuitive and promising. We gave algorithms for each step of the framework: Monte Carlo deconvolution to inject stochasticity into our infection solutions, partial reconvolution to move infections to case estimates, and residual sampling to stimulate case density forecasts. We finished by describing several open directions to improve the evaluation process.

We can also characterize the two parts of our thesis by the observedness of the target state. That is, in the first part we studied the case where we eventually get access to past state observations, whereas the latter part was motivated by the case where the state is always hidden, but we observe outcomes with an known convolutional model. Naturally, it follows that the next setting to consider is the fully hidden case.

In the context of Chapter 2, where we described the equivalences between Kalman filters, sensor fusion, and constrained regression, we discussed past state estimation through the Kalman filter (which traditionally operates in the fully hidden case), in order to use the regression formulation. We remark on another perspective and open direction. Recalling that the equivalence between sensor fusion and constrained regression relied on an empirical measurement noise covariance, we can draw from longstanding Kalman filtering techniques (e.g., Mehra [1970], Mohamed and Schwarz [1999]) to produce an estimate of the noise covariance, which can be subsequently used to derive past state estimates. It would be interesting to study what, if any, useful relationship arises when it replaces the observed state in the constrained regression. If an (even approximate) equivalence is found, it can be used to strengthen Theorem 2 and provide a comprehensive regression approach to the Kalman filter, for any situation, observed or latent.

Lastly, we finish by describe some concrete future directions for sensor fusion methodology.

5.2 Future directions

Quantile sensor fusion

The Kalman filter produces a distributional estimate through the state covariance $\hat{P}_t$. However, in settings where Gaussian assumptions may not hold, we can leverage the regression perspective (2.14), to flexibly estimate quantiles of the target state. Quantile sensor fusion solves the following minimization problem

$$\begin{aligned} \underset{b_j \in \mathbb{R}^d}{\text{minimize}} \quad & \sum_{i=1}^{t} \ell_\tau(x_{ij} - b_j^T z_i) \\ \text{subject to} \quad & H^T b_j = e_j, \end{aligned} \tag{5.1}$$

where ℓ_τ is the tilted pinball loss for given quantile level τ, $\ell_\tau(a) = a\,(\tau - \mathbf{1}\{a < 0\})$ and $\mathbf{1}$ is the indicator function. Several promising experiments (in the context of influenza nowcasting) of a median sensor fusion estimate to the original (mean) estimate shows that applying a quantile loss stabilizes estimates. Furthermore, these experiments show better coverage using the empirically constructed distribution when compared to its Gaussian counterpart. We lastly point out that these quantiles can be estimated jointly (with all accompanying extensions, e.g., non-crossing constraints).

Transferring regularization techniques

A natural extension of sensor fusion in its regression form (2.14) is to include a regularization term in the criterion:

$$\begin{aligned} \underset{b_j \in \mathbb{R}^d}{\text{minimize}} \quad & \frac{1}{t} \sum_{i=1}^{t} (x_{ij} - b_j^T z_i)^2 + \lambda_j P_j(b_j) \\ \text{subject to} \quad & H^T b_j = e_j, \end{aligned}$$

where $\lambda_j > 0$ is regularization parameter, and P_j is a penalty function. A ridge (squared ℓ_2) penalty, where $\lambda_j P_j(b_j) = ((1-\alpha)/\alpha) \sum_{\ell=1}^{d} b_{j\ell}^2$ has direct equivalence to the basic SF form (2.8), when the empirical noise covariance (2.13) is shrunken towards the identity. We provide the proof and details in Appendix A.4.

On the other hand, a lasso (ℓ_1) penalty, where $P_j(b_j) = \sum_{\ell=1}^{d} |b_{j\ell}|$ similarly improves sensor fusion, but has no obvious equivalence in the original SF form (2.8). This can be seen as a contribution from the regression approach, as it presents one solution to the problem of variable selection in SF (and hence, KF). Variable selection is a long-standing and well-studied topic in regression literature (and ℓ_1 regularization [Tibshirani, 1996], a popular solution), whereas measurement selection for the Kalman filter is, to the best of our knowledge, a relatively open problem. We give a synthetic example of performing process model selection (where we append various candidate process models to the measurement vector), detailed in Appendix A.4.

Joint learning and gradient boosting

Recall that sensorization independently transforms data sources $u_i \in \mathbb{R}^d, i = 1, ..., t$ at each time point. Joint training can be done by extending (2.14) as

$$\begin{aligned} \underset{\substack{b_j \in \mathbb{R}^d, j=1,\ldots,d \\ f \in \mathcal{F}}}{\text{minimize}} \quad & \frac{1}{t} \sum_{j=1}^{d} \sum_{i=1}^{t} \left(x_{ij} - b_j^T f(u_i) \right)^2 + \lambda P_j(f) \\ \text{subject to} \quad & H^T b_j = e_j, \quad j = 1, \ldots, k. \end{aligned} \tag{5.2}$$

where $\mathcal{F}_j$ is a space of functions from $\mathbb{R}^d$ to $\mathbb{R}^d$ (e.g., diagonal linear maps) and P_j is a regularization function to be specified by the modeler. The key in (5.2) is that we are simultaneously learning the sensors and fusing them. However, solving this minimization is difficult, as even in the linear map case the problem is nonconvex. We propose a more tractable alternative, using a training scheme inspired by gradient boosting [Friedman, 2001]. We call this approach *joint learning*, which proceeds iteratively, and cycles between state prediction and sensor fitting. This algorithm is formalized in Algorithm 5.1.

Relatedly, we propose a second gradient-boosting-inspired approach, which we differentiate by the name *sensor boosting*. Sensor boosting, detailed in Algorithm 5.2, takes a different tack. To motivate this approach, recall that sensors, as the outputs of predictive models trained for the same target, are by construction highly correlated. Here, we boost at the level of individual sensors by repeated selecting (say, cyclically) sensors and training the sensor model on the residual signal from the previous fusion step. In essence, this approach tries to decorrelate a sensor against the others, in such a manner that incorporates the leftover signal after sensor fusion. This approach assumes that the sensor models f_i are invertible, in the sense that we can recover $u_i \approx f_i^{-1}(x_i)$ (a de-sensorization of state back to data source). Experiments of this approach applied in the influenza setting have produced encouraging results.

Algorithm 5.1: Joint training of sensor fitting and fusion

```
Input: For each $j = 1, ..., d$, base learners $A_j$, data sources $u_{ij}$ over all history
       $i = 1, 2, \ldots, t$, measurement map $H$, number of boosting iterations $B$,
       small fixed learning rate $\eta > 0$.
Output: Predicted state $\hat{x}_{t+1}$.
Initialize $x_i^{(0)} = 0$, $i = 1, \ldots, t$.
for $b = 1, \ldots, B$ do
    for $j = 1, \ldots, d$ do
        Let $y_{ij}^{(b-1)} = (Hx^{(b-1)})_{ij}$, for $i = 1, \ldots, t-1$.
        Run $A_j$ with responses $\{y_{ij} - y_{ij}^{(b-1)}\}_{i=1}^{t-1}$ and covariates $\{u_{ij}\}_{i=1}^{t-1}$, to
          produce $\bar{f}_j^{(b)}$.
        Define intermediate sensors $z_{ij}^{(b)} = \bar{f}_j^{(b)}(u_{ij})$, for $i = 1, \ldots, t$.
    end
    for $j = 1, \ldots, k$ do
        Run SF as in (2.14) (possibly with regularization) with responses
          $\{x_{ij} - x_{ij}^{(b-1)}\}_{i=1}^{t-1}$ and covariates $\{z_i^{(b)}\}_{i=1}^{t-1}$, to produce $\hat{b}_j$.
        Define intermediate state fits $\bar{x}_{ij}^{(b)} = \hat{b}_j^T z_i^{(b)}$, for $i = 1, \ldots, t$.
        Update total state fits $x_{ij}^{(b)} = x_{ij}^{(b-1)} + \eta \bar{x}_{ij}^{(b)}$, for $i = 1, \ldots, t$.
    end
end
Return $\hat{x}_{t+1} = x_{t+1}^{(B)}$.
```

