Supplementary Information for Mediation Analysis of Partisanship and Policies on COVID-19 Infections via Bayesian Latent Variable Modeling

August 25, 2021

Contents

1 Complete Model Definition

Given this overview of the intuition behind our approach, we turn to a more formal definition. Our observed outcomes are the cumulative total of tests and cases reported on a given day t in a given state c. We assume that the unobserved state-specific cumulative infection rate I_{ct} can be modeled as a time-varying Beta-distributed random variable with a mean parameter $\mu \in (0, 1)$ and shape parameter $\phi > 0$. We also assume that the over-time change in the disease can be modeled as a 3-order polynomial time trend that is a function of the number of post-outbreak time periods $T_O < T$, where an outbreak begins at the first reported case in a given area.

It is important to note that the reason we employ a cubic function is based on theoretical considerations. In our model, the polynomial represents the rate of infection increase in the absence of any other covariates, or equivalently the *counterfactual* rate of infections. We know from the SIR/SEIR simulations that, in the absence of any countervailing measures, epidemics occur in ever-increasing waves until the herd immunity threshold is reached, although the curve is unlikely to be symmetric as a simpler quadratic function would require. As such, we employ this function because it represents a credible baseline for what the epidemic would do if no other factors impeded its spread. We further allow the polynomial trends to vary by states hierarchically, i.e., the information about the trends is partially pooled across states. The partial pooling is necessary to allow for state-level heterogeneity in the exact progress of the disease, which we note is further modeled by including data on when cases first were reported in each state. We believe this counterfactual infection function is considerably flexible while also motivated by our understanding of how epidemics progress.

We define the conditional distribution of the unobserved infection rate I_{ct} as:

$$\Pr(I_{ct} \mid t = T) \sim \text{Beta}(\mu\phi, (1-\mu)\phi) \tag{1}$$

$$\mu = g^{-1}(\alpha_1 + \beta_{O1} \sum_{c=1}^{C} \sum_{t=1}^{T-14} a_{ct} + \tag{2}$$

$$\beta_{I1} t_o + \beta_{I2} t_o^2 + \beta_{I3} t_o^3 + \beta_C X_{ct}) \tag{3}$$

This parameterization of the Beta distribution in terms of μ and ϕ follows from the Beta regression literature[1] so that we can model the expected value $E[I_{ct}]$ directly via μ. As such, we use $g^{-1}(\cdot)$, the inverse logit function, to scale the linear model in μ to the $(0, 1)$ interval. For the parameters, $\beta_{O1} \sum_{c=1}^{C} \sum_{t=1}^{t-14} a_{ct}$ are the sum of observed cases in the country with a 14-day lag, which represents the possibility of cross-border spread in infections. The three β_{Ii} are polynomial coefficients of the number of post-outbreak time periods t_o.

The parameter vector β_C represents the effect of independent covariate matrix X_{ct} on the latent infection rate. These are our main variables of interest, and have effects in addition to the polynomial time trends. Finally, the parameter ϕ becomes a dispersion parameter governing the variability of latent infection rate.

Because we do not have measures of I_{ct}, we need to used the observed data, tests q_{ct} and cases a_{ct}, to infer I_{ct}. First, we propose that the number of infections will almost certainly increase the number of tests as states try to stop the disease's spread via surveillance. Second, we can assume that a rising infection rate

is associated with a higher ratio of positive results (reported cases) conditional on the number of tests, that is, COVID-19 is causing positive test results. We model both of these observed indicators, tests and cases, jointly to simultaneously adjust for the infection rate's influence on both factors. It is this joint modeling that permits us to directly incorporate testing bias. In fact, our model learns about the infection rate from the level of tests rather than trying to discard this information.

To model the number of tests, we assume that each state has an unobserved level of testing capacity, which increases at a non-linear rate during the course of the epidemic. We employ a quadratic function of testing capacity to express the concept of diminishing marginal returns. States were able to ramp up testing once PCR tests were approved by the FDA, but faced constraints due to shortages of supplies, personnel and labs. The cumulative number of observed tests q_{ct} for a given time point t and state c and as a fraction of the states' population, c_p, then has a binomial distribution:

$$q_{ct} \sim \text{Binomial}(c_p, g^{-1}(\alpha_2 + \beta_b I_{ct} + \beta_{cq1} L_t + \beta_{cq2} L_t^2)). \tag{4}$$

The parameters β_{cq1} and β_{cq2} represent the quadratic increase in testing capacity that varies by state c. We similarly allow for partial pooling of these coefficients as testing capacity will show a limited level of variability across states. The parameter β_b then represents the independent contribution of the level of infections I_{ct} on the total number of requests demanded marginal of testing capacity. The intercept α_2 indicates how many tests would be performed in a state with an infection rate of zero and at time $t = 0$, and as such is likely to be very low.

The binomial model for the number of observed tests q_{ct} provides some information about I_{ct}, but not enough for useful estimates. We can learn much more about I_{ct} by also modeling the number of observed cases a_{ct} as another binomial random variable expressed as a proportion of the state population, c_p:

$$a_{ct} \sim \text{Binomial}(c_p, g^{-1}(\alpha_3 + \beta_a I_{ct})), \tag{5}$$

where $g^{-1}(\cdot)$ is again the inverse logit function, α_3 is an intercept that indicates how many cases would test positive with an infection rate of zero (approximately equal to the false positive rate of the test), and β_a is a parameter that determines how hard it is to find the infected people and test them as opposed to people who are not actually infected. The multiplication of this parameter and the infection rate determines the cumulative number of cases, a_{ct}, as a proportion of the state population, c_p.

To summarize the model, infection rates determine how many tests a state is likely to undertake and also

the number of positive tests they receive as cases. This simultaneous adjustment helps takes care of misinterpreting the observed data by not taking into account varying testing rates, which has made it hard to generalize findings concerning the disease and also led some policy makers to claim that rising case rates are solely due to increasing numbers of tests. It also allows us to learn the likely location of the infection rate conditional on what we observe in terms of tests and cases.

Because sampling from a model with a hierarchical Beta parameter can be difficult, we simplify the likelihood by combining the beta distribution and the binomial counts into a beta-binomial model for tests:

$$q_{ct} \sim \text{Beta-Binomial}(c_p, \mu_q\phi_q, (1-\mu_q)\phi_q) \tag{6}$$

$$\mu_q = g^{-1}(\alpha_2 + \beta_b I_{ct} + \beta_{cq1} L_t + \beta_{cq2} L_t^2) \tag{7}$$

and cases:

$$a_{ct} \sim \text{Beta-Binomial}(q_{ct}, \mu_a\phi_a, (1-\mu_a)\phi_a) \tag{8}$$

$$\mu_a = g^{-1}(\alpha_3 + \beta_a I_{ct}). \tag{9}$$

where I_{ct} is now equal to the linear model vector μ shown in (3) and mapped to $(0, 1)$ via the inverse logit function.

1.1 Identifiability

This model contains an unobserved latent process I_{ct}, and as such the model as presented is not identified from the data alone without further information. For example, the parameters that control the influence of the infection rate on tests and cases could increase and the latent infection rate could decrease without the probability of the observed data changing.

There two further steps taken to identify this model which we believe represent very limited additional assumptions, especially compared to existing modeling approaches. First, we must require that I_{ct} is a non-decreasing quantity. The number of infected people cannot decrease in an epidemic, but the model as

expressed does not require that to be true.[1] We can eliminate that possibility from the model by imposing an ordered constraint on I_{ct}:

$$I_{ct} = \begin{cases} I_{ct} & \text{if } t = 1 \\ I_{ct-1} + e^{I_{ct}} & \text{if } 1 < t < T \end{cases} \tag{10}$$

This transformation forces I_{ct} to be no less than I_{ct-1}. At the same time, we do not need to impose any constraints on the covariates themselves, allowing us to sample those in an unconstrained space before we transform I_{ct}.

