Modeling the spread of COVID-19 in the U.S using dynamic Bayesian Factor Models

Loc Cao, Varshant Dhar, Visweswaran Ravikumar

April 27, 2020

1 Introduction

The ongoing COVID-19 (Coronavirus disease 2019) pandemic has impacted the lives of millions of people around the world. As of April 17, 2020, there are about 700,000 confirmed cases in the United States with more than 30,000 deaths. While doctors, patients and essential workers are fighting in the front line of this war, the situation in the home front is not necessary easier: businesses are closed, public events are cancelled, unemployment rate soars and people stay at home. But this is hardly close to the end. According to the CDC (Center for Disease Control and Prevention), though different states are currently affected to different extents, the U.S as a country is still in the acceleration phase of of the pandemic.

Our goal in this project is to accurately characterize temporally evolving COVID-19 incidence across regions using the *New York Times'* COVID-19 dataset[1]. We proposed to use Factor analysis models to exploit the high rate of spatial and temporal redundancy in the data using low dimensional models, and to use a flexible Bayesian nonparametric covariance regression framework to map these location-specific trajectories onto a lower-dimensional subspace. This is done through a latent factor model with predictor-dependent factor loadings, inspired by the model proposed by Fox.E et al. [2].

Through exploratory data analysis (section 2), we addressed some inherent issues with the data set, motivating our choice of use of the Bayesian Dynamic factor framework. The model is formalized, where we specified the priors for each parameters (section 3), and the Gibb sampling algorithm provides greater details about how to structure the sampling process (section 4). We then presented our implementation result using the New York Times Coronavirus dataset (section 5). A sensitivity analysis was then provided to assess the performance of the sampler (section 6). Finally, the limitation of our study is discussed in further details in section 7.

2 Exploratory Data Analysis

The New York Times COVID-19 dataset features the cumulative count of confirmed coronavirus cases across the U.S since January 21, 2020 for 50 states and the District of Columbia. As a remark, the counts are based on the best reporting up to the moment the update is published, and that they are subject to future revision. This implies that some state experiences negative changes in the number of confirmed cases due to official revisions published by the cases. As an example, on April 12, 2020, the cumulative number of confirmed cases for Georgia decreased by 148 due to a revision in the classification methodology, exluding cases labeled by the state as "non-Georgia resident." The dataset is also characterized by substantial missing data due to a combination of the lack of testing as well as a delayed government response. As a example,

[1]`https://github.com/nytimes/covid-19-data`
[2]`http://www.jmlr.org/papers/volume16/fox15a/fox15a.pdf`

38 out of 51 regions examined reported zero cases throughout the first 40 days of the outbreak. Therefore, our model needs to be flexible and robust enough to account for potential missing and inaccurate data entries.

Another issue with the COVID-19 dataset is that the amplitude of the outbreak varies significantly, ranging from a few hundreds to 200,000 cases between the most affected and the least affected areas. Therefore, to make the observations comparable across states, we standardized the cumulative counts by the square root of the maximum variance of the counts over all the regions.

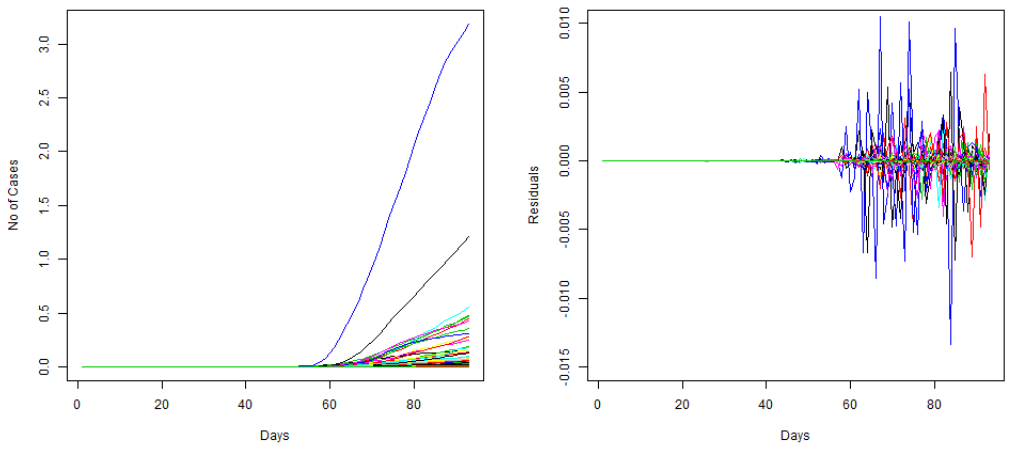

Figure 1: Plot of spline fitting of the number of cases by states, and the corresponding residuals

As our first attempt to better understand the structure of the dataset, we fitted spline curve to the cumulative number of cases by states. The residuals from spline fitting in Figure 1 shows that the errors are not independent and identically distributed across time, justifying our incentives in taking into account of possible heteroscedasticity inherent to the data.

To better understand the correlation structure of the dataset, we looked at correlation from three different lenses: through the cumulative counts, through the nominal rate of changes, defined as the net change in the number of newly reported cases per day, and finally through the relative rate of changes, defined as the percentage change in the number of newly reported cases per day. While we found the cumulative counts to be a holistic measure of the outbreak, the nominal rate put emphasis on the magnitude of the outbreak, and the relative rate focused on the state of the outbreak at a certain location.

We examined the correlation snapshot based on the three metrics proposed above utilizing all 51 regions jointly and over the entire time course. Note, geographically distant regions, in the classical Euclidean sense, could still exhibit significant correlation. For instance, Figure 2(A) illustrates that while the amplitude the outbreak between contiguous states tend to be strongly correlated (such is the case for the western states bordered with California), high correlations between non-neighboring states cannot not be ignored (for example, California and Maryland). With regards to the relative rate of change in infected numbers, Figure 2(A) shows that the spread of the outbreak between contiguous states may not be as significant as that between some distant states (such is the case of Georgia and New York). We speculated that similar to influenza and seasonal flu, Covid-19 can spread easily between non-contiguous regions through major airport hubs, such as Atlanta and New York. Also, contiguous states may experience the outbreak very differently from the neighboring states, and thus we do not explicitly incorporate geographical distances into our model.