Algorithm 5.2: Sensor boosting

```
Input: For each $j = 1, ..., d$, base learners $A_j$, data sources $u_{ij}$ over all history
        $i = 1, 2, \dots, t$, measurement map $H$, small fixed learning rate $\eta > 0$.
Output: Predicted state $\hat{x}_{t+1}$.
Initialize $x_i^{(0)} = 0, i = 1, \dots, t$.
for $j = 1, \dots, d$ do
    Run $A_j$ with responses $\{y_{ij}\}_{i=1}^{t-1}$ and covariates $\{u_{ij}\}_{i=1}^{t-1}$ to produce $\bar{f}_j^{(0)}$.
    Define initial sensors $z_{ij}^{(0)} = \bar{f}_j^{(0)}(u_{ij})$ for $i = 1, \dots, t$.
end
Run SF as in (2.14) (possibly with regularization) with responses $\{x_{ij}\}_{i=1}^{t}$ and
  covariates $\{z_i^{(0)}\}_{i=1}^{t}$, to produce $\hat{b}_j$.
Define initial state fits $\bar{x}_{ij}^{(0)} = \hat{b}_j^T z_i^{(0)}$, for $i = 1, \dots, t$.
for $b = 1, \dots, B$ do
    Select the $\ell$th sensor. Selection can proceed cyclically, through random
      choice, or by a chosen criteria (for example,
      $\ell = \text{argmax}_\ell \sum_{i=1}^{t-1} (z_{i\ell}^{(b-1)} - x_{i\ell})^2$).
    Define the intermediate de-sensorized source $\bar{u}_{i\ell} = (\bar{f}_\ell^{(b-1)})^{-1}(x_{i\ell})$, for
      $i = 1, \dots, t$.
    Run $A_\ell$ with responses $\{y_{i\ell}\}_{i=1}^{t-1}$ with covariates $\{u_{i\ell} - \bar{u}_{i\ell}\}_{i=1}^{t-1}$ to produce
      $\bar{f}_\ell^{(b)}$.
    Update the sensor values $z_{i\ell}^{(b)} = z_{i\ell}^{(b-1)} + \eta \bar{f}_\ell^{(b)}(u_{i\ell} - \bar{u}_{i\ell})$ for $i = 1, \dots, t$.
    Run SF as in (2.14) (possibly with regularization) with responses $\{x_{ij}\}_{i=1}^{t}$
      and covariates $\{z_i^{(b)}\}_{i=1}^{t}$, to produce $\hat{b}_j$.
    Update the total state fits $x_{ij}^{(b)} = \hat{b}_j^T z_i^{(b)}$, for $i = 1, \dots, t$.
end
Return $\hat{x}_{t+1} = x_{t+1}^{(B)}$.
```

Appendix A

Proofs and Additional Details for KF-SF

A.1 Proof of Theorem 1

We can write the sensor fusion update as

$$\begin{aligned}\tilde{P}_{t+1} &= (\tilde{H}^T \tilde{R}_{t+1}^{-1} \tilde{H})^{-1}\\ \hat{x}_{t+1} &= \tilde{P}_{t+1} \tilde{H}^T \tilde{R}_{t+1}^{-1} \tilde{z}_{t+1},\end{aligned}$$

where

$$\tilde{P}_{t+1} = (H^T R^{-1} H + \bar{P}_{t+1}^{-1})^{-1}.$$

By the Woodbury matrix identity, $(A+UCV^{-1}) = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$, with $A = \bar{P}_{t+1}^{-1}$ in our case, we get

$$\begin{aligned}\tilde{P}_{t+1} &= \bar{P}_{t+1} - \bar{P}_{t+1} H^T (R + H\bar{P}_{t+1}H^T)^{-1} H \bar{P}_{t+1}\\ &= (I - \bar{P}_{t+1} H^T (R + H\bar{P}_{t+1}H^T)^{-1} H)\bar{P}_{t+1}\\ &= (I - K_{t+1}H)\bar{P}_{t+1}, \qquad \text{(A.1)}\end{aligned}$$

where recall, the Kalman gain K_{t+1} is defined in (2.6).

Now let us we rewrite the Kalman gain as

$$\begin{aligned}K_{t+1} &= \bar{P}_{t+1} H^T (R + H\bar{P}_{t+1}H^T)^{-1}\\ &= \bar{P}_{t+1} H^T R^{-1} (I + H\bar{P}_{t+1}H^T R^{-1})^{-1},\end{aligned}$$

so that

$$K_{t+1}(I + H\bar{P}_{t+1}H^T R^{-1}) = \bar{P}_{t+1} H^T R^{-1},$$

and after rearranging,

$$K_{t+1} = (I - K_{t+1}H)\bar{P}_{t+1}H^T R^{-1}. \tag{A.2}$$

Putting (A.1) and (A.2) together, we get

$$\begin{aligned}
\tilde{P}_{t+1}\tilde{H}^T\tilde{R}_{t+1}^{-1}\tilde{z}_{t+1} &= (I - K_{t+1}H)\bar{P}_{t+1}(H^T R^{-1}z_{t+1} + \bar{P}_{t+1}^{-1}\bar{x}_{t+1}) \\
&= (I - K_{t+1}H)\bar{P}_{t+1}H^T R^{-1}z_{t+1} + (I - K_{t+1}H)\bar{x}_{t+1} \\
&= K_{t+1}z_{t+1} + (I - K_{t+1}H)\bar{x}_{t+1} \\
&= \bar{x}_{t+1} + K_{t+1}(z_{t+1} - H\bar{x}_{t+1}),
\end{aligned}$$

which is exactly the Kalman filter prediction, completing the proof.

A.2 Derivation of (2.12)

We first make the EKF estimate precise. Let

$$F_{t+1} = Df(\hat{x}_t), \tag{A.3}$$
$$H_{t+1} = Dh(\bar{x}_{t+1}), \tag{A.4}$$

and define

$$\bar{x}_{t+1} = F_{t+1}\hat{x}_t, \tag{A.5}$$
$$\hat{x}_{t+1} = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big), \tag{A.6}$$

where $K_{t+1} \in \mathbb{R}^{k\times d}$ is defined via

$$\bar{P}_{t+1} = F_{t+1}P_t F_{t+1}^T + Q, \tag{A.7}$$
$$K_{t+1} = \bar{P}_{t+1}H_{t+1}^T(H_{t+1}\bar{P}_{t+1}H_{t+1}^T + R)^{-1}, \tag{A.8}$$
$$P_{t+1} = (I - K_{t+1}H_{t+1})\bar{P}_{t+1}, \tag{A.9}$$

Note that (A.7)–(A.9) are exactly the same as (2.5)–(2.7), with F_{t+1}, H_{t+1} replacing F, H, respectively. Moreover, (A.5), (A.6) are *nearly* the same as (2.3), (2.4), with again F_{t+1}, H_{t+1} replacing F, H, except that the residual in (A.6) is $z_{t+1} - h(\bar{x}_{t+1})$, and not $z_{t+1} - H_{t+1}\bar{x}_{t+1}$, as would be analogous from (2.4).

Next, we make what we called the extended SF (ESF) estimate precise. Let $\tilde{z}_{t+1} \in \mathbb{R}^{d+k}$ be as in (2.11), let $\tilde{H}_{t+1} \in \mathbb{R}^{(d+k)\times k}$ be the rowwise concatentation of H_{t+1} and $I \in \mathbb{R}^{k\times k}$, and $\tilde{R}_{t+1}$ be as in (2.9). Here, $F_{t+1}, H_{t+1}, \bar{P}_{t+1}$ are as defined in (A.3), (A.4), (A.7), respectively. The ESF estimate is

$$\hat{x}_{t+1} = (\tilde{H}^T\tilde{R}_{t+1}^{-1}\tilde{H})^{-1}\tilde{H}^T\tilde{R}_{t+1}^{-1}\tilde{z}_{t+1}. \tag{A.10}$$

To see that (A.10) and (A.6) are equal, note that by following the proof of Theorem 1 directly, with F_{t+1}, H_{t+1} in place of F, H, we get

$$(\tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{H}_{t+1})^{-1} \tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{z}_{t+1} = \bar{x}_{t+1} + K_{t+1}(z_{t+1} - H_{t+1}\bar{x}_{t+1}).$$

Adding and subtracting $K_{t+1}h(\bar{x}_{t+1})$ to the right-hand side gives

$$\begin{aligned}
&(\tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{H}_{t+1})^{-1} \tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} (z_{t+1}, \bar{x}_{t+1}) \\
&\quad = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big) + K_{t+1}(h(\bar{x}_{t+1} - H_{t+1}\bar{x}_{t+1}) \\
&\quad = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big) + (I - K_{t+1}H_{t+1})\bar{P}_{t+1}H_{t+1}^T R^{-1}(h(\bar{x}_{t+1} - H_{t+1}\bar{x}_{t+1}) \\
&\quad = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big) + \tilde{P}_{t+1}H_{t+1}^T R^{-1}(h(\bar{x}_{t+1} - H_{t+1}\bar{x}_{t+1}),
\end{aligned}$$

where in the second line we used (A.2), and in the third we used (A.1). Rearranging gives

$$(\tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \tilde{H}_{t+1})^{-1} \tilde{H}_{t+1}^T \tilde{R}_{t+1}^{-1} \big(z_{t+1} + H_{t+1}\bar{x}_{t+1} - h(\bar{x}_{t+1}),\, \bar{x}_{t+1}\big) = \bar{x}_{t+1} + K_{t+1}\big(z_{t+1} - h(\bar{x}_{t+1})\big),$$

which is precisely the desired conclusion, in (2.12).

A.3 Proof of Theorem 2

Let us denote $X \in \mathbb{R}^{t\times k}$ and $Z \in \mathbb{R}^{t\times d}$ the matrices of states and sensors, respectively, for the first t time points. That is, X has rows $x_i \in \mathbb{R}^k$, $i = 1, \dots, t$ and Z has rows $z_i \in \mathbb{R}^d$, $i = 1, \dots, t$. Fix any $j = 1, \dots, k$. Let $\hat{a}_j \in \mathbb{R}^d$ be the jth column of $\hat{R}_{t+1}^{-1} H (H^T \hat{R}_{t+1}^{-1} H)^{-1}$, and let $\hat{b}_j \in \mathbb{R}^d$ be the solution of (2.14), equivalently, the solution of

$$\begin{aligned}
\underset{b_j \in \mathbb{R}^d}{\text{minimize}} \quad & \|X_j - Z b_j\|_2^2 \\
\text{subject to} \quad & H^T b_j = e_j,
\end{aligned} \tag{A.11}$$

where X_j denotes the jth column of X. We will show that $\hat{a}_j = \hat{b}_j$.