However, we also need some information about the empirical scale of testing bias to produce identified estimates of I_{ct}. We could do so by adding a prior to the model about the plausible range of total infections to reported cases, though we prefer to use information that is more precise. The Centers for Disease Control's serology surveys conducted during the pandemic represent an empirical way of relating I_{ct} to plausible estimates of infections at varying time points. By including this information, we also implicitly account for many of the variables explicitly parameterized in compartmental models such as reporting delays. Because we have an estimate of the number infected at time t that is independent of reported cases and tests, the model will find the parameter estimates that are most likely given the observed differences between the surveys and the reported data.

Because we model the infection rate as a cumulative count, it is straightforward to include this information in the model. For a given state c and time point t for which we have survey information, we model the count of infected S^P_{ct} as a proportion of the total subjects in each serology survey S^N_{ct} with the Binomial distribution:

$$S_{ct} \sim \text{Binomial}(S^N_{ct}, g^{-1}(I_{ct})) \tag{11}$$

It is important that the serology surveys enter the model in this fashion so that we can model the survey count stochastically. This is necessary to propagate uncertainty in the sample size through to our estimates of I_{ct}. This uncertainty matters as well because the serology surveys exhibit random noise and do not always increase over time, as can be seen in Table 1. By modeling the relationship as a probabilistic one, we are making the weaker assumption that the infected rate is probably close to the serology estimate, but the two do not need to be the identical. The combined posterior estimates for I_{ct} will then be weighted with the

[1] Note that we are not assuming that people cannot be re-infected, but rather that people cannot become uninfected after contracting the disease.

Table 1: Geographic Serological Surveys from the Centers for Disease Control

State	% Infected	N	Date Started	Date Ended
Connecticut	4.9%	1431	2020-04-26	2020-05-03
Connecticut	5.2%	1800	2020-05-21	2020-05-26
Connecticut	6.3%	1798	2020-06-15	2020-06-17
Connecticut	5.2%	1802	2020-07-03	2020-07-06
Louisiana	5.8%	1184	2020-04-01	2020-04-08
Louisiana	6.7%	770	2020-04-22	2020-04-27
Minnesota	2.4%	860	2020-04-30	2020-05-12
Minnesota	2.2%	1323	2020-05-25	2020-06-06
Minnesota	4.3%	1667	2020-06-15	2020-06-27
Missouri	2.6%	1882	2020-04-20	2020-04-26
Missouri	2.8%	1831	2020-05-25	2020-05-30
Missouri	0.8%	1850	2020-06-15	2020-06-20
Missouri	1.4%	1914	2020-07-05	2020-07-09
New York	3.69%	2482	2020-03-23	2020-04-01
New York	13.2%	1618	2020-04-06	2020-04-16
New York	16.49%	1116	2020-04-27	2020-05-06
New York	14.44%	1581	2020-06-15	2020-06-21
New York	13.12%	1602	2020-07-07	2020-07-11
Pennsylvania	1.98%	824	2020-04-13	2020-04-25
Pennsylvania	2.48%	1743	2020-05-26	2020-05-30
Pennsylvania	2.64%	1694	2020-06-14	2020-06-20
Pennsylvania	3.55%	1751	2020-07-06	2020-07-11
Florida	0.75%	1742	2020-04-06	2020-04-10
Florida	1.3%	1280	2020-04-20	2020-04-27
Florida	2.12%	1790	2020-05-19	2020-05-27
California	0.96%	1224	2020-04-23	2020-04-27
California	0.94%	1539	2020-05-19	2020-05-27
Utah	2.2%	1132	2020-04-20	2020-05-03
Utah	1.1%	1940	2020-05-25	2020-06-05
Utah	1.5%	1976	2020-06-15	2020-06-27
Washington	0.67%	3264	2020-03-23	2020-04-01
Washington	2.18%	1719	2020-04-27	2020-05-11
Washington	2.19%	1803	2020-06-15	2020-06-20
Washington	1.77%	1797	2020-07-06	2020-07-07

case and test likelihoods to produce the most credible estimate of I_{ct}.

As we show in the supplemental information with simulations, no other identification restrictions are necessary to estimate the model beyond weakly informative priors assigned to parameters.

These are:

$$\beta_a \sim \text{Normal}(30, 10), \tag{12}$$

$$\beta_{qci} \sim \text{Normal}(\mu_{qi}, \sigma_{qi}), \tag{13}$$

$$\sigma_{qi} \sim \text{Exponential}(1), \tag{14}$$

$$\mu_{qi} \sim \text{Normal}(0, 20), \tag{15}$$

$$\beta_C \sim \text{Normal}(0, 5), \tag{16}$$

$$\beta_{Ii} \sim \text{Normal}(\mu_{Ii}, \sigma_{Ii}), \tag{17}$$

$$\mu_{Ii} \sim \text{Normal}(0, 10), \tag{18}$$

$$\sigma_{Ii} \sim \text{Exponential}(1), \tag{19}$$

$$\alpha_1 \sim \text{Normal}(0, 10), \tag{20}$$

$$\alpha_2 \sim \text{Normal}(0, 10), \tag{21}$$

$$\alpha_3 \sim \text{Normal}(0, 10) \tag{22}$$

where the normal distribution is parameterized in terms of mean and standard deviation.

The priors to note are the hierarchical regularizing prior put on the varying testing adjustment parameters β_{qci} and varying polynomial trends β_{Ii} with shared means and standard deviations. This partial pooling permits a reasonable degree of heterogeneity in the parameters while still constraining overall dispersion. Relatively informative priors are put on the hierarchical variance parameters σ_{qi} and σ_{Ii} to suggest that while state heterogeneity does exist, we do still expect the states' estimates to be within a given range.

We note that a crucial advantage of this framework is providing a way to measured the count of infected adjusting for known biases in the number of tests. By comparing numbers of tests per capita and growth rates in cases across regions, the model is able to backwards infer a likely number of infected individuals in a given area. As such it exploits both within-area and between-area variance to adjust for the biases of imperfect testing. The wide variety of covariates we add to the model, which we describe in the next section, provide the mechanism through which the model can infer test/case relationships even in states which have not had a CDC serology survey.

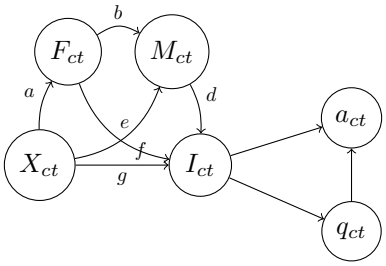
We also extend this model in order to analyze the mediation of a subset of covariates X'_{ct} by adding mediators M_{ct} for mobility and F_{ct} for fear of the disease to the causal diagram, as in Figure 2. Figure 2 has several paths due to the fact that the influence of covariates X_{ct} affects the two mediators differently. Given that

beliefs and preferences precede actions, the covariates X'_{ct} first influence I_{ct} along the ae and abd path through perceptions of how dangerous the disease is. These beliefs both affect the chance of an individual getting infected and thus I_{ct} directly on the path ae, such as by causing an individual to adopt social distancing behaviors, and also on an indirect path abd by which an increase in a people's fear of the disease reduces mobility as people prefer to stay home.