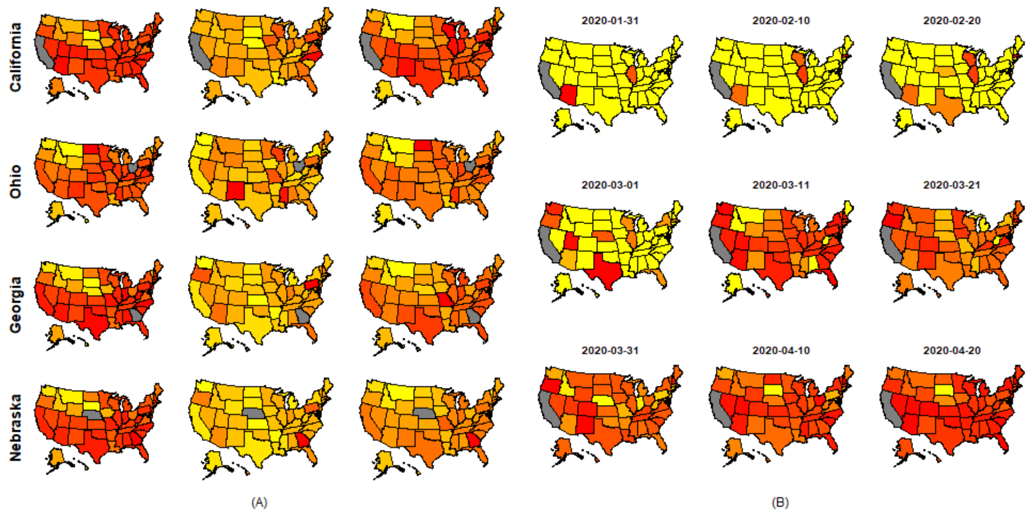

Figure 2: (A) Correlation plot using cumulative counts, relative rates and nominal rates (from left to right), (B) Correlation plot as a function of time, using cumulative counts

Figure 2(B) provides strong evidence how the spatial structure may change through time and that there are redundancies in the cumulative counts data across time. The correlation snapshots of the cumulative counts for the state of California shows that there are relative few change between April 04, 2020 and April 20, 2020. Therefore, our model for this dataset needs to incorporate low-rank approximations to fully exploit this redundancy in information. Though promising, one of the major flaws of these correlation snapshots in Figure 2 is missing data. Indeed, to obtain the chart, we assumed that missing data had zero correlation with existing data points, which is not always true.

Based on the observations above, our model needs to be flexible enough to accommodate missing and erroneous data entries, while complex enough to capture the temporal fluctuations and heteroscedasticity in the dataset. Also, we don't want the model to explicitly enforce any spatial constraints and be able to exploit the spatial and temporal redundancy structure embedded in the data. These considerations motivated us to use the Bayesian Dynamic Factor models approach to study this dataset.

3 Model Specification

Let the response vector, y_i be the cumulative number of COVID-19 confirmed cases or deaths across the 50 states along with the District of Columbia starting from January 21, 2020 to April 17, 2020. Using multivariate Gaussian non-parametric mean-covariance regression model,

$$y_i = \mu(x_i) + \epsilon_i \tag{1}$$

we want to find a low-rank approximation of the model above, where x_i are the time indices, $\epsilon_i \sim N_p(0, \Sigma(x_i))$, and $\mathbf{y}_i$'s are the cumulative number of confirmed cases in p regions at time x_i.

$$\mathbf{y}_i = \Lambda(x_i)\eta_i + \epsilon_i \tag{2}$$

where $\Lambda(x_i) \in R^{p\times k}, k \ll p$ is the factor loading matrix corresponding to time x_i, $\eta_i \in R^{k\times 1}$ are the latent, not observed factors associated with y_i. We assumed that $\eta_i \sim N_k(\psi(x_i), I_k)$, where $\psi(x_i)$ is the latent factor mean function and $\epsilon_i \sim N_p(0, \Sigma_0)$, for Σ_0 a diagonal not necessary constant multiple of I_p.

Note, conditioned on η_i, we have

$$y_i|\eta_i \sim N_p\left(\Lambda(x_i)\psi(x_i), \Lambda(x_i)\Lambda(x_i)^T + \Sigma_0\right) \tag{3}$$

Similar to the idea proposed in Fox and Dunson, we were looking for a model that is complex enough to incorporate all the criteria spelled out in the Exploratory Analysis section though fairly simple to implement. This motivated us to use a Bayesian approach, which requires us to impose priors on the parameters.

3.1 Prior for factor loadings matrix Λ

Even if the number of factors k were small, since Λ is a p x k matrix, specifying every element of this matrix as a function of time would be computationally very expensive and infeasible. There the loadings matrix is set to be a sparse weighted linear combination of a much smaller set of $L \times k$ predictor-dependent basis elements ξ_{lh},

$$\Lambda(x_i) = \Theta\xi(x_i) \tag{4}$$

where $\Theta \in R^{p\times L}$, and $\xi(x_i) = \{\xi_{lh}(x_i), l = 1, ..., L, h = 1, ..., k\}$. We assumed each $\xi_{lh} \sim \text{GP}(0, c)$, where c is the κ squared exponential covariance kernel. As a remark, L is often referred to as truncation parameter.