The Lagrangian of problem (A.11) is

$$L(b_j, u_j) = \|X_j - Z b_j\|_2^2 + u_j^T (H^T b_j - e_j),$$

for a dual variable (Lagrange multiplier) $u_j \in \mathbb{R}^k$. Taking the gradient of the Lagrangian and setting it equal to zero at an optimal pair $(\hat{b}_j, \hat{u}_j)$ gives

$$0 = Z^T (Z\hat{b}_j - X_j) + H\hat{u}_j,$$

and rearranging gives

$$\hat{b}_j = (Z^T Z)^{-1} (Z^T X_j - H\hat{u}_j). \tag{A.12}$$

The dual solution $\hat{u}_j$ can be determined by plugging (A.12) into the equality constraint $H^T\hat{b}_j = e_j$, but for our purposes, the explicit dual solution is unimportant.

We will now show that $\hat{b}_j = \hat{R}_{t+1}^{-1}H\hat{\beta}_j$ for some $\hat{\beta}_j \in \mathbb{R}^k$. Write

$$\begin{aligned}\hat{R}_{t+1} &= \frac{1}{t}(Z - XH^T)^T(Z - XH^T) + (1-\alpha)I \\ &= \frac{1}{t}(Z^TZ - HX^TZ - Z^TXH^T + HX^TXH^T).\end{aligned}$$

Then

$$\begin{aligned}\hat{R}_{t+1}\hat{b}_j &= \frac{1}{t}(Z^TZ\hat{b}_j - HX^TZ\hat{b}_j - Z^TXH^T\hat{b}_j + HX^TXH^T\hat{b}_j) \\ &= \frac{1}{t}(Z^TX_j - H\hat{u}_j - HX^TZ\hat{b}_j - Z^TX_j + HX^TX_j) \\ &= H\underbrace{\left(\frac{X^TX_j - \hat{u}_j - X^TZ\hat{b}_j}{t}\right)}_{\hat{\beta}_j},\end{aligned}$$

as desired, where in the second line we have used (A.12) and the constraint $H^T\hat{b}_j = e_j$.

Observe that $\hat{a}_j = \hat{R}_{t+1}^{-1}H\hat{\alpha}_j$ for some $\hat{\alpha}_j \in \mathbb{R}^k$, in particular, for $\hat{\alpha}_j$ defined to be the jth column of $(H^T\hat{R}_{t+1}^{-1}H)^{-1}$. Further,

$$e_j = H^T\hat{a}_j = H^T\hat{b}_j$$

the constraint on $\hat{a}_j$ holding by direct verification, and the constraint on $\hat{b}_j$ holding by construction in (A.11). That is,

$$H^T\hat{R}_{t+1}^{-1}H\hat{\alpha}_j = H^T\hat{R}_{t+1}^{-1}H\hat{\beta}_j,$$

and since $H^T\hat{R}_{t+1}^{-1}H$ is invertible, this leads to $\hat{\alpha}_j = \hat{\beta}_j$, and finally $\hat{a}_j = \hat{b}_j$, completing the proof.

A.4 Further SF-regression equivalences

A.4.1 More regularization: covariance shrinkage

Covariance shrinkage—which broadly refers to the technique of adding a well-conditioned matrix to a covariance estimate to provide stability and regularity—is widely used and well-studied in modern multivariate statistics, data mining, and machine learning. As such, it would be natural to replace the empirical covariance matrix estimate (2.13) for the measurement noise covariance by

$$\hat{R}_{t+1} = \frac{\alpha}{t}\sum_{i=1}^{t}(z_i - Hx_i)(z_i - Hx_i)^T + (1-\alpha)I, \tag{A.13}$$

for a parameter $\alpha \in [0, 1]$. For sensor fusion in the flu nowcasting problem, this is considered (in some form) in Farrow [2016], and leads to significant improvements in nowcasting accuracy.

Our next result shows that when we use shrinkage as in (A.13) to estimate the measurement noise covariance in SF, this is equivalent to adding a ridge penalty in the regression formulation.

Corollary 1. *Let $\hat{R}_{t+1}$ be as in* (A.13), *for some value $\alpha \in [0, 1]$. Consider the SF prediction at time $t + 1$, with $\hat{R}_{t+1}$ in place of R, denoted $\hat{x}_{t+1} = \hat{B}^T z_{t+1}$. Then each column of $\hat{B}$, denoted $\hat{b}_j \in \mathbb{R}^d$, $j = 1, \dots, k$, solves*

$$\begin{aligned} \underset{b_j \in \mathbb{R}^d}{\text{minimize}} \quad & \frac{1}{t}\sum_{i=1}^{t}(x_{ij} - b_j^T z_i)^2 + \frac{(1-\alpha)}{\alpha}\|b_j\|_2^2 \\ \text{subject to} \quad & H^T b_j = e_j. \end{aligned}$$

Proof. As before, let $X \in \mathbb{R}^{t \times k}$ and $Z \in \mathbb{R}^{t \times d}$ denote the matrix of states and sensors, respectively, over the first t time points. We can write $\hat{R}_{t+1}$ in (A.13)

$$\frac{\alpha}{t}(Z - XH^T)^T(Z - XH^T) + (1-\alpha)I = \frac{1}{t}(\tilde{Z} - \tilde{X}H^T)^T(\tilde{Z} - \tilde{X}H^T),$$

where $\tilde{Z} \in \mathbb{R}^{(t+d)\times d}$ is the rowwise concatenation of $\sqrt{\alpha/t}Z$ and $\sqrt{1-\alpha/t}I$, and $\tilde{X} \in \mathbb{R}^{(t+k)\times k}$ is the rowwise concatenation of $\sqrt{\alpha/t}X$ and $0 \in \mathbb{R}^{k\times k}$ (the matrix of all 0s). Applying Theorem 2 to $\tilde{X}, \tilde{Z}$, expanding the criterion in the regression problem, and then multiplying the criterion by $1/\alpha$, gives the result. □

A.4.2 Less regularization: zero padding

In the opposite direction, we now show that we can modify SF and obtain an equivalent regression formulation with less regularization, specifically, without constraints.

Corollary 2. *At each $t = 1, 2, 3, \dots$, suppose we augment our measurement vector by introducing k measurements that are identically zero, denoted $\tilde{z}_t = (z_t, 0) \in \mathbb{R}^{d+k}$. Suppose that we augment our measurement map accordingly, defining $\tilde{H} \in \mathbb{R}^{(d+k)\times k}$ to be the rowwise concatention of H and the identity $I \in \mathbb{R}^{k\times k}$. Consider running SF on this augmented system, using the empirical covariance to estimate R, and let $\hat{x}_{t+1} = \hat{B}^T z_{t+1}$ denote the SF prediction at time $t + 1$. Then each column of $\hat{B}$, denoted $\hat{b}_j \in \mathbb{R}^d$, $j = 1, \dots, k$, solves*

$$\underset{b_j \in \mathbb{R}^d}{\text{minimize}} \; \sum_{i=1}^{t}(x_{ij} - b_j^T z_i)^2.$$

Proof. Applying Theorem 2 to the augmented system gives the equivalent regression problem

$$\begin{aligned} \underset{b_j \in \mathbb{R}^d,\, a_j \in \mathbb{R}^k}{\text{minimize}} \quad & \sum_{i=1}^{t} (x_{ij} - b_j^T z_i - a_j^T 0)^2 \\ \text{subject to} \quad & H^T b_j + a_j = e_j. \end{aligned}$$

The constraint is satisfied with $a_j = e_j - H^T b_j$. But a_j has no effect on the criterion, so the constraint can be removed. □

Remark 5. The analogous equivalence holds for covariance shrinkage and ridge regression. That is, in Corollary 2, if instead of the empirical covariance, we use α times the empirical covariance plus $(1-\alpha)I$, then SF on the augmented system is equivalent to unconstrained ridge, at tuning parameter $(1-\alpha)/\alpha$.

A.5 Example of process model selection

Here we give a simple empirical example of process model selection using the regression formulation of SF. We initialized $x_0 = 1$, and generated data according to

$$\begin{aligned} x_t &= 0.5 x_{t-1} + 0.05 \sin(0.126 t) + \delta_t, \\ z_t &= H x_t + \epsilon_t, \end{aligned}$$

for $t = 1, \dots, 200$. Here $H \in \mathbb{R}^{4\times 1}$ is simply the column vector of all 1s, and the noise is drawn as $\delta_t \sim N(0, 0.01)$, $\epsilon_t \sim N(0, I)$, independently, over $t = 1, \dots, 150$.

The prediction setup is as follows. At each time $t+1$, when making a prediction of x_{t+1}, we observe all past states x_i, $i = 1, \dots, t$ and all measurements z_i, $i = 1, \dots, t+1$. We fit 5 different candidate process models to past state data:

1. linear autoregression;
2. quadratic autoregression;
3. spline regression on time;
4. sine regression on time;
5. cosine regression on time.