In addition to pathways through the fear mediator F_{ct}, a covariate could influence infections along the pathway through mobility ed without increasing or decreasing fear. This situation could arise if government policies forced people to stay at home against their will and despite their unconcern about the disease. Finally, a covariate could have an unmediated direct effect g on the infection rate. The total effect of a covariate X_{ct} on the spread of the disease is then the sum of all the paths, $abd + af + ed + g$. To calculate the indirect effects and direct effects given the use of the inverse logit function $g^{-1}(\cdot)$, we employ the chain rule as in[2] to calculate the marginal effect of covariates with respect to different pathways to I_{ct}.

Adding the mediators to the model is relatively simple as they do not have link functions and can be included as Normal distributions (i.e., OLS regression) as in.[3] It should be noted that there are in fact five mobility covariates as explained in the following section, and so we explicitly model the covariance in mobility via a multivariate Normal distribution with a covariance matrix parameter Σ_m.

Figure 1: Directed Acyclic Graph for Latent Infection Rate with Mediators



This figure adds mediators M_{ct} (mobility data) and F_{ct} (fear of COVID-19) that mediate the relationship between state-level covariates X'_{ct} and the latent infection rate I_{ct}. Because beliefs precede actions, F_{ct} is causally prior to M_{ct} and can affect infections both via reducing mobility (path abd) and directly apart from mobility (path ae), such as by encouraging individuals to remain socially distant.

To add our mediation covariates M_{ct} and F_{ct}, which we describe in more detail in the next section, we multiply the following likelihoods with the joint posterior:

$$M_{ct} \sim MVN(\alpha_m + \beta_m X'_{ct}, \Sigma_m) \tag{23}$$

$$F_{ct} \sim N(\alpha_f + \beta_f X'_{ct}, \sigma_f) \tag{24}$$

We also include all of M_{ct} and F_{ct} as linear predictors in (3).

We fit this model using Markov Chain Monte Carlo in the Stan software package.[4] We run the sampler for 1000 iterations with 500 warmup iterations and two chains to test for convergence.