To allow for an adaptive choice of basis size, a multiplicative gamma process shrinkage prior for Θ is used where the elements of the Θ matrix are assigned inverse-gamma priors given by,

$$\theta_{jl} \sim N(0, \phi_{jl}^{-1}\tau_l^{-1}) \tag{5}$$

where ϕ_{jl} is a local precision parameter and $\tau_l = \prod_{h=1}^{l}\delta_h$ a column-specific multiplier. This results in the columns of the Θ matrix being increasingly penalized to zero, and allows us to adaptively learn the size of our basis set $\xi(x_i)$ from the data.

The precision terms have the usual gamma priors with $\phi_{jl} \sim \text{Ga}\left(\frac{\gamma}{2}, \frac{\gamma}{2}\right)$, $\delta_1 \sim \text{Ga}(a_1, 1)$ and $\delta_h \sim \text{Ga}(a_2, 1)$ for $h \geq 2$ and $a_2 > 1$, with Ga being a Gamma distribution.

3.2 Gaussian Process Prior

To allow for sufficient flexibilty in the model, Gaussian process (GP) priors are used to model the covariate dependent basis elements $\xi(x_i)$ as well as the factor means $\psi(x_i)$ as non-parametric functions of the time indices x_i. Let $\text{GP}(m, c)$ denote the Gaussian Process with mean function m and covariance kernel c. By definition, a function $f : X \to R \sim \text{GP}(m, c)$ if and only if for all n, and $x_1, ..., x_n$,

$$p(f(x_1), ..., f(x_n)) \sim N_n(\mu, \mathbf{K})$$

with $\mu = [m(x_1), ..., m(x_n)]$ and $\mathbf{K}$ be the $n \times n$ Gram matrix with entries $K_{ij} = c(x_i, x_j)$.

3.3 Prior for ψ

Let $\psi(x_i) = \{\psi_1(x_i), ..., \psi_k(x_i)\}$ where $\psi_h \sim \text{GP}(0, c_\psi)$ for $h = 1, ..., k$, and c_ψ is the squared exponential covariance kernel, i.e, $c_\psi(x, x') = \exp\left(-\kappa_\psi\|x - x'\|_2^2\right)$.

3.4 Prior for Σ_0

The diagonal covariance matrix Σ_0 is specified as $\text{diag}(\sigma_j^{-2})$ with inverse-gamma priors for the individual variance terms $\sigma_j^{-2} \sim \text{Ga}(a_\sigma, b_\sigma)$.

So, our complete model for the data is therefore given by the following expressions:

$$y_{p\times 1} = \Theta_{p\times L}\xi(x)_{L\times k}\eta(x)_{k\times 1} + \epsilon_{p\times 1} \tag{6}$$

The quantities that we are interested are the spatial conditional correlation structure at time x_i captured by the mean $\mu(x_i) = \Theta\xi(x_i)\psi(x_i)$ and the covariance $\Sigma(x_i) = \Theta\xi(x_i)\xi(x_i)^T\Theta_T + \Sigma_0$. As a final remark, while we factored in the complexity of the temporal changes through the use of Gaussian processes, we expected and assumed that k and L are much smaller than p.

4 Gibbs Sampling

For convenience, we rewrote some of the parameters in the following way:

$$\eta_i = \psi(x_i) + \nu_i \tag{7}$$

$$y_{ij} = \sum_{m=1}^{k} \eta_{im} \sum_{l=1}^{L} \theta_{jl}\xi_{lm}(x_i) + \epsilon_{ij} \tag{8}$$

$$\Omega_i = \Theta\xi(x_i) \tag{9}$$

In addition, for $l = 1, ..., L$, $h = 1, ..., p$,$m = 1, ..., k$ denote the following:

$$\xi^{-lm} = \{\xi_{rs}, r \neq l, s \neq m\} \tag{10}$$

$$\psi^{-l} = \{\psi_r, r \neq l\} \tag{11}$$

$$\tau_l^{-h} = \prod_{t=1, t\neq h}^{l} \delta_t \tag{12}$$

$$\left\{\xi_{lm}(x_i)\right\}_{i=1}^{n} = \begin{pmatrix} \xi_{lm}(x_1) \\ \vdots \\ \xi_{lm}(x_n) \end{pmatrix} \tag{13}$$

For fixed hyperparameters $L, k, a_\sigma, b_\sigma, a_1, a_2, \kappa_\psi$ and κ, with some algebraic manipulation detailed in Fox and Dunson, the Gibb sampling algorithm can summarized in the following way:

- Initialize all parameters
- <u>Step 1</u>: Sample ξ_{lm} given $\{y_i\}, \Theta, \xi_{-lm}, \{\eta_i\}, \Sigma_0$

$$\left\{\xi_{lm}(x_i)\right\}_{i=1}^{n} | \{y_i\}, \{\eta_i\}, \xi^{-lm}, \Sigma_0 \sim N_n\left(\tilde{\Sigma_\xi}\left\{\eta_{im}\sum_{j=1}^{p}\theta_{jl}\sigma_j^{-2}\tilde{y_{ij}}\right\}_{i=1}^{n}, \tilde{\Sigma_\xi}\right) \tag{14}$$

where

$$\tilde{y}_{ij} = y_{ij} - \sum_{r \neq l, s \neq m} \theta_{jr} \xi_{rs}(x_i)$$
$$\Sigma_{\xi}^{-1} = K^{-1} + \text{diag}\left(\eta_{1m}^2 \sum_{j=1}^{p} \theta_{jl}^2 \sigma_j^{-2}, ..., \eta_{nm}^2 \sum_{j=1}^{p} \theta_{jl}^2 \sigma_j^{-2} \right)$$

with K being the GP covariance matrix with $K_{ij} = c(x_i, x_j)$ defined as in Section 3.