To be clear, models 1 and 2 regress x_i on x_{i-1} and x_{i-1}^2, respectively, over $i = 1, \dots, t$. Models 3–5 regress x_i on a spline, sine, and cosine transformation of i, respectively, over $i = 1, \dots, t$. The sine and cosine transformations are given the true frequency. The spline is a cubic smoothing spline (with a knot at every data point) and its tuning parameter is chosen by cross-validation (using only the past data). After being fit, we use each of the candidate process models 1–5 to make a prediction of x_{t+1}, given z_{t+1}. We take this as its ouput.

For $t = 151, \ldots, 200$, we define $\tilde{z}_t \in \mathbb{R}^9$ to be the measurement vector $z_t \in \mathbb{R}^4$ augmented with the outputs of the 5 candidate process models as described above (the burn-in period of 150 time points ensures that the candidate process models have enough training data to make reasonable predictions). Figure A.1 shows the outputs from these models over the last 50 time points.

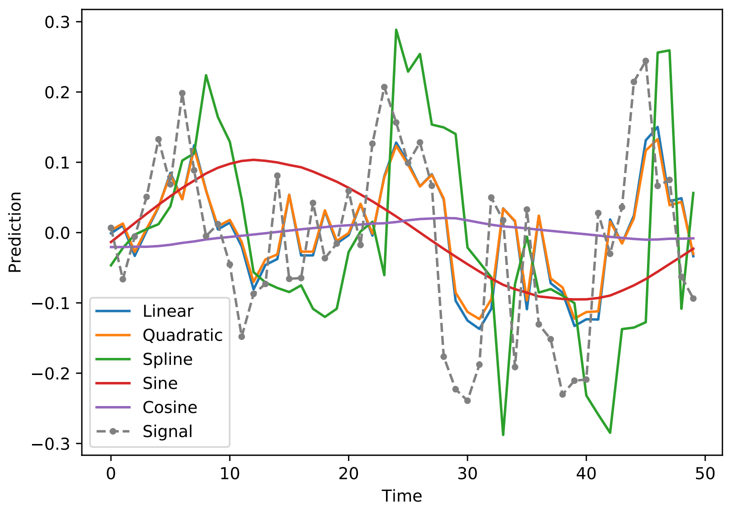


Figure A.1: *Simple process model selection example: outputs from 5 candidate process models, over the last 50 time points.*

Finally, in the last 50 time points, to get an assimilated prediction of $\hat{x}_{t+1}$ at each time $t + 1$, we solve the constrained regression problem with a lasso penalty, using cross-validation to select λ (again, using only past data). Further, we penalize only the coefficients of the candidate process models (not the pure measurements). Table A.1 shows the median of the coefficients over the last 50 time points (in this table, the coefficients for the pure measurement sensors are aggregated as one). We see that the lasso tends to select the linear and sine sensors, as expected (because these two make up the true dynamical model), and places a small weight on the spline sensor (which is flexible, and can mimic the contribution of the sine sensor).

	Linear	Quadratic	Spline	Sine	Cosine	Measurements
Median Coefficient	0.643	0.000	0.094	0.189	0.000	0.0175

Table A.1: *Simple process model selection example: median regression coefficients for the sensors, over the last 50 time points.*

Appendix B

Supplementary Deconvolution Material

B.1 ADMM for solving deconvolution problems

Here we give details on the ADMM approach used to solve the regularized least squares deconvolution problems in Sections 3.5 and 3.6. We first focus on problem (3.5), and then we discuss the modifications needed when incorporating extra regularization for real-time deconvolution as in (3.10). To simplify notation, we will henceforth drop the subscript dependnece of all quantities on the location ℓ, as well as the superscript dependence on the nowcast date t for the real-time problems.

We also use $\hat{P}$ to denote the (Toeplitz) convolution matrix with rows determined by $\hat{p}_s$, $s < t$, i.e., such that for any vector x (of appropriate dimension)

$$(\hat{P}x)_s = \sum_{k=1}^{d} \hat{p}_k x_{s-k}.$$

(We leave the dimensions of $\hat{P}$ and x here purposely ambiguous, which should always be clear from the context anyway; this allows us to borrow similar notation across problems with different underlying dimensions.) Thus we can rewrite (3.5) as

$$\underset{x}{\text{minimize}} \;\; \|y - \hat{P}x\|_2^2 + \lambda \|D^{(4)}x\|_1.$$

To apply ADMM, we must introduce auxiliary variables, and as in Ramdas and Tibshirani [2016], we use the following "specialized" decomposition (which improves the convergence speed):

$$\begin{aligned}&\underset{x}{\text{minimize}}\ \|y - \hat{P}x\|_2^2 + \lambda\|D^{(1)}\alpha\|_1\\&\text{subject to}\ \ \alpha = D^{(3)}x,\end{aligned}$$

where we used the recursive nature of the difference operators, writing the 4th-order operator as a product of the 1subject to- and 3rd-order operators: $D^{(4)} = D^{(1)}D^{(3)}$. The above problem gives rise to the augmented Lagrangian:

$$\mathcal{L}(x, \alpha, u) = \|y - \hat{P}x\|_2^2 + \lambda\|D^{(1)}\alpha\|_1 + \rho\|\alpha - D^{(3)}x + u\|_2^2 - \rho\|u\|_2^2,$$

which corresponds to following ADMM updates, writing $D = D^{(3)}$ for brevity:

$$\begin{aligned}x &\leftarrow (\hat{P}^T\hat{P} + \rho D^T D)^{-1}\big(\hat{P}^T y + \rho D^T(\alpha + u)\big)\\\alpha &\leftarrow \underset{z}{\text{argmin}}\ \|Dx - u - z\|_2^2 + \frac{\lambda}{\rho}\|D^{(1)}\alpha\|_1\\u &\leftarrow u + \alpha - Dx.\end{aligned}$$

The α-update here requires solving a 1-dimensional fused lasso problem, which can be done in linear-time with the dynamic programming approach of Johnson [2013]. The x-update is more expensive than in pure trend filtering (with no convolution operator) but owing to the bandedness of $\hat{P}$ (and D, though the bandwidth d of $\hat{P}$ dominates), it can still be solved in $O(nd)$ operations. Further, in this and all applications of ADMM, we follow the recommendation of Ramdas and Tibshirani [2016] and set the Lagrangian parameter equal to the tuning parameter, $\rho = \lambda$.

As for the two extensions presented in (3.10), the natural trend filtering constraints can be be enforced by introducing a linear interpolant matrix as described in Section 11.2 of Tibshirani [2020]. This effectively replaces the convolution matrix $\hat{P}$ and the 3rd difference operator D, in the ADMM steps above, by $\tilde{P}$ and $\tilde{D}$, respectively, which are given by right multiplying P and D by the interpolant matrix.

Moreover, the additional tapered smoothing term can be pushed into the augmented Lagrangian, and only alters the x-update, now becoming:

$$x \leftarrow (\tilde{P}^T\tilde{P} + \gamma M^T M + \rho\tilde{D})^{-1}\big(\tilde{P}^T y + \rho\tilde{D}^T(\alpha + u)\big),$$

where M is the matrix $W^{(t)}D^{(1)}$ in the tapered penalty in (3.10) times the linear interpolant matrix.

B.2 Additional evaluation results

Figures B.1 and B.2 are analogous to Figures 3.11 and 3.12, but with the inclusion of the Google-AA sensor. Similarly, Figures B.3 and B.4 are the counterparts to Figures 3.13 and 3.14, but with the inclusion of claims-based sensors.