2 Model Stan Code

The Stan code used to fit the model in the paper is as follows:

```
// Coronavirus tracking model
//
//
// January 5, 2021
functions {


   // if(r_in(3,{1,2,3,4})) will evaluate as 1
  int r_in(int pos,int[] pos_var) {

    for (p in 1:(size(pos_var))) {
       if (pos_var[p]==pos) {
       // can return immediately, as soon as find a match
          return p;
       }
    }
    return 0;
    }

  real partial_sum(int[,] y_slice,
                   int start, int end,
                   int[] tests,
                   int[] cases,
                   vector phi,
                   int[] country_pop,
                   int num_country,
                   int num_rows,
                   vector mu_poly,
                   vector sigma_poly,
                   vector poly1,
                   vector poly2,
                   vector poly3,
                   real[] alpha_test,
                   real alpha_infect,
                   matrix count_outbreak,
                   real world_infect,
                   vector month_cases,
                   vector suppress_effect_raw,
                   vector lockdown_effect_raw,
                   vector mob_effect_raw,
                   matrix Q_supp,
```

```
                matrix Q_supp2,
                matrix Q_lock,
                matrix Q_mob,
                int[] cc,
                real pcr_spec,
                real finding,
                real test_baseline,
                //real test_lin_counter,
                vector country_test_raw,
                int S,
                int G,
                int L,
                int R,
                int[] sero_row,
                int[,] sero,
                matrix mobility,
                vector[] mob_array,
                vector mob_alpha_const,
                vector[] lockdown_med_raw,
                vector[] suppress_med_raw,
                vector suppress_med_raw_fear,
                vector lockdown_med_raw_fear,
                real[] sigma_test_raw,
                vector country_test_raw2,
                vector country_test_raw3,
                //real test_lin_counter2,
                //real test_max_par,
                //vector test_max,
                matrix lin_counter,
                real[] mu_test_raw,
                real[] mu_test_raw2,
                real[] mu_test_raw3,
                real[] sigma_test_raw2,
                real[] sigma_test_raw3,
                matrix M_Sigma,
                vector fear,
                real fear_const,
                real sigma_fear) {

// big loop over states
real log_prob = 0;
for(r in 1:size(y_slice)) {

    int s = y_slice[r,1];
    int start2 = y_slice[r,2];
    int end2 = y_slice[r,3];

    vector[end2 - start2 + 1] prop_infected; // modeled infection rates for domestic transmission\

    int obs = end2 - start2 + 1;
    real poly_nonc1; // non-centered poly parameters
    real poly_nonc2; // non-centered poly parameters
    real poly_nonc3; // non-centered poly parameters
    real country1s;
```

```
real country2s;
real country3s;
real mu_infect;
real sd_infect;

vector[end2 - start2 + 1] prop_success;
vector[end2 - start2 + 1] prop_fail;
vector[end2 - start2 + 1] mu_cases;
vector[end2 - start2 + 1] mu_tests;
vector[G] mu_mob[end2 - start2 + 1];

poly_nonc1 = mu_poly[1] + sigma_poly[1]*poly1[s];
poly_nonc2 = mu_poly[2] + sigma_poly[2]*poly2[s];
poly_nonc3 = mu_poly[3] + sigma_poly[3]*poly3[s];

country1s = mu_test_raw[1] + sigma_test_raw[1]*country_test_raw[s];
country2s = mu_test_raw2[1] + sigma_test_raw2[1]*country_test_raw2[s];
country3s = mu_test_raw3[1] + sigma_test_raw3[1]*country_test_raw3[s];

// latent infection rate (unobserved), on the logit scale (untransformed)
// constrained to *always* increase

for(i in 1:obs) {
      if(i==1) {
        prop_infected[1] = alpha_infect +
        count_outbreak[start2,1] * poly_nonc1 +
          count_outbreak[start2,2] * poly_nonc2 +
            count_outbreak[start2,3] * poly_nonc3 +
            world_infect*month_cases[start2] +
            Q_supp[start2,1:S]*suppress_effect_raw +
            Q_lock[start2,1:L]*lockdown_effect_raw +
            Q_mob[start2,1:G]*mob_effect_raw;
      } else {
        prop_infected[i] = exp(alpha_infect +
        count_outbreak[start2+i-1,1] * poly_nonc1  +
          count_outbreak[start2+i-1,2] *poly_nonc2 +
            count_outbreak[start2+i-1,3] * poly_nonc3 +
            world_infect*month_cases[start2+i-1] +
            Q_supp[start2+i-1,1:S]*suppress_effect_raw +
            Q_lock[start2+i-1,1:L]*lockdown_effect_raw +
            Q_mob[start2+i-1,1:G]*mob_effect_raw) + prop_infected[i - 1];
      }
}

    //need a recursive function to transform to ordered vector



mu_infect = mean(prop_infected);
sd_infect = sd(prop_infected);

prop_success = prop_infected;
prop_fail = 1 - inv_logit(prop_infected);
```

```
mu_cases = inv_logit(pcr_spec + finding*prop_success);

log_prob += normal_lpdf(country_test_raw[s]|0,1); // more likely near the middle than the ends
log_prob += normal_lpdf(country_test_raw2[s]|0,1); // more likely near the middle than the ends
log_prob += normal_lpdf(country_test_raw3[s]|0,1);

log_prob += normal_lpdf(poly1[s]|0,1);
log_prob += normal_lpdf(poly2[s]|0,1);
log_prob += normal_lpdf(poly3[s]|0,1);

//mobility mediation

for(g in 1:G) {
  mu_mob[1:obs,g] = to_array_1d(mob_alpha_const[g] +
  Q_lock[start2:end2,1:L]*lockdown_med_raw[g]  +
  Q_supp[start2:end2,1:S]*suppress_med_raw[g]);
}

  log_prob += multi_normal_cholesky_lpdf(mob_array[start2:end2,1:G]|mu_mob,M_Sigma);

// fear mediation
  log_prob += normal_lpdf(fear[start2:end2]|fear_const + Q_lock[start2:end2,1:L]*lockdown_med_ra
                        Q_supp2[start2:end2,1:(S-1)]*suppress_med_raw_fear,sigma_fear);


  mu_tests = inv_logit(alpha_test[1] +
                  country1s * lin_counter[start2:end2,1] +
                  country2s * lin_counter[start2:end2,2] +
                  //country3s * lin_counter[start2:end2,3] +
                  test_baseline * prop_success);

// observed data model
// loop over serology surveys to add informative prior information

for(n in start2:end2) {

  int q = r_in(n,sero_row);

  if(q <= 0) {

    log_prob += beta_binomial_lpmf(cases[n]|country_pop[n],mu_cases[n-start2+1]*phi[1],(1-mu_cas
    log_prob += beta_binomial_lpmf(tests[n]|country_pop[n],mu_tests[n-start2+1]*phi[2],(1-mu_tes

  } else if(q > 0) {

    // scaling function. we use seroprevalance data to
    // set a ground truth for the relationship between covariates and
    // infections measured non-parametrically

    log_prob += binomial_lpmf(sero[q,1]|sero[q,2],inv_logit(prop_infected[n-start2+1]));

  }
}
```

```
      }
      return log_prob;
    }


}
data {
  int time_all;
  int num_country;
  int num_rows;
  int cc[num_rows]; // country counter
  int cases[num_rows];
  int tests[num_rows];
  int S; // number of suppression measures
  int G; // google mobility data (by type of mobility)
  int L; // just lockdown data (for hierarchical predictor)
  int R; // number of seroprevalence essays
  matrix[num_rows,S] suppress; // time-varying suppression measures
  matrix[num_rows,S-1] suppress2; // without COVID poll
  matrix[num_rows,G] mobility; // time-varying mobility measures
  matrix[num_rows,L] lockdown; // hierachical lockdown predictors
  vector[num_rows] fear; // COVID poll
  matrix[num_rows,3] count_outbreak;
  vector[num_rows] month_cases;
  vector[num_rows] test_max;
  int sero[R,2]; // sero-prevalence datas
  int sero_row[R];
  int country_pop[num_rows];
  matrix[num_rows,3] lin_counter;
  vector[2] phi_scale; // prior on how much change there could be in infection rate over time
  int states[num_country,3];
}
transformed data {

  matrix[num_rows,S] Q_supp;
  matrix[S, S] R_supp;
  matrix[S, S] R_supp_inverse;

  matrix[num_rows,S-1] Q_supp2;
  matrix[S-1, S-1] R_supp2;
  matrix[S-1, S-1] R_supp_inverse2;

  matrix[num_rows, G] Q_mob;
  matrix[G, G] R_mob;
  matrix[G, G] R_mob_inverse;

  matrix[num_rows, L] Q_lock;
  matrix[L, L] R_lock;
  matrix[L, L] R_lock_inverse;
  vector[G] mob_array[num_rows];


  // thin and scale the QR decomposition
  Q_supp = qr_Q(suppress)[, 1:S] * sqrt(num_rows - 1);
```

```
  Q_supp2 = qr_Q(suppress2)[, 1:(S-1)] * sqrt(num_rows - 1);
  R_supp = qr_R(suppress)[1:S, ] / sqrt(num_rows - 1);
