Supplementary Information for A Retrospective Bayesian Model for Measuring Covariate Effects on Observed COVID-19 Test and Case Counts

Robert Kubinec & Luiz Max Carvalho

April 10th, 2020

1 Materials and Methods

In this document we put forward a model for the rate of infected people $I_{ct} \in (0, 1)$ in a given state/region c given the the number of cases a_{ct} and the number of tests q_{ct} for all time periods $t \in T$. The outcome is here defined as the tests and cases reported on a given day rather than the cumulative total for ease of modeling purposes (i.e., the lagged difference of the cumulative total). We assume that I_{ct} is an unobserved Beta-distributed variable with a 3-order polynomial time trend of the number of post-outbreak time periods $T_O < T$, where an outbreak begins at the first reported case in a given area. By using a single time function, the model assumes that the coronavirus follows a similar pattern of infection across states, as appears to be the case (i.e., all states are infected with the same virus).

The unobserved infection rate I_{ct} is a function of the following parameters:

$$\Pr(I_{ct} \mid T = t) \sim \text{Beta}(\alpha_1 + \beta_{O1} \sum_{c=1}^{C} \mathbf{1}(a_{ct'} > 0) \forall t' < t + \beta_{S1} X_c + \tag{1}$$

$$\beta_{I1} t_o + \beta_{I2} t_o^2 + \beta_{I3} t_o^3 + \beta_{S2} X_c, \phi), \tag{2}$$

where $g(\cdot)$ is the inverse logit function, $\beta_{O1} \sum_{c=1}^{C} \mathbf{1}(a_{ct-1} > 0) \forall t' < t$ is the sum of states with at least one case of infection in the world at any previous time point, which we term the world infection rate, and the three β_{Ii} are polynomial coefficients of the number of post-outbreak time periods t_o.

The world infection parameter measures the spread of the virus due to travel across state borders, and as such will increase rapidly as more states are infected. By contrast, the polynomial time trends represent

possible within-state transmission of the virus, which can cause exponentially growing case counts compared to the linear sum of infected states parameter. Finally, the parameter ϕ is a dispersion parameter governing how precise the infection rate estimate is.

Given these two primary ways that the infection rate can increase, there are two ways that possible suppression measures targeted at the outbreak can enter the model. The first is a constant factor, β_{S1}, which is a parameter indicating the strength of state-level suppression measures X_c. This constant parameter is meant to capture the ability of states or regions to stop the spread of transmission due to foreign travelers arriving in the state/region, which occurs at a roughly constant level over time.

The second way suppression measures enter the model is through $\beta_{S2}X_c$, which can increase over time as the virus increases. This parameter reflects possible measures which will grow more effective as domestic transmission of the virus increases (i.e. as the polynomial time trend takes off). As such, it is assumed that any deviation from the common domestic transmission pattern is due to these time-varying suppression measures.

Once an outbreak has started in state c, which is indicated by a positive case count in time $t-1$, a time counter $t_o = 1$ starts for that state and increases by 1 for each following time point until T.

Given this specification of the infection rate process, we can then move to the generation of the observed data, tests q_{ct} and cases a_{ct}. The infection rate is assumed to influence both of these quantities. First, an increasing number of infections is associated with more tests as states try to identify who may have the virus. Furthermore, a rising infection rate is associated with a higher ratio of positive results (reported cases) conditional on the number of tests. We model both of these observed indicators, tests and cases, jointly to simultaneously adjust for the infection rate's influence on both factors.

To model the number of tests, we assume that each state has an unobserved level of testing parameter, $\beta_{cq} > 0$, indicating how strongly each state is willing and able to perform tests as a factor of the unobserved infection rate. The number of observed tests q_{ct} for a given time point t and state c conditional on the states' population, c_p, has a binomial distribution:

$$q_{ct} \sim \text{Binomial}(c_p, g(\alpha_2 + \beta_{cq} I_{ct})). \tag{3}$$

The parameter β_{cq} serves to scale the infection rate I_{ct} so that an increasing infection rate has heterogeneous effects on the number of tests by state. The intercept α_2 indicates how many tests would be performed in a state with an infection rate of zero. It is assumed that this number is quite low, though not necessarily zero.