- Step 2: Sample ψ_l given $\{y_i\}, \{\eta_i\}, \psi_{r \neq l}, \Theta, \psi, \Sigma_0$

$$\left\{ \psi_l(x_i) \right\}_{i=1}^{n} | \{y_i\}, \{\eta_i\}, \psi^{-l}, \Sigma_0, \xi \sim N_n \left(\tilde{\Sigma}_{\psi} \left\{ \Omega_{il}^T \left(\Omega_i \Omega_i^T + \Sigma_0 \right)^{-1} \tilde{y}_i^{-l} \right\}_{i=1}^{n}, \tilde{\Sigma}_{\psi} \right)$$

where,

$$\tilde{y}_i^{-l} = y_i - \sum_{r \neq l} \Omega_{ir} \psi_r(x_i)$$
$$\tilde{\Sigma}_{\psi} = K^{-1} + \text{diag}\left(\Omega_{1l}^T \left(\Omega_i \Omega_i^T + \Sigma_0 \right)^{-1} \Omega_{1l}, ..., \Omega_{nl}^T \left(\Omega_i \Omega_i^T + \Sigma_0 \right)^{-1} \Omega_{nl} \right)$$

- Step 3: Sample ν_i given $\{y_i\}, \psi(x_i), \Theta, \xi(x_i), \Sigma_0$.

$$\nu_i \mid \{y_i\}, \psi(x_i), \Theta, \xi(x_i), \Sigma_0 \sim N_k \left(\{I + \xi(x_i)^T \Theta^T \Sigma_0^{-1} \Theta \xi(x_i)\}^{-1} \xi(x_i)^T \Theta^T \Sigma_0^{-1} (y_i - \Omega \psi(xi_i), \{I + \xi(x_i)^T \Theta^T \Sigma_0^{-1} \Theta \xi(x_i)\}^{-1} \right) \tag{15}$$

Compute η_i,

$$\eta_i = \psi(x_i) + \nu_i \tag{16}$$

- Step 4: Sample σ_j^2 given $\{y_i\}, \{\eta_i\}, \Theta, \psi$. Let $\theta_{j.}$ be the jth row of Θ

$$\sigma_j^{-2} | \{y_i\}, \{\eta_i\}, \Theta, \psi \sim \text{Ga}\left(a_{\sigma} + \frac{n}{2}, b_{\sigma} + \frac{1}{2} \sum_{i=1}^{n} (y_{ij} - \theta_{j.} \xi(x_i) \eta_i)^2 \right) \tag{17}$$

- Step 5: Sample $\theta_{j.}$ given $\{y_i\}, \{\eta_i\}, \psi, \phi, \tau$

$$\theta_{j.} | \{y_i\}, \{\eta_i\}, \psi, \phi, \tau \sim N_L \left(\tilde{\Sigma_{\theta}} \left\{ \xi(x_i) \eta_i \right\}_{i=1}^{n} \sigma_j^{-2} \left\{ y_{ij}^T \right\}_{i=1}^{n}, \tilde{\Sigma_{\theta}} \right) \tag{18}$$

where

$$\tilde{\Sigma}_{\theta}^{-1} = \sigma_j^{-2} \{\xi(x_i) \eta_i\}_{i=1}^{n} \{\eta_i^T \xi^T(x_i)\}_{i=1}^{n} + \text{diag}(\phi_{j1} \tau_1, ..., \phi_{jL} \tau_L) \tag{19}$$

- Step 6: Sample the hyperparameters $\phi_{jl}, \delta_1, \delta_h$

1. ϕ_{jl} given θ_{jl}, τ_l

$$\phi_{jl} | \theta_{jl}, \tau_l \sim \text{Ga}\left(2, \frac{\gamma + \tau_l \theta_{jl}^2}{2} \right) \tag{20}$$

2. δ_l given $\Theta, \tau_{r \neq 1}$

$$\delta_1 | \Theta, \tau^{-1} \sim \text{Ga}\left(a_1 + \frac{pL}{2}, 1 + \frac{1}{2} \sum_{l=1}^{L} \tau_l^{-1} \sum_{j=1}^{p} \phi_{jl} \theta_{jl}^2 \right) \tag{21}$$

3. δ_h given $\Theta, \tau_{r\neq h}$

$$\delta_h|\Theta, \tau^{-h} \sim \text{Ga}\left(a_2 + \frac{p(L-h+1)}{2}, 1 + \frac{1}{2}\sum_{l1}^{L}\tau_l^{-h}\sum_{j=1}^{p}\phi_{jl}\theta_{jl}^2\right) \tag{22}$$

Compute τ_l,

$$\tau_l = \prod_{h=1}^{l}\delta_h \tag{23}$$

A summary of how our parameters are related to each other is provided in Figure 3. As a remark for the hyperparameters κ_ψ and κ, though there exists heuristic data-driven approach to estimate κ, referred to as length-scale parameter in Fox and Dunson, our model turned out to be relatively robust for the choice of κ (Section 6). Also, for simplicity, we set $\kappa = \kappa_\psi$.

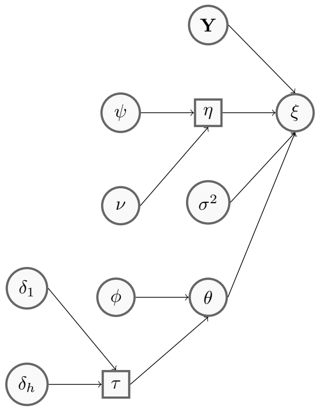

Figure 3: Dependencies of the parameters

5 Implementation & Result

Our dynamic latent factor model allows factors as well as the loadings to vary as functions of the predictor variable, which are discrete days in our case. This allows temporal changes to be implicitly modeled through the proposed framework, allowing for continuous variations in the spatial marginal correlation structure. Thus, we get a separate correlation structure between the regions in our analysis for each day.

Let's also reiterate that our model does not explicltly encode any spatial structure between the regions, motivated by the fact that our exploratory analysis presented that geographically distant regions could be highlight correlated.