  R_supp2 = qr_R(suppress2)[1:(S-1), ] / sqrt(num_rows - 1);
  R_supp_inverse = inverse(R_supp);
  R_supp_inverse2 = inverse(R_supp2);

  Q_mob = qr_Q(mobility)[, 1:G] * sqrt(num_rows - 1);
  R_mob = qr_R(mobility)[1:G, ] / sqrt(num_rows - 1);
  R_mob_inverse = inverse(R_mob);

  Q_lock = qr_Q(lockdown)[, 1:L] * sqrt(num_rows - 1);
  R_lock = qr_R(lockdown)[1:L, ] / sqrt(num_rows - 1);
  R_lock_inverse = inverse(R_lock);

  for(g in 1:G) {
    for(n in 1:num_rows) {
      mob_array[n,g] = mobility[n,g];
    }
  }


}
parameters {
  vector[num_country] poly1; // polinomial function of time
  vector[num_country] poly2; // polinomial function of time
  vector[num_country] poly3; // polinomial function of time
  real mu_test_raw[1];
  real finding; // difficulty of identifying infected cases
  //vector<lower=0,upper=1>[R] survey_prop; // variable that stores survey proportions from CDC data
  real<lower=0> world_infect; // infection rate based on number of travelers
  vector[S] suppress_effect_raw; // suppression effect of govt. measures, cannot increase virus transmis
  vector[L] lockdown_effect_raw;
  vector[L] lockdown_med_raw[G];
  vector[S] suppress_med_raw[G];
  vector[L] lockdown_med_raw_fear;
  vector[S-1] suppress_med_raw_fear;
  //real test_lin_counter;
  real test_baseline;
  //real test_lin_counter2;
  real mu_test_raw2[1];
  real mu_test_raw3[1];
  real pcr_spec; // anticipated 1 - specificity of RT-PCR tests (taken from literature)
  // constraint equal to upper limit spec of 98 percent, 2% FPR of PCR test results
  vector[3] mu_poly; // hierarchical mean for poly coefficients
  vector[G] mob_effect_raw;
  //real test_max_par;
  vector<lower=0>[3] sigma_poly; // varying sigma polys
  vector[G] mob_alpha_const; // mobility hierarchical intercepts
  vector[num_country] country_test_raw; // unobserved rate at which countries are willing to test vs. nu
  vector[num_country] country_test_raw2;
  vector[num_country] country_test_raw3;
  real alpha_infect; // other intercepts
  real alpha_test[1];
  vector<lower=0>[2] phi_raw; // shape parameter for infected
```

```
  real<lower=0> sigma_test_raw[1]; // estimate of between-state testing heterogeneity
  real<lower=0> sigma_test_raw2[1];
  real<lower=0> sigma_test_raw3[1];
  cholesky_factor_corr[G] M_Omega; // these are for the MVN for mobility data
  vector<lower=0>[G] M_sigma;
  real fear_const;
  real<lower=0> sigma_fear;
}
transformed parameters {
  vector[2] phi;

  phi = (1 ./ phi_scale) .* phi_raw;
}
model {
  matrix[G, G] M_Sigma;
  int grainsize = 1;

  sigma_poly ~ exponential(1);
  mu_poly ~ normal(0,30);
  mu_test_raw ~ normal(0,20);

  mu_test_raw2 ~ normal(0,20);
  mu_test_raw3 ~ normal(0,20);
  world_infect ~ normal(0,3);
  lockdown_effect_raw ~ normal(0,5);
  alpha_infect ~ normal(0,10); // this can reach extremely low values
  alpha_test ~ normal(0,20);

  phi_raw ~ exponential(1);
  mob_effect_raw ~ normal(0,5);
  suppress_effect_raw ~ normal(0,5);
  //test_max_par ~ normal(0,5);
  test_baseline ~ normal(0,20);
  //test_lin_counter ~ normal(0,20);
  //test_lin_counter2 ~ normal(0,20);

  mob_alpha_const ~ normal(0,5);
  pcr_spec ~ normal(0,20);

  finding ~ normal(0,20);
  sigma_test_raw ~ exponential(1);
  sigma_test_raw2 ~ exponential(1);
  sigma_test_raw3 ~ exponential(1);
  sigma_fear ~ exponential(.1);
  fear_const ~ normal(0,5);

  for(g in 1:G) {
    suppress_med_raw[g] ~ normal(0,5);
    lockdown_med_raw[g] ~ normal(0,5);
  }

  suppress_med_raw_fear ~ normal(0,5);
  lockdown_med_raw_fear ~ normal(0,5);
```

```
  M_Omega ~ lkj_corr_cholesky(4);
  M_sigma ~ cauchy(0, 2.5);

  M_Sigma = diag_pre_multiply(M_sigma, M_Omega); // Cholesky decomp for MVN for mobility

target += reduce_sum_static(partial_sum, states,
                     grainsize,
                    tests,
                     cases,
                     phi,
                     country_pop,
                     num_country,
                     num_rows,
                     mu_poly,
                     sigma_poly,
                     poly1,
                     poly2,
                     poly3,
                     alpha_test,
                     alpha_infect,
                     count_outbreak,
                     world_infect,
                     month_cases,
                     suppress_effect_raw,
                     lockdown_effect_raw,
                     mob_effect_raw,
                     Q_supp,
                     Q_supp2,
                     Q_lock,
                     Q_mob,
                     cc,
                     pcr_spec,
                     finding,
                     test_baseline,
                     //test_lin_counter,
                     country_test_raw,
                     S,
                     G,
                     L,
                     R,
                     sero_row,
                     sero,
                     mobility,
                     mob_array,
                     mob_alpha_const,
                     lockdown_med_raw,
                     suppress_med_raw,
                     suppress_med_raw_fear,
                     lockdown_med_raw_fear,
                     sigma_test_raw,
                     country_test_raw2,
                     country_test_raw3,
                     //test_lin_counter2,
                     //test_max_par,
```

```
                    //test_max,
                    lin_counter,
                    mu_test_raw,
                    mu_test_raw2,
                    mu_test_raw3,
                    sigma_test_raw2,
                    sigma_test_raw3,
                    M_Sigma,
                    fear,
                    fear_const,
                    sigma_fear);

}
generated quantities {

  // convert QR estimates back to actual numbers

  vector[S] suppress_effect; // suppression effect of govt. measures, cannot increase virus transmissio
  vector[L] lockdown_effect;
  vector[G] mob_effect;
  vector[L] lockdown_med[G];
  vector[S] suppress_med[G];
  vector[L] lockdown_med_fear;
  vector[S-1] suppress_med_fear;
  vector[num_rows] prop_infect_out;
  vector[num_rows] cov_out;

  suppress_effect = R_supp_inverse * suppress_effect_raw;
  lockdown_effect = R_lock_inverse * lockdown_effect_raw;
  mob_effect = R_mob_inverse * mob_effect_raw;

  for(g in 1:G) {
    lockdown_med[g] = R_lock_inverse * lockdown_med_raw[g];
    suppress_med[g] = R_supp_inverse * suppress_med_raw[g];
  }

  suppress_med_fear = R_supp_inverse2 * suppress_med_raw_fear;
  lockdown_med_fear = R_lock_inverse * lockdown_med_raw_fear;


  for(s in 1:num_country) {

        real poly_nonc1; // non-centered poly parameters
        real poly_nonc2; // non-centered poly parameters
        real poly_nonc3; // non-centered poly parameters

        int start2 = states[s,2];
        int end2 = states[s,3];
        int obs = end2 - start2 + 1;

        poly_nonc1 = mu_poly[1] + sigma_poly[1]*poly1[s];
        poly_nonc2 = mu_poly[2] + sigma_poly[2]*poly2[s];
        poly_nonc3 = mu_poly[3] + sigma_poly[3]*poly3[s];
```

```
        //poly_nonc1 = poly1[1];
        //poly_nonc2 = poly2[2];
        //poly_nonc3 = poly3[3];

    for(i in 1:obs) {
              if(i==1) {
                prop_infect_out[start2] = alpha_infect + count_outbreak[start2,1] * poly_nonc1 +
                    count_outbreak[start2,2] * poly_nonc2  +
                    count_outbreak[start2,3] * poly_nonc3  +
                    world_infect*month_cases[start2] +
                    Q_supp[start2,1:S]*suppress_effect_raw +
                    Q_lock[start2,1:L]*lockdown_effect_raw +
                    Q_mob[start2,1:G]*mob_effect_raw;

                 cov_out[start2] = alpha_infect + Q_mob[start2,1:G]*mob_effect_raw;

              } else {
                prop_infect_out[start2 + i - 1] = exp(alpha_infect + count_outbreak[start2+i-1,1] * poly
                    count_outbreak[start2+i-1,2] * poly_nonc2 +
                    count_outbreak[start2+i-1,3] * poly_nonc3 +
                    world_infect*month_cases[start2+i-1] +
                    Q_supp[start2+i-1,1:S]*suppress_effect_raw +
                    Q_lock[start2+i-1,1:L]*lockdown_effect_raw +
                    Q_mob[start2+i-1,1:G]*mob_effect_raw) + prop_infect_out[start2 + i - 2];

                cov_out[start2 + i - 1] = exp(alpha_infect + Q_mob[start2+i-1,1:G]*mob_effect_raw) + co
              }
        }
  }

}
```