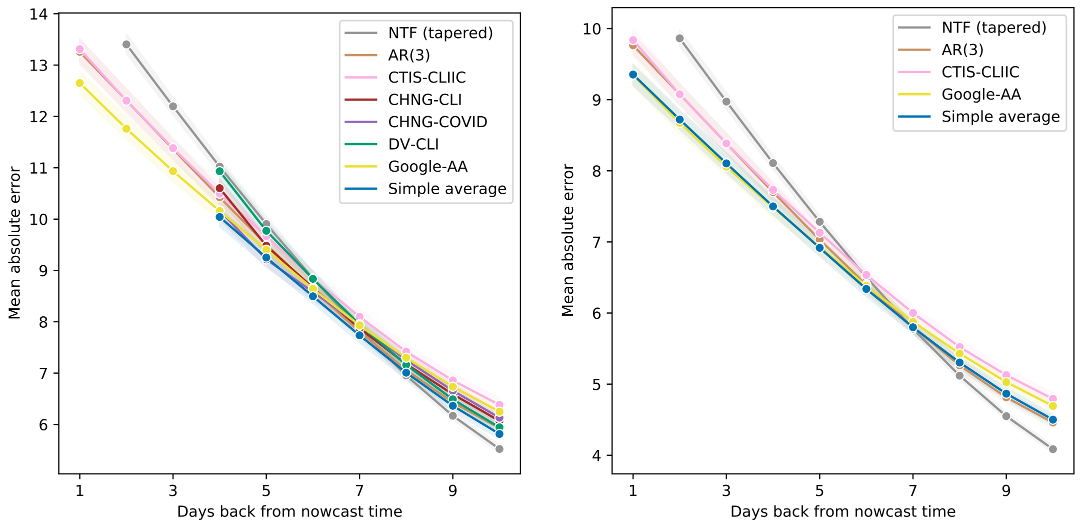


Figure B.1: *As in Figure 3.11, but including Google-AA.*

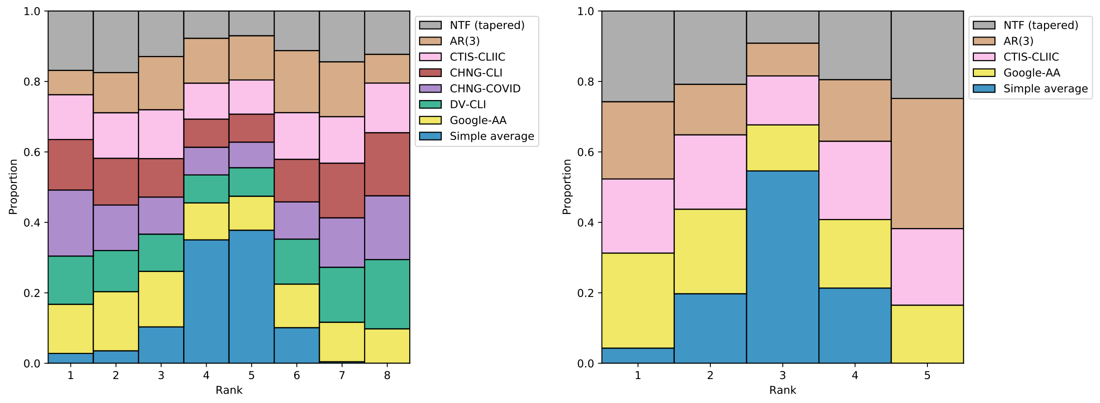


Figure B.2: *As in Figure 3.12, but including Google-AA.*

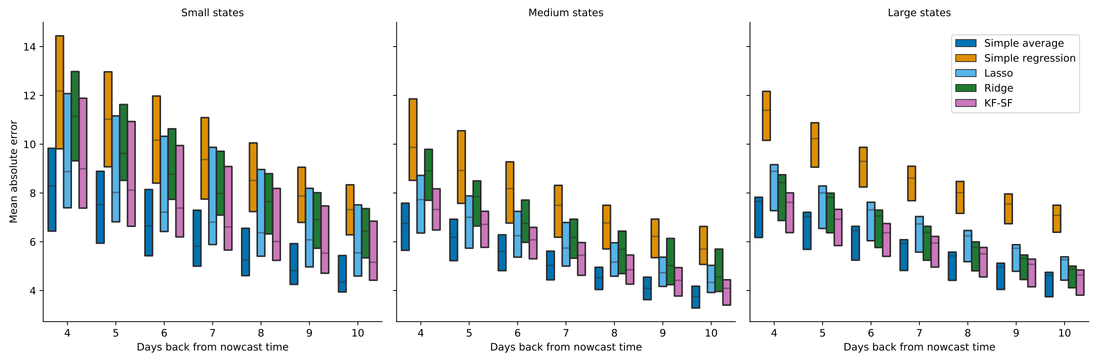


Figure B.3: *As in Figure 3.13, but including claims-based signals.*

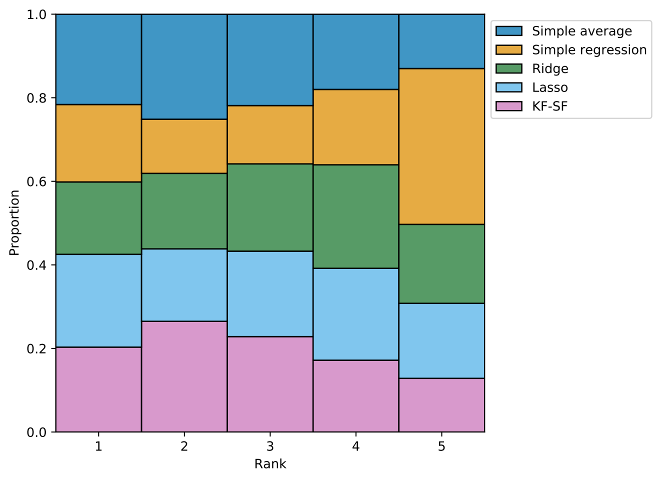


Figure B.4: *As in Figure 3.14, but including claims-based signals.*

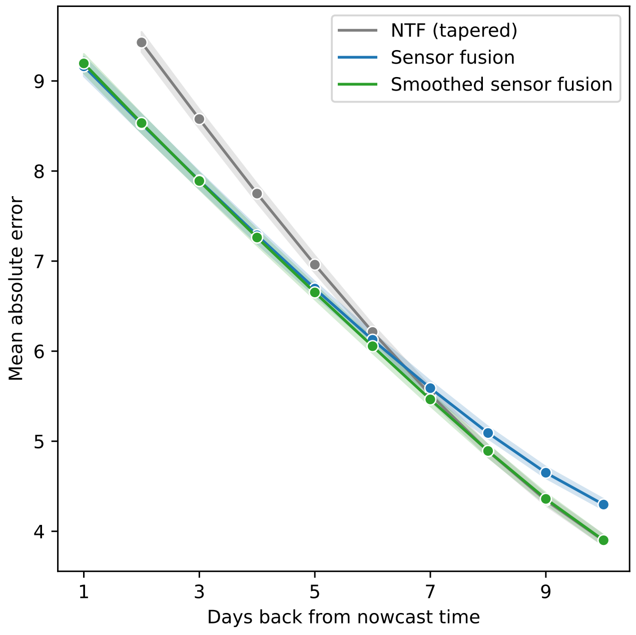


Figure B.5: *As in Figure 3.11, but comparing post hoc smoothed sensor fusion to the original sensor fusion and NTF methods. Smoothed sensor fusion method achieves the best performance.*

Appendix C

Additional Reconvolution Results

Figures C.1 and C.2 are analogous to Figure 4.5, but using trend filtering and natural trend filtering (tapered), deconvolution methods, respectively. Similarly, Figures C.3 and C.4 are analogous to Figure 4.6, but with the different deconvolution methods.

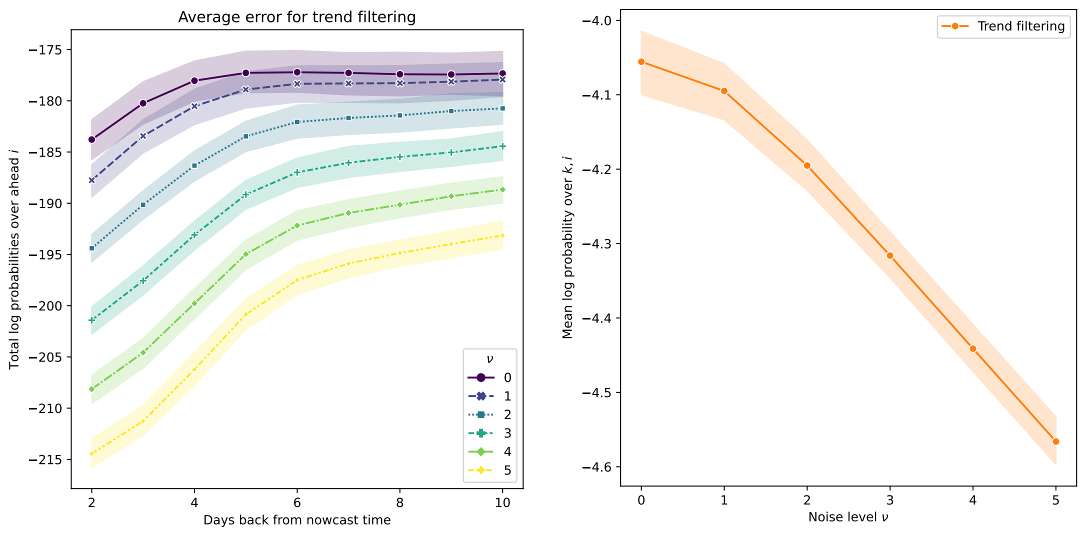


Figure C.1: *As in Figure 4.5, but for trend filtering deconvolution.*

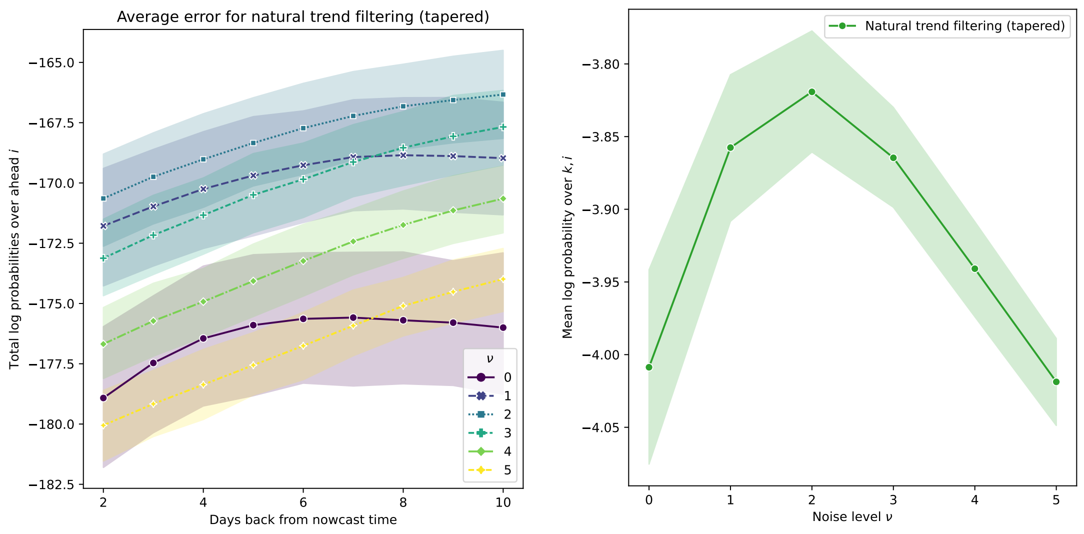


Figure C.2: *As in Figure 4.5, but for natural trend filtering deconvolution with a tapered penalty.*