Given the parameter β_{cq}, a state could test almost no one or test far more than are actually infected depending on their willingness to impose tests. However, the number of tests is increasing in I_{ct} conditional

on a state's willingness to test people. That is, regardless of how much a state wants to test people, as the outbreak grows the number of tests will increase though at very different rates.[1]

Given the number of observed tests q_{ct}, we can then generate the number of observed cases a_{ct} as a binomial random variable conditional on the number of tests q_{ct}:

$$a_{ct} \sim \text{Binomial}(q_{ct}, g(\alpha_3 + \beta_a I_{ct})), \tag{4}$$

where $g(\cdot)$ 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 (equal to the false positive rate of the test), and $\beta_a > 0$ 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 observed number of cases a_{ct} as a proportion of the number of observed tests q_{ct}.

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 conditional on a certain number of tests. This simultaneous adjustment helps takes care of mis-interpreting the observed data by not taking into account varying testing rates, which is likely why some policy makers argue that the epidemiological models are wrong.

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

$$q_{ct} \sim \text{Beta-Binomial}(c_p, g(\alpha_2 + \beta_q I_{ct}), \phi_q), \tag{5}$$

and cases:

$$a_{ct} \sim \text{Beta-Binomial}(q_{ct}, g(\alpha_3 + \beta_a I_{ct}), \phi_a). \tag{6}$$

For computational reasons, the infection rates I_{ct} are put in to the beta-binomial model on the logit scale instead of transforming them to (0,1), but they can be transformed to proportions post-estimation with the inverse logit function.

1.1 Identification

This model contains an unobserved latent process I_{ct}, and as such there are further constraints necessary in order to have a unique scale and rotation of the latent variable. The most important restrictions are

[1] For a very compelling visualiation of this process with empirical data from the COVID-19 pandemic, we refer the reader to this website: https://ourworldindata.org/grapher/covid-19-tests-cases-scatter-with-comparisons.

positivity constraints on the parameters β_a and β_q that govern the relationship between the infection rate and test and case counts. It is assumed that infection rates will increase both the number of tests and the number of cases relative to the number of tests, though at necessarily different rates.

The other important restriction is to fix the intercept for the cases model α_3 to a fixed value of 0.1, or -2.2 on the logit scale. The reason for this restriction is because the intercept necessarily equals the false positive rate of a COVID-19 test. While there is still ongoing research as to the false positive rate, it is clear that it is likely in the area of 1 in 10 false positives for rapid tests.[2] As such, by pinning this value, we can lower-bound the number of cases we are likely to see given an infection rate of zero, providing a helpful lower-bound for the estimate.

As we will show, no other identification restrictions are necessary to estimate the model beyond weakly informative priors assigned to parameters. These are:

$$\beta_a \sim \text{Exponential}(.1), \tag{7}$$

$$\beta_{qc} \sim \text{Exponential}(\sigma_q), \tag{8}$$

$$\sigma_q \sim \text{Exponential}(.1), \tag{9}$$

$$\beta_{Si} \sim \text{Normal}(0, 2), \tag{10}$$

$$\beta_{Ii} \sim \text{Normal}(0, 5), \tag{11}$$

$$\alpha_1 \sim \text{Normal}(0, 10), \tag{12}$$

$$\alpha_2 \sim \text{Normal}(0, 10), \tag{13}$$

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

The one prior to note is that a hierarchical regularizing prior is put on the varying testing adjustment parameters β_{qc} for regularization purposes due to the limited data available to inform the parameter.

Other than these weakly informative priors, the model is identified, as we show in the next section. However, it is important to emphasize that there is no information in the model that identifies the *true* number of infected people. Rather, the infection rate is a latent process, and as such it is not known exactly what scale to assign to it without further information. However, both the relative growth in infection rates are identified, along with the effect of suppression measures, so *the model is useful without being fully identified.* Furthermore, by incorporating insights from SIR/SEIR models we can also identify the latent scale with reasonable informative priors, as we show in the data analysis section.

[2]For a discussion, see https://www.realclearpolitics.com/articles/2020/03/18/the_perils_of_mass_coronavirus_testing_142693.html.