3 Simulation

Because our model is fully generative, we can simulate it using Monte Carlo methods. The simulation presented here is a simplification of the model presented in the main paper for clarity of exposition. We do not have varying polynomial time trends but rather a single global time trend, and we do not implement the cumulative sum transformation on the latent vector. Despite these simplifications, the simulation is very important as it is the only way to demonstrate that the model is globally identified and can in fact capture unobserved parameters like suppression effects and relative infection rates. We also are able to show how the bias in only using observed cases and tests can affect the estimates of parameters.

The following R code generates data from the model and plots the resulted unobserved infection rate and observed values for tests and cases along with an exogenous suppression covariate:

```r
# simulation parameters
num_state <- 50

time_points <- 100
# allows for linear growth that later becomes explosive
polynomials <- c(.03,0.0003,-0.00001)

# factor that determines how many people a state is willing/able to test
# states that suppress or don't suppress
# induce correlation between the two

 cor_vars <- MASS::mvrnorm(n = num_state, mu = c(0, 5),
                           Sigma = matrix(c(1, -.5, -.5, 1), 2, 2))


state_test <- cor_vars[, 2]
suppress_measures <- cor_vars[, 1]

# size of states

state_pop <- rpois(num_state, 10000)

# assumt t=1 is unmodeled = exogenous start of the infection

t1 <- c(1, rep(0, num_state-1))

# create a suppression coefficient
# first is for preventing domestic transmission from occuring
# second is for preventing further domestic transmission once it starts

suppress1 <- -0.5
suppress2 <- -0.05

# high value of phi = high over-time stability
phi <- c(300, 300)

# parameter governing how hard it is to find infected people and test them
# strictly positive

finding <- 1.5

# recursive function to generate time-series data by state

out_poly <- function(time_pt, end_pt, time_counts, tested, case_count, rate_infected, pr_domestic) {

  if(time_pt==1) {
    time_counts <- as.matrix(c(1, rep(0, num_state-1)))
    rate_infected <- as.matrix(c(.0001, rep(0, num_state-1)))
    tested <- as.matrix(rep(0, num_state))
    case_count <- as.matrix(c(1, rep(0,num_state-1)))
  }

  # if at time = t infected, start time tracker at t
```

```
# need to know how many states have reported at least one case = infection start

world_count <- sum(case_count[, time_pt]>0)


if(time_pt==1) {

  rate_infected_new <-  plogis(-5 + time_counts[, time_pt]*polynomials[1] +
                                 suppress1*suppress_measures +
                  suppress2*suppress_measures*time_counts[, time_pt] +
                  .05*sum(world_count) +
    (time_counts[, time_pt]^2)*polynomials[2] +
      (time_counts[, time_pt]^3)*polynomials[3])

  # conservative time counter that only starts when first case is recorded

  time_counts_new <- ifelse(time_counts[, time_pt]>0 | case_count[, time_pt]>0, time_counts[, time_pt]

} else {

  rate_infected_new <- plogis(-5 + time_counts[, time_pt]*polynomials[1] +
                  suppress1*suppress_measures +
                  suppress2*suppress_measures*time_counts[, time_pt] +
                  .05*sum(world_count) +
    (time_counts[, time_pt]^2)*polynomials[2] +
      (time_counts[, time_pt]^3)*polynomials[3])

  # conservative time counter that only starts when first case is recorded

  time_counts_new <- ifelse(time_counts[, time_pt]>0 | case_count[, time_pt]>0, time_counts[, time_pt]
}

# of these, need to calculated a set number tested
mu_test <- plogis(-7 + state_test*rate_infected_new)
tested_new <- rbbinom(num_state, state_pop, mu_test*phi[1], (1-mu_test)*phi[1])

# determine case count as percentage number tested
# this is what we always observe
mu_case <- plogis(-2.19 + finding*rate_infected_new)
case_count_new <- rbbinom(num_state, tested_new, mu_case*phi[2], (1-mu_case)*phi[2])


if(time_pt<end_pt) {
  out_poly(time_pt = time_pt+1,
           end_pt = end_pt,
           time_counts = cbind(time_counts,
                               time_counts_new),
           rate_infected = cbind(rate_infected, rate_infected_new),
           tested = cbind(tested, tested_new),
           case_count = cbind(case_count, case_count_new))
} else {
  return(list(time_counts = time_counts,
              tested = tested,
```

```
                rate_infected = rate_infected,
                case_count = case_count))
  }

}

check1 <- out_poly(1, time_points)

check1 <- lapply(check1, function(c) {
  colnames(c) <- as.numeric(1:time_points)
  c
})

all_out <- bind_rows(list(time_counts=as_tibble(check1$time_counts),
                          `Proportion Population\nInfected`=as_tibble(check1$rate_infected),
                          `Number of Cases`=as_tibble(check1$case_count),
                          `Proportion of Cases from Domestic Transmission`=as_tibble(check1$pr_domestic
                          `Number of Tests`=as_tibble(check1$tested)),.id="Series")

all_out$state <- rep(paste0("state_",1:num_state),times=length(check1))
all_out$suppress_measures <- rep(suppress_measures,times=length(check1))

all_out %>%
  gather(key = "time_id",value="indicator",-Series,-state,-suppress_measures) %>%
  mutate(time_id=as.numeric(time_id)) %>%
  filter(!(Series %in% c("time_counts"))) %>%
  ggplot(aes(y=indicator,x=time_id)) +
  geom_line(aes(colour=suppress_measures,group=state),alpha=0.3) +
  xlab("Days Since Outbreak") +
  ylab("") +
  facet_wrap(~Series,scales="free_y") +
  theme(panel.background = element_blank(),
        panel.grid=element_blank(),
        strip.background = element_blank(),
        strip.text = element_text(face="bold"),
        legend.position = "top")
```

Figure 2 shows one line for each state's trajectory from a total of 100 states and 100 time points. As can be seen, the shading indicating strength of suppression policies diverges substantially over time. However, the numbers of observed tests and cases show far more random noise due to the difficulty in inferring the true rate from the observed data. It is possible that some states simply want to test more, and end up with more cases, or that the infection rate is in fact higher. As such, this model is able to incorporate that measurement uncertainty between the true (unobserved) rate and the observed indicators, tests and cases.

The data were also generated so that the suppression covariate, which shades the infection rates in Figure 2, is positively correlated with a state's willingness to test. As such, this covariation will lead a naive model to dramatically *over-estimate* the effect of suppression policies as the case/test ratio mechanically falls due

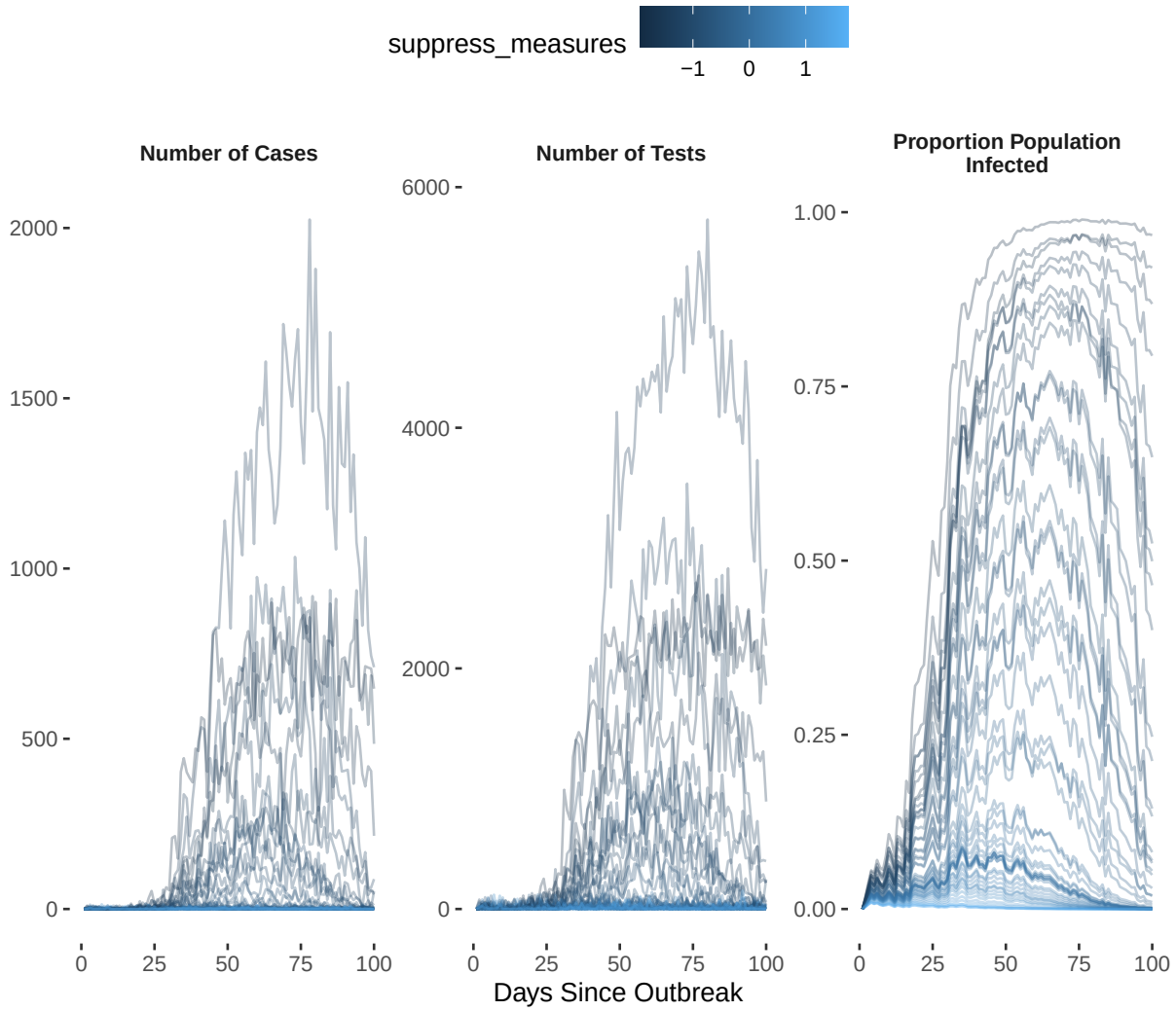


Figure 2: Simulation of Observed Tests and Cases Given Unobserved Infectious Process

to increased tests independent of the infection rate.