In Figure 4 we have correlation map snapshots that examine correlations utilizing all 51 states jointly and the entire time course of approximately 90 days. These maps present the specific time-varying geographic structure of the inferred correlations, clearly showing temporal changes in the spatial correlations of the COVID-19 dataset. This is even in the presence of substantial missing information as on January 21, 2020 the first case in the United States was reported in Washington, so the presented correlation structure was inferred before any data was available for the states listed. This is also the case with Ohio, Nebraska and Georgia on February 15, 2020 as the first cases reported in each of these states occurred after this date.

This is enabled by the use of a weighted combination of basis elements to model the factor loadings, where a row Θ corresponding to Nebraska, for example, is informed by all of Nebraska's available data while the

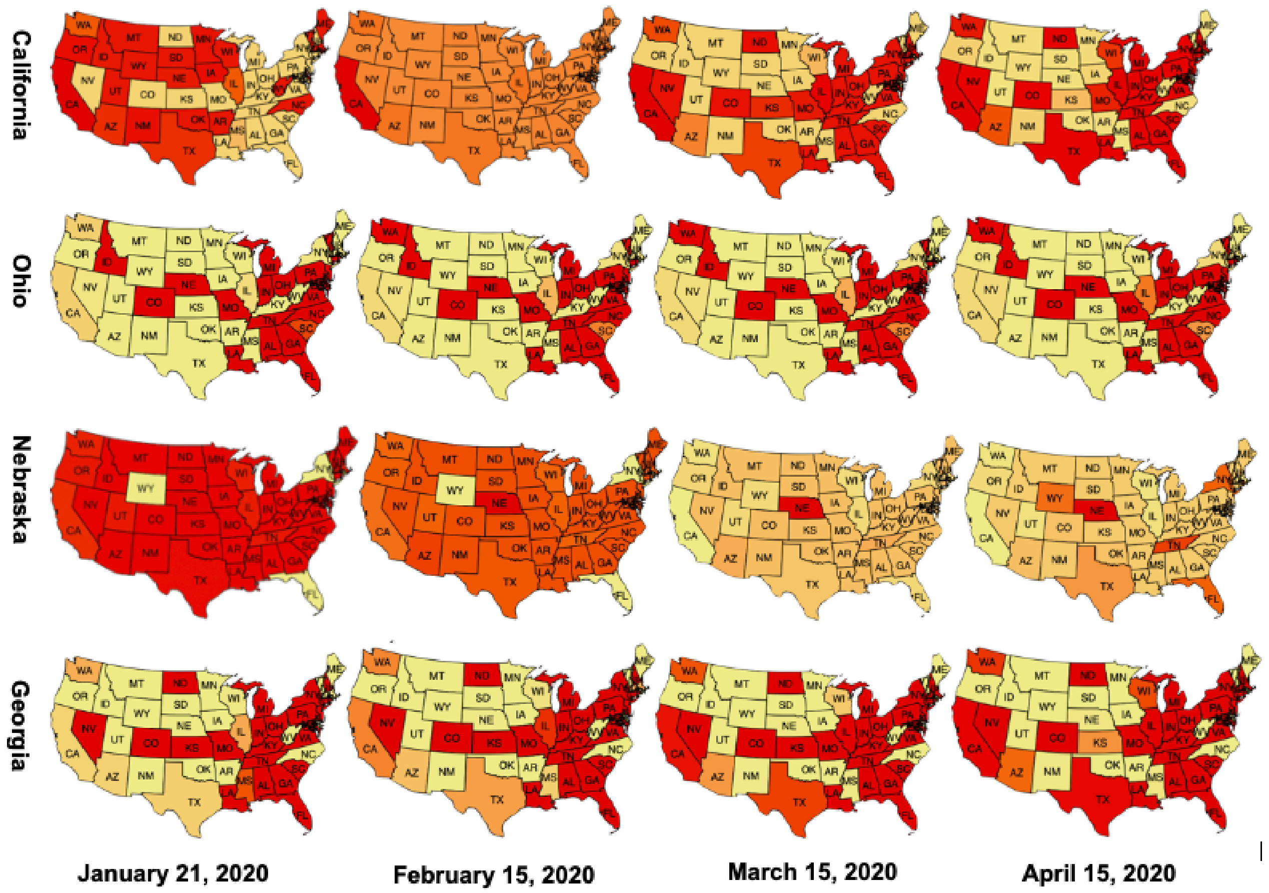

Figure 4: Correlation maps based on the posterior mean estimate of $\Sigma(x)$.

latent Gaussian Process basis elements ξ_{lk} are informed by all of the other regions' data in addition to assumed continuity of ξ_{lk} which shares information across time.

For all four of these states on January 21, 2020 we see a strong local spatial correlation structure with neighboring states.

Focusing on California, we see that the spatial correlation structure is centered on the western quadrant of the US, initially. Interestingly, we see the correlation structure for California shift over time to the Eastern Quadrant. Aside from the availability of data that was previously missing, we surmise that this shift can be attributed to traveller frequency from larger cities in California to the larger cities on the Eastern side, such as Chicago, Atlanta, Miami, Detroit and Dallas since the onset of COVID-19 is substantially more in urban environments than rural areas. We also see more substantial correlations with neighboring states Nevada and Arizona as more data is presented. The southern part of California has substantially more cases than the northern side, and we guess that the proximity of these cases to Nevada and Arizona is attributable to the inferred correlation.

For Ohio, we don't see substantial shifts over time in the correlation structure, the notable ones being an increase in correlation to Washington and Illinois. Among all the states presented in the diagram, Ohio had the latest date of the first reported case, March 9th, and so substantially less data was available compared to California whose first case was January 25th. The lack of correlation between Kentucky and Ohio is also fascinating as Ohio shows positive correlation with the other neighboring states and Kentucky's first case is just 3 days before Ohio's first case. Kentucky's rate of new cases is far more eradic than Ohio's, i.e. days with sharp drops in new cases followed by days with sharp increases. Thus, we see how the rate of testing could be attributable to the lack of correlation between these two states.