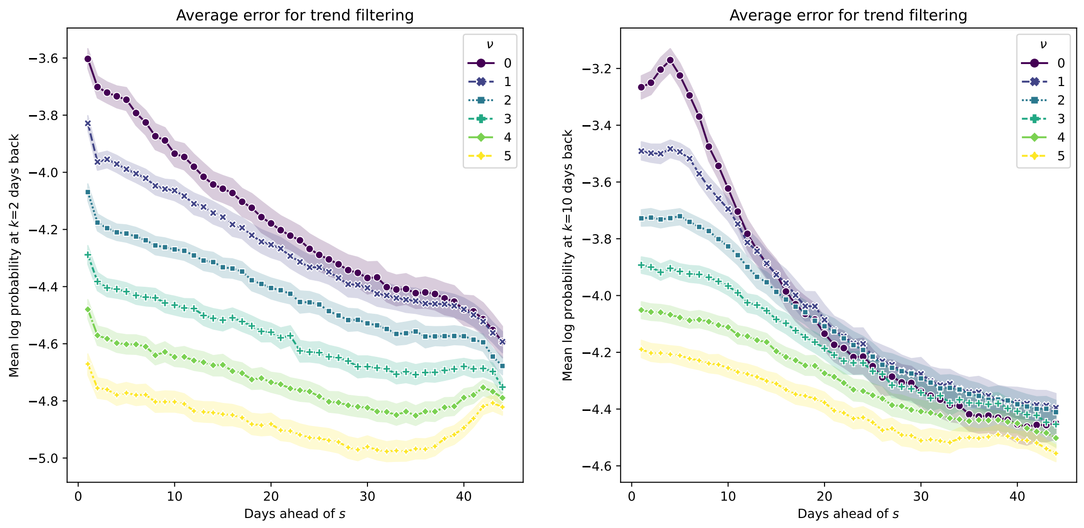


Figure C.3: *As in Figure 4.6, but for trend filtering deconvolution with $k = 2, 10$. The left panel is fixed at $k = 2$, which is the first lag available for this method.*

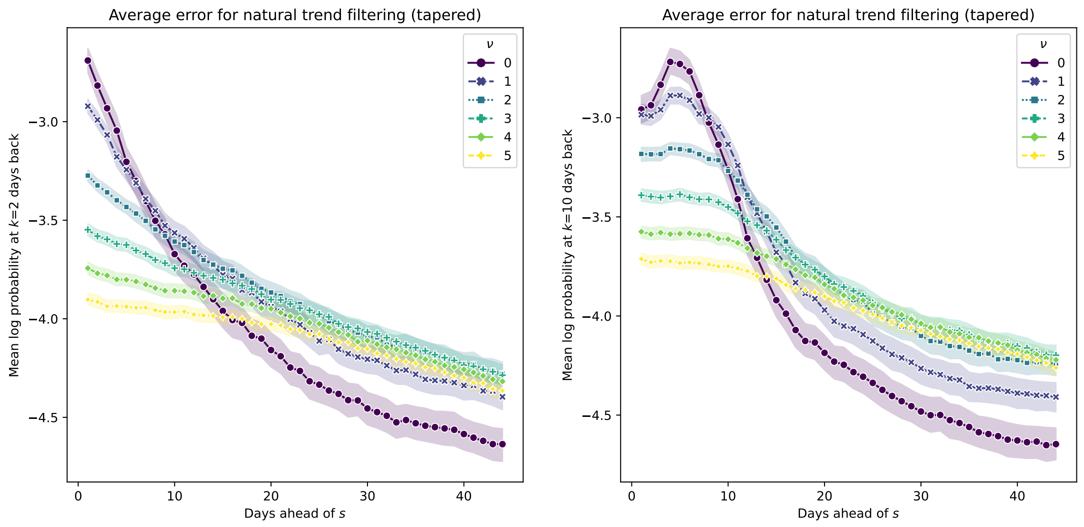


Figure C.4: *As in Figure 4.6, but for natural trend filtering deconvolution with a tapered penalty with $k = 2, 10$. The left panel is fixed at $k = 2$, which is the first lag available for this method.*

Bibliography

Sam Abbott, Joel Hellewell, Robin N. Thompson, Katharine Sherratt, Hamish P. Gibbs, Nikos I. Bosse, James D. Munday, Sophie Meakin, Emma L. Doughty, June Young Chun, Yung-Wai Desmond Chan, Flavio Finger, Paul Campbell, Akira Endo, Carl A. B. Pearson, Amy Gimma, Tim Russell, CMMID COVID modelling group, Stefan Flasche, Adam J. Kucharski, Rosalind M. Eggo, and Sebastian Funk. Estimating the time-varying reproduction number of SARS-CoV-2 using national and subnational case counts. *Wellcome Open Research*, 5(112), 2020. 3.3, 4.3

Patrice Abry, Nelly Pustelnik, Stéphane G. Roux, Pablo Jensen, Patrick Flandrin, Rémi Gribonval, Charles-Gérard Lucas, Éric Guichard, Pierre Borgnat, and Nicolas Garnier. Spatial and temporal regularization to estimate COVID-19 reproduction number $R(t)$: Promoting piecewise smoothness via convex optimization. *PLOS ONE*, 15(8): 1–22, 08 2020. 4.3

Aarah F. Ackley, Sarah Pilewski, Vladimir S. Petrovic, Lee Worden, Erin Murray, and Travis C. Porco. assessing the utility of a smart thermometer and mobile application as a surveillance tool for influenza and influenza-like illness. *Health Informatics Journal*, 26(3):2148–2158, 2020. 3.2

Emily L. Aiken, Andre T. Nguyen, Cecile Viboud, and Mauricio Santillana. Toward the use of neural networks for influenza prediction at multiple spatial resolutions. *Science Advances*, 7(25), 2021. doi: 10.1126/sciadv.abb1237. 1.1

Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, 1979. 2.7

Shailesh Bavadekar, Andrew Dai, John Davis, Damien Desfontaines, Ilya Eckstein, Katie Everett, Alex Fabrikant, Gerardo Flores, Evgeniy Gabrilovich, Krishna Gadepalli, Shane Glass, Rayman Huang, Chaitanya Kamath, Dennis Kraft, Akim Kumok, Hinali Marfatia, Yael Mayer, Benjamin Miller, Adam Pearce, Irippuge Milinda Perera, Venky Ramachandran, Karthik Raman, Thomas Roessler, Izhak Shafran, Tomer Shekel, Charlotte Stanton, Jacob Stimes, Mimi Sun, Gregory Wellenius, , and Masrour

Zoghi. Google COVID-19 search trends symptoms dataset: Anonymization process description. arXiv: 2009.01265, 2020. 3.7.2

Luis M. A. Bettencourt and Ruy M. Ribeiro. Real time Bayesian estimation of the epidemic potential of emerging infectious diseases. *PLOS ONE*, 3(5):e2185, 2008. 3.3, 4.3

Johannes Bracher, Evan L. Ray, Tilmann Gneiting, and Nicholas G. Reich. Evaluating epidemic forecasts in an interval format. *PLOS Computational Biology*, 17(2):1–15, 02 2021. URL https://doi.org/10.1371/journal.pcbi.1008618. 4.3

Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996. 2.1.3

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 2.6.1

Logan C. Brooks. *Pancasting: Forecasting epidemics from provisional data*. PhD thesis, Carnegie Mellon University, 2020. 1.1, 3.2, 3.3, 4.3

Logan C. Brooks, David C. Farrow, Sangwon Hyun, Ryan J. Tibshirani, and Roni Rosenfeld. Flexible modeling of epidemics with an empirical Bayes framework. *PLOS Computational Biology*, 11(8):1–18, 08 2015. doi: 10.1371/journal.pcbi.1004382. URL https://doi.org/10.1371/journal.pcbi.1004382. 1.1

Logan C. Brooks, David C. Farrow, Sangwon Hyun, Ryan J. Tibshirani, and Roni Rosenfeld. Nonmechanistic forecasts of seasonal influenza with iterative one-week-ahead distributions. *PLOS Computational Biology*, 14(6):1–29, 06 2018a. URL https://doi.org/10.1371/journal.pcbi.1006134. 1.1