The primary advantage of this model is that it allows for the testing of covariates that affect the true infection rate without requiring more heavy-duty approaches like SEIR/SIR, such as modeling reproduction numbers and other disease-specific mechanisms. The intention is to have a more parsimonious model that can see how the effect of variables like suppression measures have on different states/states infection numbers over time. In particular, this model increases our understanding of the connection between contextual factors and human behavior to the virus' progression.

The model could be further extended with more complicated processes, such as spatial modeling, but for the purposes of this exposition we do not look further at such extensions. We would note that the world infection parameter is implicitly, if not explicitly, a spatial measure.

3.1 Estimation

We can then fit an empirical model using the Hamiltonian Monte Carlo (HMC) Markov Chain Monte Carlo (MCMC) sampler in Stan[4] to model the unobserved infection rate given the simulated observed data. We also fit a Bayesian binomial model of the proportion of counts to state population using the same covariates but with the observed counts as the outcome. This naive model is fitted to indicate the amount of bias that can occur in estimates as a result of ignoring the underlying infection process.

```
# all data from simulation
# primarily case and test counts

# need to make centered, ortho-normal polynomials

ortho_time <- poly(scale(1:time_points, scale = F),degree=3)

init_vals <- function() {
  list(phi1 = 300,
       phi2 = 300,
       world_infect = .1,
       finding = 1,
       poly = c(0, 0, 0),
       state_test = rnorm(num_state, 5, .25),
       alpha = c(-7, 0,-2),
       sigma_test_raw = 1)
}

sim_data <- list(time_all = time_points,
                 num_country = num_state,
                 country_pop = state_pop,
                 cases = check1$case_count,
                 S = 1,
```

```
                  phi_scale = 1/100,
                  ortho_time = ortho_time,
                  tests = check1$tested,
                  count_outbreak = as.numeric(scale(apply(check1$time_counts, 2, function(c) sum(c>0)),
                                                    scale = FALSE)),
                  time_outbreak = check1$time_counts,
                  time_outbreak_center = matrix(scale(c(check1$time_counts), scale = FALSE), nrow = nrow
                                                ncol = ncol(check1$time_counts)),
                  suppress = as.matrix(suppress_measures))

# need to make a regular type data frame to do observed modeling with rstanarm

obs_data <- as_tibble(check1$case_count) %>%
  mutate(num_state = 1:n()) %>%
  gather(key = "time_points", value="cases", -num_state)


# join in outbreak timing + covariates

time_data <- as_tibble(sim_data$time_outbreak) %>%
  mutate(num_state = 1:n()) %>%
  gather(key = "time_points", value = "time_outbreak", -num_state) %>%
  mutate(time_points = stringr::str_extract(time_points, "[0-9]+"))

obs_data <- left_join(obs_data, time_data, by = c("time_points", "num_state")) %>%
  left_join(tibble(state_pop=state_pop,
                   suppress=suppress_measures,
                   num_state=1:length(state_pop)),by="num_state") %>%
  mutate(time_points=as.numeric(time_points),
         time_points_scale=as.numeric(scale(time_points,scale=T))) %>%
  left_join(tibble(count_outbreak=sim_data$count_outbreak,
                   time_points=as.numeric(1:time_points)),by="time_points")

if(run_model) {

  pan_model <- stan_model("corona_tscs_betab.stan")

# run model

pan_model_est <- sampling(pan_model,data=sim_data,chains=2,cores=2,iter=1200,warmup=800,init=init_vals,
                          control=list(adapt_delta=0.95))

naive_model <- stan_glm(cbind(cases,state_pop-cases)~poly(time_points_scale,3) +
                          count_outbreak +
                          suppress +
                          suppress:poly(time_points_scale,3)[,1],
                        data=obs_data,
                        family="binomial",
                        cores=1,
                        chains=1)

saveRDS(pan_model_est,"data/pan_model_est.rds")
saveRDS(naive_model,"data/naive_model_sim.rds")
```

```
} else {
  pan_model_est <- readRDS("data/pan_model_est.rds")
  naive_model <- readRDS("data/naive_model_sim.rds")
}
```

After fitting the latent model, we can access the estimated infection rates and plot them:

```
all_est <- as.data.frame(pan_model_est,"num_infected_high") %>%
  mutate(iter=1:n()) %>%
  gather(key="variable",value="estimate",-iter) %>%
  group_by(variable) %>%
  mutate(estimate=estimate) %>%
  summarize(med_est=quantile(estimate,.5),
            high_est=quantile(estimate,.95),
            low_est=quantile(estimate,.05)) %>%
  mutate(state_num=as.numeric(str_extract(variable,"(?<=\\[)[1-9][0-9]?0?")),
         time_point=as.numeric(str_extract(variable,"[1-9][0-9]?0?(?=\\])")))

all_est <- left_join(all_est,tibble(state_num=1:num_state,
                                    suppress_measures=suppress_measures),by="state_num")

all_est %>%
  ggplot(aes(y=med_est,x=time_point)) +
  geom_ribbon(aes(ymin=low_est,
  ymax=high_est,
  group=state_num,
  fill=suppress_measures),alpha=0.5) +
  theme_minimal() +
  scale_color_brewer(type="div") +
  ylab("Latent Infection Scale") +
  xlab("Days Since Outbreak Start") +
  theme(panel.grid = element_blank(),
        legend.position = "top")
```

We see how the model is able to partially recover the infection rate as shown in Figure 3. The estimates reveal the same general arc as the generated data, with higher suppression policies associated with lower infection counts, but the scale of the infection rate is no longer identified. As such, the model is only able to recover the relative rather than absolute trajectory of the infection rate.

However, because we have inferred the correct arc, we can also see what the estimated suppression parameters are. We also compare those to the same suppression parameters from the naive model in Figure 4.

```
require(bayesplot)
require(patchwork)

p1 <- mcmc_intervals_data(as.array(pan_model_est,pars=c("suppress_effect[1,1]",
                                                        "suppress_effect[2,1]")))

p2 <- mcmc_intervals_data(naive_model,pars=c("suppress",
```