Nebraska shows a substantial spatial correlation with the rest of the country before the first case is reported on February 17th. We see from January to March, the initial spatial correlation diminishing over time as we collect more data on neighboring states. Wyoming's correlation with Nebraska is especially interesting as initially a negative correlation is surmised. However, over time, we see that these two states get more and more correlated. By April, we see that Nebraska shows its most correlation with the highly populated states of Texas, Florida, and New York along with Tennessee and Wyoming. We note that neighboring states Kansas, Missouri, Iowa and Colorado report their first case nearly 2 weeks after Nebraska which we surmise expresses the lack of correlation between these neighboring states.

Georgia initially shows little correlation with states like Washington, California, Arizona and Texas but with time we see that these correlations are more substantial. In Georgia, the confirmed COVID-19 cases are densely populated in the Atlanta region which is very well connected with urban cities like San Francisco, Phoenix, Dallas, Los Angeles and Seattle. Thus, despite the fact that Georgia's first reported case occurred on March 2nd compared to Washington (January 21st) and California (January 25th) we think the traveller frequency between these states is attributable to the inferred correlation structure.

There are definitely geographic cases that we don't understand like the correlation between North Dakota and California. That being said, we think our model has detected attributable correlation structures across time.

6 Sensitivity Analysis, Convergence and Mixing Diagnostics

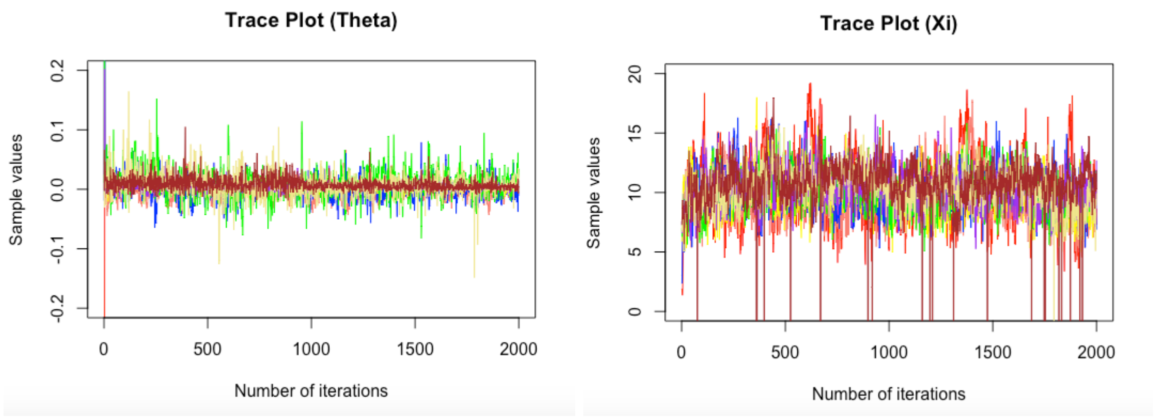

Figure 5: Trace plots showing a fair amount of mixing and convergence for two significant variables in our calculation of $\Sigma(x)$. Each color corresponds to a different truncation level, L, and the values plotted are from the diagonal elements of $\xi(x)_{L\times k}$ and $\Theta_{p\times L}$ in each iteration.

The hyper-parameters were initially set with $L = 8$ and $k = 12$ and the Gaussian process length-scale hyper-parameter set to $\kappa = \kappa_\psi = 10$ to account for the time-scale (days) at which the number of cases were changing in our dataset. We assessed convergence and mixing by plotting trace plots for $\xi(x)$ and Θ, the main terms determining the time-varying spatial correlation in our model. We see in 5, that both of these parameters show a good level of mixing and convergence with oscillating trace plots after 2,000 Gibbs iterations. We also performed hyper-parameter sensitivity, letting $\kappa_\psi = \kappa = 20$ to induce less temporal correlation and using a larger truncation level of $L = k = 15$ with less stringent shrinkage hyper-parameters $a_1 = a_2 = 0.75$ (instead of $a_1 = a_2 = 2$). The results of mixing and convergence were identical.

7 Limitations and Future Work

One of the limitations of our implementation is computational efficiency. The R code we wrote took between 55 and 65 minutes to run 2,000 iterations, depending on whether or not we chose to calculate the posterior mean of $\Sigma(x)$ for each iteration. The functions we used in the Gibbs sampler are provided in the appendix. As a result, it was unfeasible to perform a OAT (one-at-a-time) sensitivity analysis on our model. It was also unfeasible to run more than 2,000 Gibbs iterations due to time constraints, though our trace plots highlight a fair bit of mixing and convergence within the 2,000 iterations.

As another limitation, for simplicity, we used multivariate Gaussian distribution to model count data, which follows a discrete, strictly non-negative distribution. One of the disadvantages of this approach is that the response variable may be negative. Though we were mainly interested in the mean and covariance as a function of time, and therefore having some predicted negative values does not affect our parameters of interest, a potential extension to the framework proposed is to replace the multivariate Gaussian with the equivalent multivariate Poisson distribution. Extending this framework to work with generalized linear models would make it significantly more powerful and applicable to several other settings.

8 Author Contributions

The authors contributed equally to conceptualizing the project. Loc Cao performed the exploratory data analysis. Varshant Dhar wrote and executed the codes for the Gibbs sampler and generated all the figures for results. Visweswaran Ravikumar helped with improving the code and performing sensitivity analysis.