1.2 Simulation

Because this model is fully generative, we can simulate it using Monte Carlo methods. 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. The following R code generates data from the model and plots the resulted unobserved infection rate and observed values for tests and cases:

```
# 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 1 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 1, 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 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

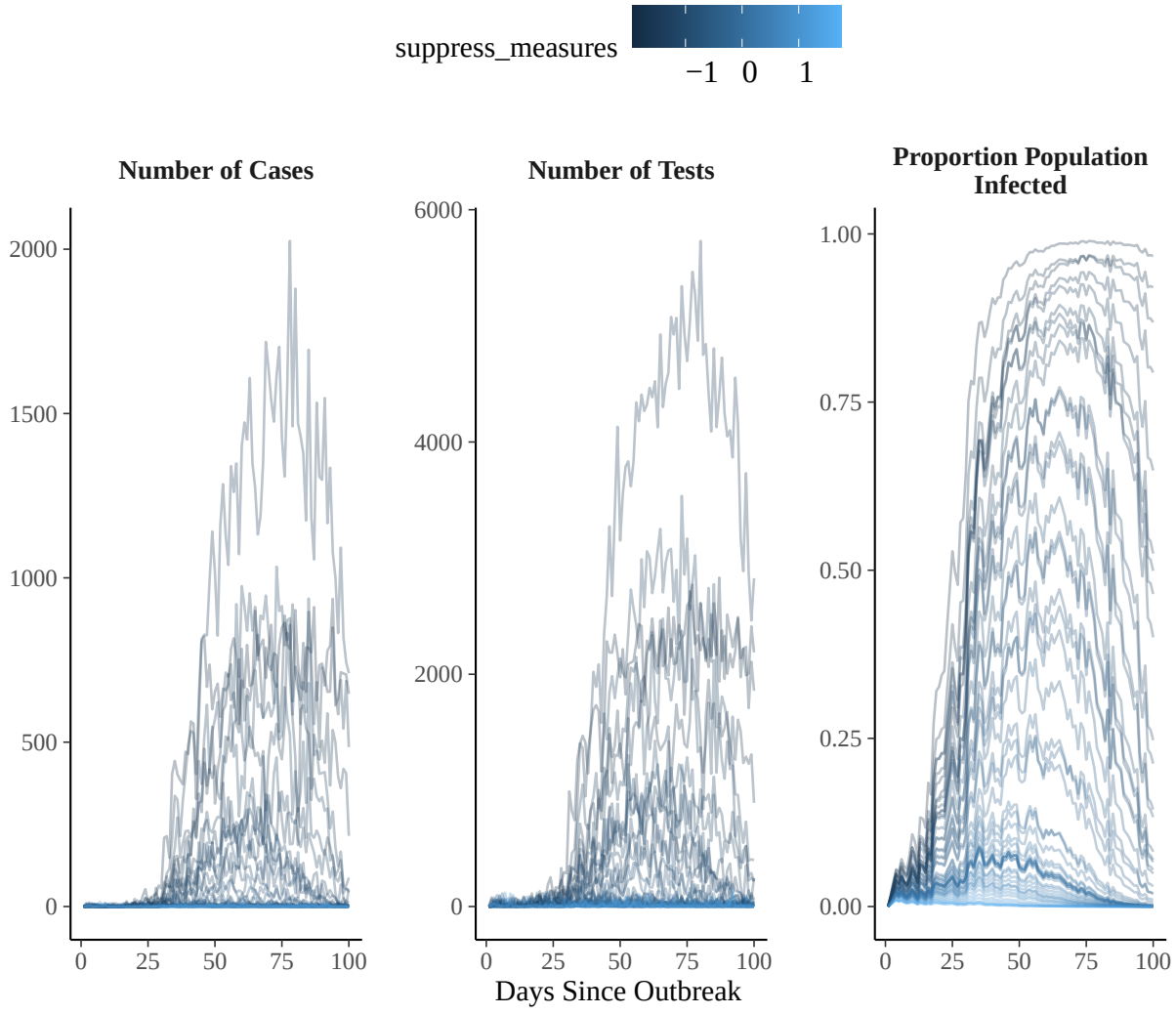


Figure 1: Simulation of Observed Tests and Cases Given Unobserved Infectious Process

infection parameter is implicitly, if not explicitly, a spatial measure.