Logan C. Brooks, David C. Farrow, Sangwon Hyun, Ryan J. Tibshirani, and Roni Rosenfeld. Nonmechanistic forecasts of seasonal influenza with iterative one-week-ahead distributions. *PLOS Computational Biology*, 14(6):1–29, 2018b. 4.3

P. J. Brown. Multivariate calibration. *Journal of the Royal Statistical Society: Series B*, 44(3):287–321, 1982. 2.1.3

Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons, 2012. 1.2, 2.1.3

John S. Brownstein, Clark C. Freifeld, and Lawrence C. Madoff. Digital disease detection — harnessing the web for public health surveillance. *New England Journal of Medicine*, 360(21):2153–2157, 2009. 3.2

Sandra J. Carlson, Craig B. Dalton, Michelle T. Butler, John Fejsa, Elissa Elvidge, and David N. Durrheim. Flutracking weekly online community survey of influenza-like illness annual report 2011 and 2012. *Communicable diseases intelligence quarterly report*, 37(4):E398–406, 2013. 3.2

Centers for Disease Control and Prevention, COVID-19 Response. COVID-19 Case Surveillance Public Use Data. https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data/vbim-akqf, 2020a. Data accessed on November 3, 2021. 3.2.1, 3.4, 3.5.2, 4.2.2

Centers for Disease Control and Prevention, COVID-19 Response. COVID-19 Case Surveillance Restricted Access Detailed Data. https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Restricted-Access-Detai/mbd7-r32t, 2020b. Data accessed on November 3, 2021. 3.5.2

Vivek Charu, Scott Zeger, Julia Gog, Ottar N. Bjørnstad, Stephen Kissler, Lone Simonsen, Bryan T. Grenfell, and Cécile Viboud. Human mobility and the spatial transmission of influenza in the United States. *PLOS Computational Biology*, 13(2):1–23, 02 2017. 3.2

Melanie H. Chitwood, Marcus Russi, Kenneth Gunasekera, Joshua Havumaki, Virginia E. Pitzer, Joshua A. Salomon, Nicole Swartwood, Joshua L. Warren, Daniel M. Weinberger, and Ted Cohen. Reconstructing the course of the COVID-19 epidemic over 2020 for US states and counties: results of a Bayesian evidence synthesis model. *medRxiv*, 2021. doi: 10.1101/2020.06.17.20133983. 3.2.2, 3.3, 4.3

Hyunyoung Choi and Hal Varian. Predicting the Present with Google Trends. *Economic Record*, 88(s1):2–9, 2012. 1.1

Anne Cori, Neil M. Ferguson, Christophe Fraser, and Simon Cauchemez. A new framework and software to estimate time-varying reproduction numbers during epidemics. *American Journal of Epidemiology*, 178(9):1505–1512, 2013. 3.3, 4.3

Estee Y. Cramer, Evan L. Ray, Velma K. Lopez, Johannes Bracher, Andrea Brennen, Alvaro J. Castro Rivadeneira, others, and Nicholas G. Reich. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the US. *medRxiv*, 2021. doi: 10.1101/2021.02.03.21250974. 4.3

Chester Curme, Tobias Preis, H. Eugene Stanley, and Helen Susannah Moat. Quantifying the semantics of search behavior before stock market moves. *Proceedings of the National Academy of Sciences*, 111(32):11600–11605, 2014. 1.1

H. W. J. Debeye and P. Van Riel. L_p-norm deconvolution. *Geophysical Prospecting*, 38 (4):381–403, 1990. 3.3

Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–544, 2020. 3.2.1, 3.4, 4.2.2

Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009. 4.7

Arnaud Doucet, Nando De Freitas, and Neil James Gordon. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001. 4.7

Geir Evensen. Sequential data assimilation with nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):143–162, 1994. 2.1.1

David C. Farrow. *Modeling the Past, Present, and Future of Influenza*. PhD thesis, Carnegie Mellon University, 2016. 1.1, 1.2, 1.2, 2.4, 3.2, 3.3, 5, 4.3, 5.1, A.4.1

David C. Farrow, Logan C. Brooks, Aaron Rumack, Ryan J. Tibshirani, and Roni Rosenfeld. Delphi Epidata API. https://github.com/cmu-delphi/delphi-epidata, 2015. 1.2

Jerome Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1190–1232, 2001. 5.2

Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009. 1.1, 3.2

Edward Goldstein, Jonathan Dushoff, Junling Ma, Joshua B. Plotkin, David J. D. Earn, and Marc Lipsitch. Reconstructing influenza incidence by deconvolution of daily mortality time series. *Proceedings of the National Academy of Sciences*, 106(51): 21825–21829, 2009. 3.3, 4.3

Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F, Radar and Signal Processing*, 140(2):107–113, 1993. 2.1.1

Katelyn M. Gostic, Lauren McGough, Edward B. Baskerville, Sam Abbott, Keya Joshi, Christine Tedijanto, Rebecca Kahn, Rene Niehus, James A. Hay, Pablo M. De Salazar, Joel Hellewell, Sophie Meakin, James D. Munday, Nikos I. Bosse, Katharine Sherrat, Robin N. Thompson, Laura F. White, Jana S. Huisman, Jérémie Scire, Sebastian Bonhoeffer, Tanja Stadler, Jacco Wallinga, Sebastian Funk, Marc Lipsitch, and Sarah Cobey. Practical considerations for measuring the effective reproductive number, r_t. *PLOS Computational Biology*, 16:1–21, 12 2020. 3.3, 4.3

Iwona Hawryluk, Henrique Hoeltgebaum, Swapnil Mishra, Xenia Miscouridou, Ricardo P. Schnekenberg, Charles Whittaker, Michaela Vollmer, Seth Flaxman, Samir Bhatt, and Thomas A. Mellan. Gaussian process nowcasting: Application to COVID-19 mortality reporting. In *Conference on Uncertainty in Artificial Intelligence*, 2021. 3.3

H. Heffes. The effect of erroneous models on the Kalman filter response. *IEEE Transactions on Automatic Control*, 11(3):541–543, 1966. 1.2

Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4): 599–653, 2000. 1.1

P. L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998. 2.1.1

Maria Jahja, David C. Farrow, Roni Rosenfeld, and Ryan J. Tibshirani. Kalman Filter, Sensor Fusion, and Constrained Regression: Equivalences and Insights. In *Advances in Neural Information Processing Systems*, pages 13187–13196, 2019. 1.1, 1.2, 3.2, 3.3, 5, 3.7.3

Maria Jahja, Andrew Chin, and Ryan J. Tibshirani. Real-Time Estimation of COVID-19 Infections: Deconvolution and Sensor Fusion. *Statistical Science*, 2022. 1.2

Nicholas Johnson. A dynamic programming algorithm for the fused lasso and l_0-segmentation. *Journal of Computational and Graphical Statistics*, 22(2):246–260, 2013. B.1

Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition*, 1997. 2.1.1

Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 1.2, 2.1.1

Edward L. Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958. 1.2, 3.6.2

Taha A. Kass-Hout and Hend Alhinnawi. Social media in public health. *British Medical Bulletin*, 108(1):5–24, 2013. 3.2

Taha A. Kass-Hout and Xiaohui Zhang. *Biosurveillance: Methods and Case Studies.* CRC Press, 2011. 3.2

William Ogilvy Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A*, 115 (772):700–721, 1927. 1.1

Sequoia I. Leuba, Reza Yaesoubi, Marina Antillon, Ted Cohen, and Christoph Zimmer. Tracking and predicting U.S. influenza activity with a real-time surveillance network. *PLOS Computational Biology*, 16(11):1–14, 11 2020. 3.2

Daniel J. McDonald, Jacob Bien, Alden Green, Addison J. Hu, Nat DeFries, Sangwon Hyun, Natalia L. Oliveira, James Sharpnack, Jingjing Tang, Robert Tibshirani, Valérie Ventura, Larry Wasserman, and Ryan J. Tibshirani. Can auxiliary indicators improve COVID-19 forecasting and hotspot prediction? *Proceedings of the National Academy of Sciences*, 118(51):e2111453118, 2021. doi: 10.1073/pnas.2111453118. 1.1, 3.4.1, 3.4.2, 3.7, 4.3

Sarah F. McGough, Michael A. Johansson, Marc Lipsitch, and Nicolas A. Menzies. Nowcasting by Bayesian smoothing: A flexible, generalizable model for real-time epidemic tracking. *PLOS Computational Biology*, 16(4):1–20, 04 2020. 3.3

David J. McIver and John S. Brownstein. Wikipedia usage estimates prevalence of influenza-like illness in the United States in near real-time. *PLOS Computational Biology*, 10(4):e1003581, 2014. 1.1, 3.2

R. Mehra. On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on Automatic Control*, 15(2):175–184, 1970. doi: 10.1109/TAC.1970.1099422. 5.1

A. H. Mohamed and K. P. Schwarz. Adaptive Kalman filtering for INS/GPS. *Journal of Geodesy*, 73(4):193–203, 1999. 5.1

Alan V. Oppenheim and George C. Verghese. *Signals, Systems and Inference.* Pearson, 2017. 1.2, 3.3

Peter Van Overshee and Bart De Moor. *Subspace Identification for Linear Systems.* Kluwer Academic, 1996. 2.7