9 Appendix

Functions from the Gibbs sampler:

```
# (Step 1) Sample Xi in our Gibbs sample notation
sample_zeta <- function(y, theta, eta, invSig_vec,zeta,invK,inds_y){
  # We derive the sequential sampling of the zeta_{ll,kk} by reformulating
  # the full regression problem as one that is solely in terms of
  # zeta_{ll,kk} having conditioned on the other latent GP functions:

  # y_i = eta(i,m)*theta(:,ll)*zeta_{ll,kk}(x_i) + tilde(eps)_i,

  # Initialize the structure for holding the conditional mean for additive
  # Gaussian noise term tilde(eps)_i and add values based on previous zeta:

  p = dim(theta)[1]
  L = dim(theta)[2]
  k = dim(eta)[1]
  N = dim(eta)[2]

  mu_tot = matrix(0, nrow=p,ncol=N)
  error_nn = matrix(0, nrow=L,ncol=N)
  for (nn in c(1:N)) {
    mu_tot[,nn] = theta%*%zeta[, ,nn] %*% eta[,nn]
    error_nn[,nn] = t(theta^2) %*% (as.matrix(invSig_vec) * as.matrix(1-inds_y[,nn]))
  }
  for (ll in c(1:L)) {
    theta_ll = theta[,ll]
    for (kk in randperm(k,k)) {
      eta_kk = eta[kk,]
      zeta_ll_kk = t(drop(zeta[ll,kk,]))
      mu_tot = mu_tot - matrix(theta_ll,nrow=p,ncol=N,byrow = FALSE) * matrix(eta_kk,nrow=
          ↪ p,ncol=N,byrow = TRUE) * matrix(zeta_ll_kk,nrow=p,ncol=N,byrow = TRUE)

      # Using standard Gaussian identities, form posterior of
      # zeta_{ll,kk} using information form of Gaussian prior and likelihood:
      A_lk_invSig_A_lk = (eta[kk,]^2)*(t(theta[,ll]^2) %*% as.matrix(invSig_vec) - error_
          ↪ nn[ll,])
      theta_tmp = t(theta[,ll]) * t(invSig_vec)
      ytilde = (y - mu_tot)*inds_y # normalize data by subtracting mean of tilde(eps)
      theta_lk = as.matrix(eta[kk,]) * t(theta_tmp %*% ytilde)
      # Transform information parameters
      cholSig_lk_trans = mldivide(chol(invK + diag(A_lk_invSig_A_lk,nrow = N)), diag(nrow=
          ↪ N))
      m_lk = cholSig_lk_trans%*%(t(cholSig_lk_trans) %*% theta_lk)

      # Sample zeta_{ll,kk} from posterior Gaussian
      zeta[ll,kk,] = m_lk + cholSig_lk_trans%*%as.matrix(rnorm(N,1))
      zeta_ll_kk = t(drop(zeta[ll,kk,]))
      mu_tot = mu_tot + matrix(theta_ll,nrow=p,ncol=N,byrow=FALSE) * matrix(eta_kk,nrow=p,
          ↪ ncol=N,byrow=TRUE) * matrix(zeta_ll_kk,nrow=p,ncol=N,byrow=TRUE)
    }
  }
  return(zeta)
}

# (Step 2) Psi in our Gibbs Sample notation
sample_psi_margxi <- function(y,theta,invSig_vec,zeta,psi,invK,inds_y,iter){
  # When modeling a non-zero latent mean \mu(x), sample the latent mean
  # GPs \psi(x) marginalizing \xi.
  p = dim(theta)[1]
  L = dim(theta)[2]
  k = dim(psi)[1]
  N = dim(psi)[2]
  Sigma_0 = diag(1/invSig_vec)

  mu_tot = matrix(0,nrow = p, ncol = N)
  Omega = array(rep(0,times = p*k*N), c(p,k,N))
  OmegaInvOmegaOmegaSigma0 = array(rep(0,times = k*p*N), c(k,p,N))
  for (nn in 1:N) {
    Omega[inds_y[,nn], ,nn] = theta[inds_y[,nn],]%*%zeta[, ,nn]
    if(dim(as.matrix(Omega[inds_y[,nn], ,nn]))[2]==1){
      temp = mldivide((t(Omega[inds_y[,nn], ,nn]) %*% as.matrix(Omega[inds_y[,nn], ,nn]) +
          ↪  Sigma_0[inds_y[,nn],inds_y[,nn]]) , diag(sum(inds_y[,nn])))
      OmegaInvOmegaOmegaSigma0[,inds_y[,nn],nn] = as.matrix(Omega[inds_y[,nn], ,nn]) %*%
          ↪ temp
    } else {
      temp = mldivide((as.matrix(Omega[inds_y[,nn], ,nn]) %*% t(Omega[inds_y[,nn], ,nn]) +
          ↪  Sigma_0[inds_y[,nn],inds_y[,nn]]), diag(sum(inds_y[,nn])))
      OmegaInvOmegaOmegaSigma0[,inds_y[,nn],nn] = t(Omega[inds_y[,nn], ,nn])%*%temp
    }
    mu_tot[,nn] = Omega[, ,nn]%*%psi[,nn]
  }
  if (sum(sum(mu_tot))==0){ # if this is a call to initialize psi
    numToIters = 50
  } else {
    numToIters = 5
  }
  for (numIter in c(1:numToIters)) {
    for (kk in randperm(k,k)) { # create random ordering for kk in sampling zeta_{ll,kk}
      Omega_kk = drop(Omega[,kk,])
      psi_kk = psi[kk,]
      mu_tot = mu_tot - Omega_kk*matrix(psi_kk,nrow=p,ncol=N,byrow=TRUE)

      theta_k = as.matrix(diag(t(drop(OmegaInvOmegaOmegaSigma0[kk, ,]))%*%(y-mu_tot)))
      Ak_invSig_Ak = diag(t(drop(OmegaInvOmegaOmegaSigma0[kk, ,]))%*%Omega_kk)