1.3 Estimation

We can then fit an empirical model using the Hamiltonian Monte Carlo (HMC) Markov Chain Monte Carlo (MCMC) sampler in Stan[1] 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 specification as (2). 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),
       sigma_test_raw = 1)
}

sim_data <- list(time_all = time_points,
                 num_state = num_state,
                 state_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 2. 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 3.

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

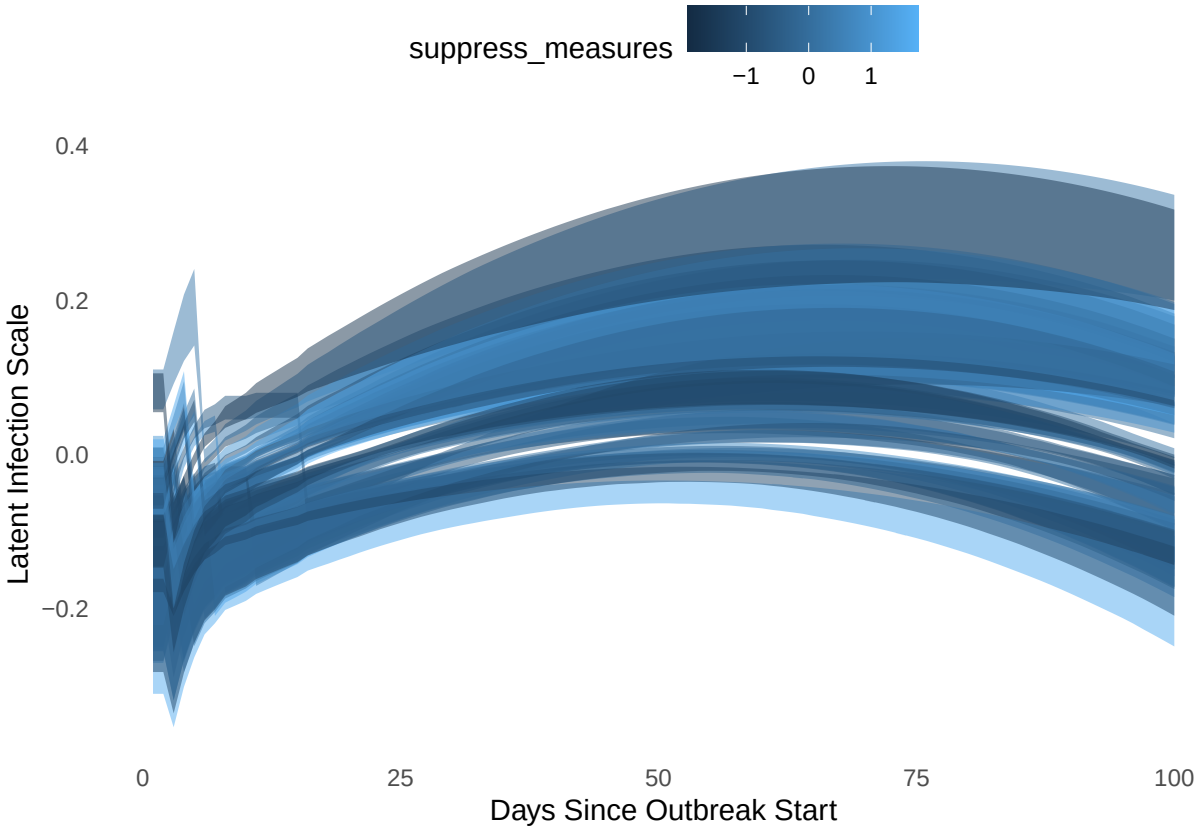


Figure 2: Estimated Simulated Infection Rates