Barbara Pascal, Patrice Abry, Nelly Pustelnik, Stéphane G. Roux, Rémi Gribonval, and Patrick Flandrin. Nonsmooth convex optimization to estimate the Covid-19 reproduction number space-time evolution with robustness against low quality data. arXiv: 2109.09595, 2021. 4.3

Michael J. Paul and Mark Dredze. Social monitoring for public health. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 9(5):1–183, 2017. 3.2

Tobias Preis, Daniel Reith, and H. Eugene Stanley. Complex Dynamics of Our Economic Life on Different Scales: Insights from Search Engine Query Data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1933):5707–5719, 2010. 1.1

Jennifer M. Radin, Nathan E. Wineinger, Eric J Topol, and Steven R Steinhubl. Harnessing wearable device data to improve state-level real-time surveillance of influenza-like illness in the USA: A population-based study. *The Lancet Digital Health*, 2(2): e85–e93, 2020. 3.2

Aaditya Ramdas and Ryan J. Tibshirani. Fast and flexible ADMM algorithms for trend filtering. *Journal of Computational and Graphical Statistics*, 25(3):839–858, 2016. 3.5.3, B.1

Nicholas G. Reich. It's logical to think that rises in #COVID19 cases in the US would precede rises in hospitalizations. After all, one has to get infected and typically become symptomatic before being hospitalized. But the data tell a slightly different story. https://twitter.com/reichlab/status/1471545304814178305, 12 2021. Tweet by @reichlab. 3.8.4

Reich Lab. The COVID-19 Forecast Hub. https://covid19forecasthub.org, 2020. 3.2.1

Alex Reinhart, Logan Brooks, Maria Jahja, Aaron Rumack, Jingjing Tang, Sumit Agrawal, Wael Al Saeed, Taylor Arnold, Amartya Basu, Jacob Bien, Ángel A. Cabrera, Andrew Chin, Eu Jing Chua, Brian Clark, Sarah Colquhoun, Nat DeFries, David C. Farrow, Jodi Forlizzi, Jed Grabman, Samuel Gratzl, Alden Green, George Haff, Robin Han, Kate Harwood, Addison J. Hu, Raphael Hyde, Sangwon Hyun, Ananya Joshi, Jimi Kim, Andrew Kuznetsov, Wichada La Motte-Kerr, Yeon Jin Lee, Kenneth Lee, Zachary C. Lipton, Michael X. Liu, Lester Mackey, Kathryn Mazaitis, Daniel J. McDonald, Phillip McGuinness, Balasubramanian Narasimhan, Michael P. O'Brien, Natalia L. Oliveira, Pratik Patil, Adam Perer, Collin A. Politsch, Samyak Rajanala, Dawn Rucker, Chris Scott, Nigam H. Shah, Vishnu Shankar, James Sharpnack, Dmitry Shemetov, Noah Simon, Benjamin Y. Smith, Vishakha Srivastava, Shuyi Tan, Robert Tibshirani, Elena Tuzhilina, Ana Karina Van Nortwick, Valérie Ventura, Larry Wasserman, Benjamin Weaver, Jeremy C. Weiss, Spencer Whitman, Kristin Williams, Roni Rosenfeld, and Ryan J. Tibshirani. An open repository of real-time COVID-19 indicators. *Proceedings of the National Academy of Sciences*, 118(51):e2111452118, 2021. doi: 10.1073/pnas.2111452118. 3.4.1, 3.4.1, 3.4.2, 3.7, 3.7.1, 3.8.4, 4.2.3

Roni Rosenfeld and Ryan J. Tibshirani. Epidemic tracking and forecasting: Lessons learned from a tumultuous year. *Proceedings of the National Academy of Sciences*, 118(51):e2111456118, 2021. doi: 10.1073/pnas.2111456118. 3.2.1

L.I. Rudin and S. Osher. Total variation based image restoration with free local constraints. In *International Conference on Image Processing*, volume 1, pages 31–35, 1994. 3.3

Aaron Rumack. Doctor Visits, Day-of-Week Adjustment, 2020. URL https://cmu-delphi.github.io/delphi-epidata/api/covidcast-signals/doctor-visits.html. 4.2.3

Marcel Salathé, Linus Bengtsson, Todd J. Bodnar, Devon D. Brewer, John S. Brownstein, Caroline Buckee, Ellsworth M. Campbell, Ciro Cattuto, Shashank Khandelwal, Patricia L. Mabry, and Alessandro Vespignani. Digital epidemiology. *PLOS Computational Biology*, 8(7):1–3, 2012. 3.2

Joshua A. Salomon, Alex Reinhart, Alyssa Bilinski, Eu Jing Chua, Wichada La Motte-Kerr, Minttu M. Rönn, Marissa Reitsma, Katherine Ann Morris, Sarah LaRocca, Tamer Farag, Frauke Kreuter, Roni Rosenfeld, and Ryan J. Tibshirani. The COVID-19 Trends and Impact Survey: Continuous real-time measurement of COVID-19 symptoms, risks, protective behaviors, testing and vaccination. *Proceedings of the National Academy of Sciences*, 118(51):e2111454118, 2021. doi: 10.1073/pnas.2111454118. 3.4.1

Mauricio Santillana, André T. Nguyen, Mark Dredze, Michael J. Paul, Elaine O. Nsoesie, and John S Brownstein. Combining search, social media, and traditional data sources to improve influenza surveillance. *PLOS Computational Biology*, 11(10):e1004513, 2015. 3.2

Mauricio Santillana, André T. Nguyen, Tamara Louie, Anna Zink, Josh Gray, Iyue Sung, and John S. Brownstein. Cloud-based electronic health records for real-time, region-specific influenza surveillance. *Scientific Reports*, 6(1):1–8, 2016. 1.1, 3.2

Jeffrey Shaman and Sasikiran Kandula. Improved Discrimination of Influenza Forecast Accuracy Using Consecutive Predictions. *PLoS Currents*, 7, 2015. 1.1

Gerald L. Smith, Stanley F. Schmidt, and Leonard A. McGee. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle. *National Aeronautics and Space Administration Tech Report*, 1962. 2.1.1

Mark S. Smolinski, Adam W. Crawley, Kristin Baltrusaitis, Rumi Chunara, Jennifer M. Olsen, Oktawia Wójcik, Mauricio Santillana, André T. Nguyen, and John S. Brownstein. Flu Near You: Crowdsourced symptom reporting spanning 2 influenza seasons. *American Journal of Public Health*, 105(10):2124–2130, 2015. 3.2

Kevin Systrom, Thomas Vladek, and Mike Krieger. Rt.live. https://github.com/rtcovidlive/covid-model, 2020. 3.3

Howard L. Taylor, Stephen C. Banks, and John F. McCoy. Deconvolution with the ℓ_1 norm. *Geophysics*, 44(1):39–52, 1979. 3.3

R. N. Thompson, J. E. Stockwin, R. D. van Gaalen, J. A. Polonsky, Z. N. Kamvar, P. A. Demarsh, E. Dahlqwist, S. Li, E. Miguel, T. Jombart, J. Lessler, S. Cauchemez, and A. Cori. Improved inference of time-varying reproduction numbers during infectious disease outbreaks. *Epidemics*, 29:100356, 2019. 3.3, 4.3

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996. 5.2

Ryan J. Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 2014. 3.5.3, 2

Ryan J. Tibshirani. Divided Differences, Falling Factorials, and Discrete Splines: Another Look at Trend Filtering and Related Problems. *arXiv preprint arXiv:2003.03886*, 2020. 3.5.3, 3.6.1, B.1

Cécile Viboud, Vivek Charu, Donald Olson, Sébastien Ballesteros, Julia Gog, Farid Khan, Bryan Grenfell, and Lone Simonsen. Demonstrating the use of high-volume electronic medical claims data to monitor local and regional influenza activity in the US. *PLOS ONE*, 9(7):1–12, 07 2014. 1.1, 3.2

Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series.* The MIT Press, 1964. 3.3

David Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. 2.1.3

Yuexin Wu, Yiming Yang, Hiroshi Nishiura, and Masaya Saitoh. Deep Learning for Epidemiological Predictions. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1085–1088, 2018. 1.1

Cheng-Yi Yang, Ray-Jade Chen, Wan-Lin Chou, Yuarn-Jang Lee, and Yu-Sheng Lo. An integrated influenza surveillance framework based on national influenza-like illness incidence and multiple hospital electronic medical records for early prediction of influenza epidemics: Design and evaluation. *Journal of Medical Internet Research*, 21 (2):e12341, 2019. 3.2

Shihao Yang, Mauricio Santillana, and S. C. Kou. Accurate estimation of influenza epidemics using Google search data via ARGO. *Proceedings of the National Academy of Sciences*, 112(47):14473–14478, 2015. 1.1, 3.2, 3.3

Wan Yang, Alicia Karspeck, and Jeffrey Shaman. Comparison of Filtering Methods for the Modeling and Retrospective Forecasting of Influenza Epidemics. *PLOS Computational Biology*, 10(4):1–15, 04 2014. doi: 10.1371/journal.pcbi.1003583. 1.1

Qingyu Yuan, Elaine O. Nsoesie, Benfu Lv, Geng Peng, Rumi Chunara, and John S. Brownstein. Monitoring influenza epidemics in China with search query from Baidu. *PLOS ONE*, 8(5):e64323, 2013. 1.1