      cholSig_k_trans = mldivide(chol(invK + diag(Ak_invSig_Ak)),diag(nrow=N))

      # Transform information parameters
      m_k = cholSig_k_trans%*%(t(cholSig_k_trans)%*%theta_k)

      # Sample zeta_{ll,kk} from posterior Gaussian
      psi[kk,] = m_k+cholSig_k_trans%*%matrix(rnorm(N),nrow=N,ncol=1)

      psi_kk = psi[kk,]
      mu_tot = mu_tot + Omega_kk*matrix(psi_kk,nrow=p,ncol=N,byrow=TRUE)
    }
  }
  return(psi)
}

# (Step 3) Sample Nu in our Gibbs Sample notation
sample_xi <- function(y,theta,invSig_vec,zeta,psi,inds_y){
  # Sample latent factors eta_i using standard Gaussian identities based on
  # the fact that:

  # y_i = (theta*zeta)*eta_i + eps_i, eps_i \sim N(0,Sigma_0),
  # eta_i \sim N(0,I)

  # and using the information form of the Gaussian likelihood and prior.
  # eta = psi + xi
  p = dim(y)[1]
  N = dim(y)[2]
  L = dim(zeta)[1]
  k = dim(zeta)[2]

  xi = matrix(0, nrow=k, ncol=N)
  for (nn in c(1:N)) {
    theta_zeta_n = theta%*%zeta[, ,nn]
    y_tilde_n = y[,nn]-theta_zeta_n%*%psi[,nn]
    invSigMat = invSig_vec*t(inds_y[,nn])
    invSigMat = matrix(invSigMat, nrow=k,ncol=p,byrow=TRUE)
    zeta_theta_invSig = t(theta_zeta_n)*invSigMat
    cholSig_xi_n_trans = mldivide(chol(diag(nrow=k) + zeta_theta_invSig%*%theta_zeta_n),
        ↪ diag(nrow=k))
    m_xi_n = cholSig_xi_n_trans%*%(t(cholSig_xi_n_trans)%*%(zeta_theta_invSig%*%y_tilde_n
        ↪ ))
    xi[,nn] = m_xi_n + cholSig_xi_n_trans%*%as.matrix(rnorm(k,1))
  }
  return(xi)
}

# (Step 4) sigma^2 in our Gibbs sampling notation
sample_sig <- function(y,theta, eta, zeta,prior_params,inds_y){
  # Sample latent factor model additive Gaussian noise covariance
  # (diagonal matrix) given the data, weightings matrix, latent GP
  # functions, and latent factors:
  p = dim(y)[1]
  N = dim(y)[2]
  a_sig = prior_params["sig.a_sig"]
  b_sig = prior_params["sig.b_sig"]

  inds_vec = seq(1,N)
  invSig_vec = seq(1,p)
  for (pp in seq(1,p)) {
    sq_err = 0
    for (nn in inds_vec[inds_y[pp,]]) {
      sq_err = sq_err + (y[pp,nn] - theta[pp,]%*%zeta[ , ,nn]%*%eta[,nn])^2
    }
    invSig_vec[pp] = rgamma(n=1, shape = (a_sig + 0.5*rowSums(inds_y)[pp]),scale=1)/(b_
        ↪ sig + 0.5*sq_err)
  }
  return(invSig_vec)
}


# (Step 5) Theta in Gibbs Sample notation
sample_theta <- function(y,eta,invSig_vec,zeta,phi,tau,inds_y){
  p = dim(y)[1]
  N = dim(y)[2]
  L = dim(zeta)[1]
  theta = matrix(0, nrow = p, ncol = L)

  eta_tilde = matrix(0, nrow = L, ncol = N)
  for (nn in c(1:N)) {
    eta_tilde[,nn] = zeta[, ,nn]%*%eta[,nn]
  }
  eta_tilde = t(eta_tilde)

  for (pp in c(1:p)) {
    inds_y_p = t(inds_y[pp,])
    eta_tilde_p = eta_tilde*matrix(inds_y_p, nrow=N, ncol=L,byrow = FALSE)
    chol_Sig_theta_p_trans = mldivide(chol(diag(phi[pp,]*tau,nrow=L)+ invSig_vec[pp]*(t(
        ↪ eta_tilde_p)%*%eta_tilde_p)),diag(nrow=L))
    m_theta_p = invSig_vec[pp]*(t(chol_Sig_theta_p_trans)%*%chol_Sig_theta_p_trans)%*%(t(
        ↪ eta_tilde_p)%*%y[pp,])
    theta[pp,] = m_theta_p + chol_Sig_theta_p_trans%*%as.matrix(rnorm(L,1))
  }
  return(theta)
}


# (Step 6) Phi and Tau in Gibbs Sample notation
sample_hypers <- function(theta, phi, tau, prior_params){
  # Sample weightings matrix hyperparameters given weightings matrix:,
  p = dim(theta)[1]
  L = dim(theta)[2]

  a1 = prior_params["hypers.a1"]
  a2 = prior_params["hypers.a2"]
  a_phi = prior_params["hypers.a_phi"]
  b_phi = prior_params["hypers.b_phi"]
  a = c(a1,rep(1,times=L-1))
  delta = c(exp(log(tau[1])),exp(diff(log(tau))))
  for (numIter in c(1:50)) {
    phi = matrix(rgamma(n=p*L,shape=(a_phi + 0.5),scale=1),nrow=p,ncol=L) / (b_phi + 0.5*
        ↪ matrix(tau,nrow=p,ncol=L,byrow = TRUE)*(theta^2))
    sum_phi_theta = colSums(phi*(theta^2))
    for (hh in c(1:L)) {
      tau_hh = exp(cumsum(log(delta)))*c(rep(0,times=hh-1),rep(1,times=L-hh+1)/delta[hh])
      delta[hh] = rgamma(n=1,shape=a[hh] + 0.5*p*(L-hh+1),scale = 1)/(1+0.5*sum(tau_hh*sum
          ↪ _phi_theta))
    }
    tau = exp(cumsum(log(delta)))
  }
  phi_tau <- list("phi"=phi, "tau"=tau)
  return(phi_tau)
}
```