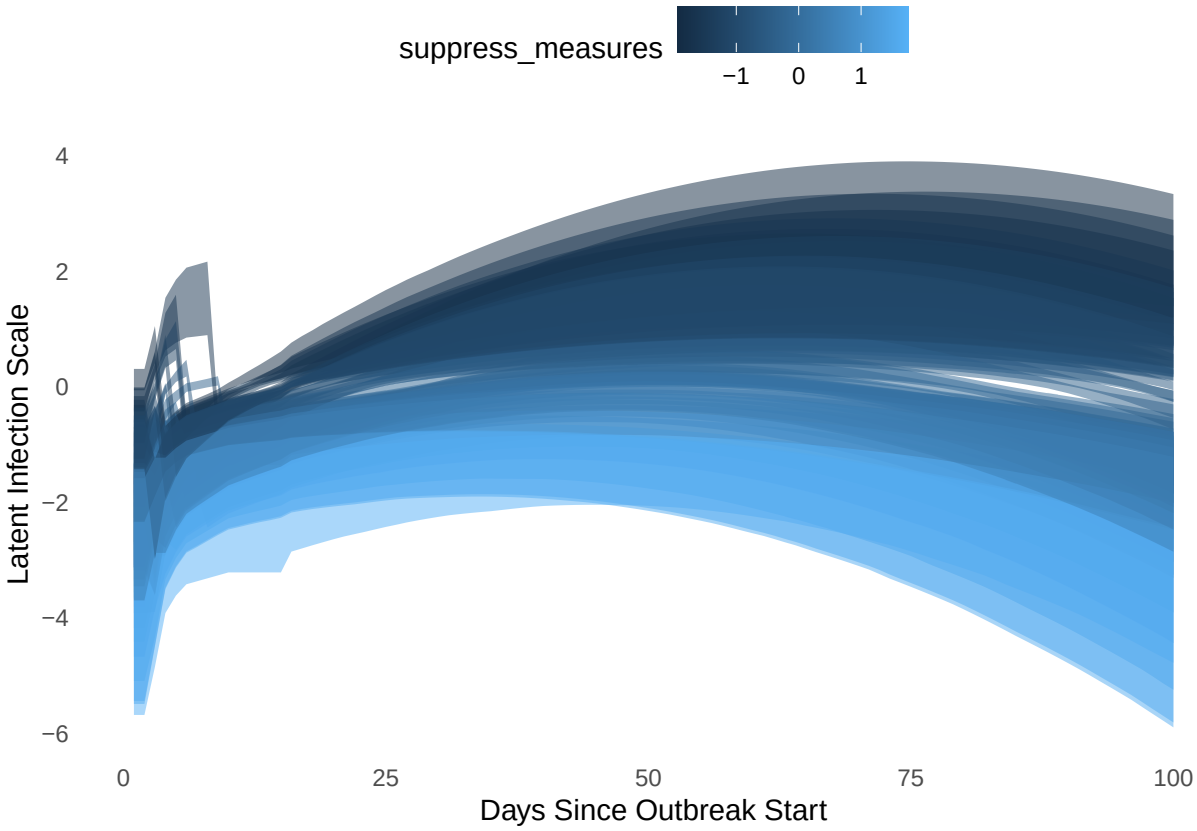


Figure 3: Estimated Simulated Infection Rates

```
                                                "suppress:poly(time_points_scale, 3)[, 1]"))

bind_rows(list(Latent=p1,
               Observed=p2),.id="Model") %>%
  mutate(parameter=recode(parameter,
                          `suppress_effect[1,1]`="Constant\nEffect",
                          `suppress_effect[2,1]`="Over-Time\nEffect",
                          `suppress`="Constant\nEffect",
                          `suppress:poly(time_points_scale, 3)[, 1]`="Over-Time\nEffect")) %>%
  ggplot(aes(y=m,x=Model)) +
  geom_pointrange(aes(ymin=ll,ymax=hh),position=position_dodge(0.5)) +
  theme_minimal() +
  geom_hline(yintercept=0,linetype=3) +
  scale_color_brewer(type="qual") +
  guides(color="none") +
  ylab("Parameter Estimate") +
  theme(panel.grid = element_blank(),
        legend.position = "top") +
  coord_flip() +
  facet_wrap(~parameter,scales="free_x",ncol=1)
```

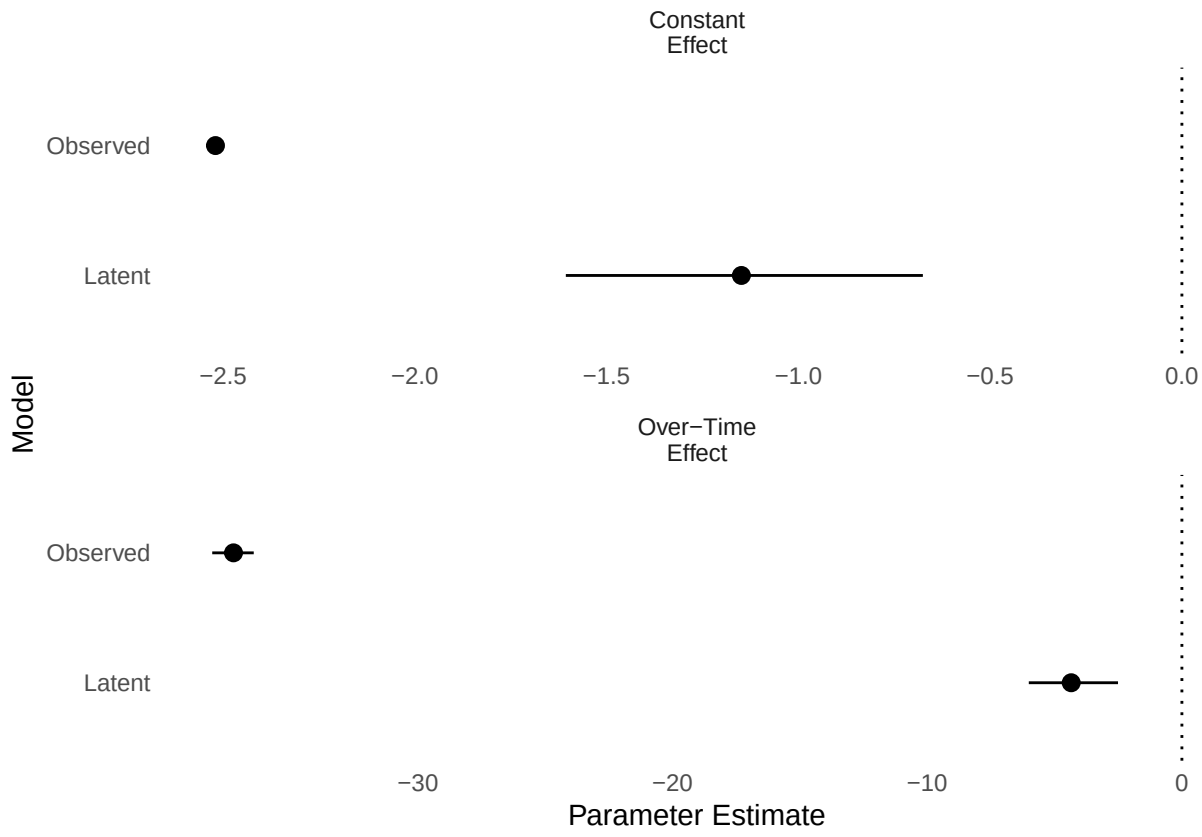


Figure 4: Recovered Simulated Virus Suppression Parameters Versus Naive Model

We can see in Figure 4 that despite the uncertainty in not perfectly observing the infection rate, we can still

get a precise credible interval on the suppression effect. However, while the observed model's parameters have the same sign, they are wildly inflated, and far too precise. The constant effect in particular is ten times the size of the latent model with virtually no uncertainty interval. Because the infection process is ignored, the model obfuscates the infection/testing relationship with the effect of suppression policies, wildly over-estimating their effect at lowering infection counts.

The advantage of this model, as can be seen, is that with testing numbers and case counts, we can model the effect of state-level (or region-level) variables on the unseen infection rate up to an unknown constant. It is far simpler than existing approaches while still permitting inference on these hard-to-quantify measures. It is no substitute for infectious disease modeling–for example, it produces no estimates of the susceptible versus recovered individuals in the population, nor death rates–but rather a way to measure the effect of different background factors of interest to social and natural sciences on disease outcomes as well as approximate disease trajectory. Furthermore, the model's main quantity is a better measure to share with the public than misleading numbers of positive case counts.

References

1. Ferrari, S. & Cribari-Neto, F. Beta regression for modelling rates and proportions. *Journal of Applied Statistics* **31**, 799–815 (2004).

2. Winship, C. & Mare, R. D. Structural equations and path analysis for discrete data. *The American Journal of Sociology* **89**, 54–110 (1983).

3. Yuan, Y. & MacKinnon, D. P. Bayesian mediation analysis. *Psychological methods* **14**, 301–322 (2009).

4. Carpenter, B. *et al.* Stan: A probabilistic programming language. *Journal of Statistical Software* **76**, (2017).