```
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)
```

We can see in Figure 3 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.

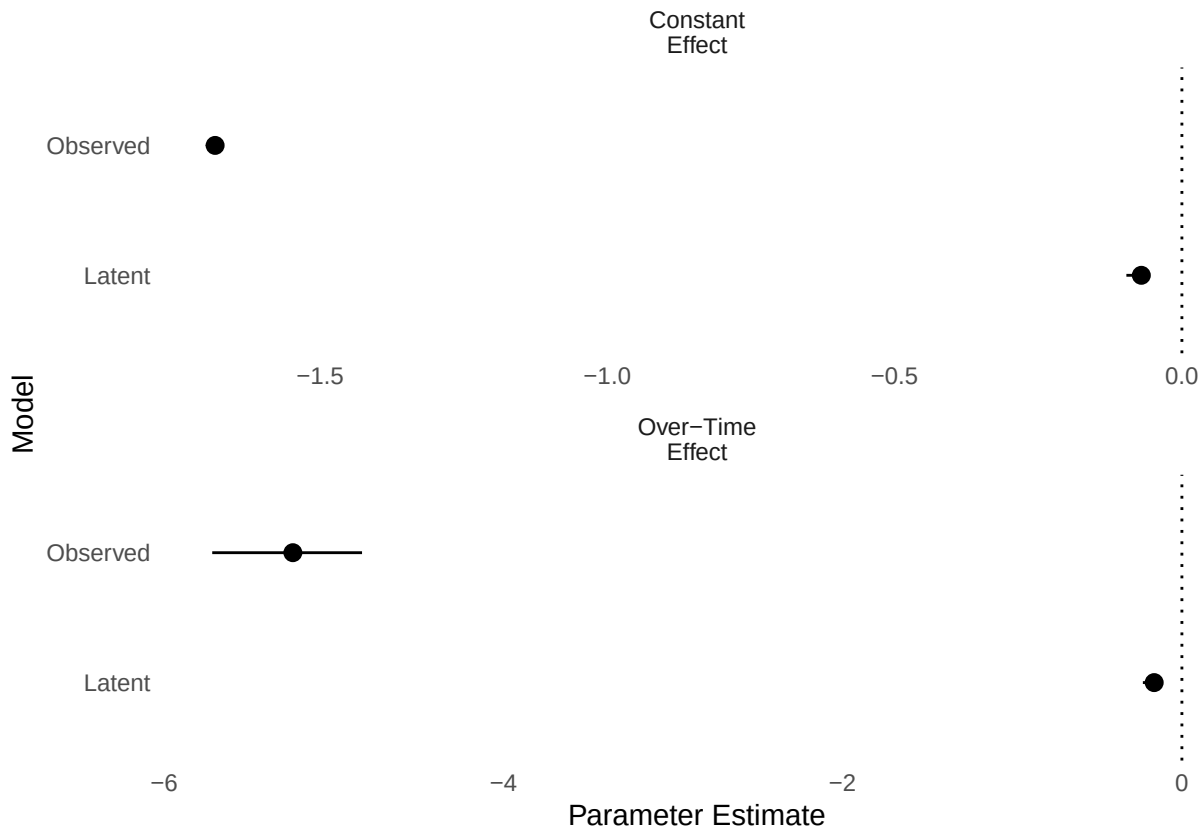


Figure 3: Recovered Simulated Virus Suppression Parameters Versus Naive Model

While the partial identification of this model is interesting and potentially useful in situations where nothing else can be inferred about the virus, it is possible to further identify the scale of the latent variable by incorporating information from SIR/SEIR models, which we demonstrate in the next section.

2 Identifying the Latent Scale

To show how we can further attempt to identify the scale of the latent variable, instead of only relative differences between states, we show in this section how we can add in information from SEIR/SIR modeling to identify the total number of infected persons in the model. The crucial missing piece of information in the model is the ratio between the proportion of infected persons I_{ct} and the proportion of tests per state population q_{ct}:

$$\frac{q_{ct}}{I_{ct}} \tag{14}$$

Another way of framing the problem is to think of how much the number of tests should increase given a one-percentage increase in the infection rate. How many of these infected people will be tested? Without an idea of the true number of infected people, it is impossible to answer this question and thus identify the latent scale.

However, an increasing number of SIR/SEIR papers show that it is likely that as few as 10% of the total number of infected persons are actually recorded as cases, including in the United States. This number provides us with a conservative lower bound if we consider that the number of tests should be at least 10% of the total proportion of infected persons. In other words, we can consider adding the following information into the model:

$$\frac{q_{ct}}{I_{ct}} > 0.1. \tag{15}$$

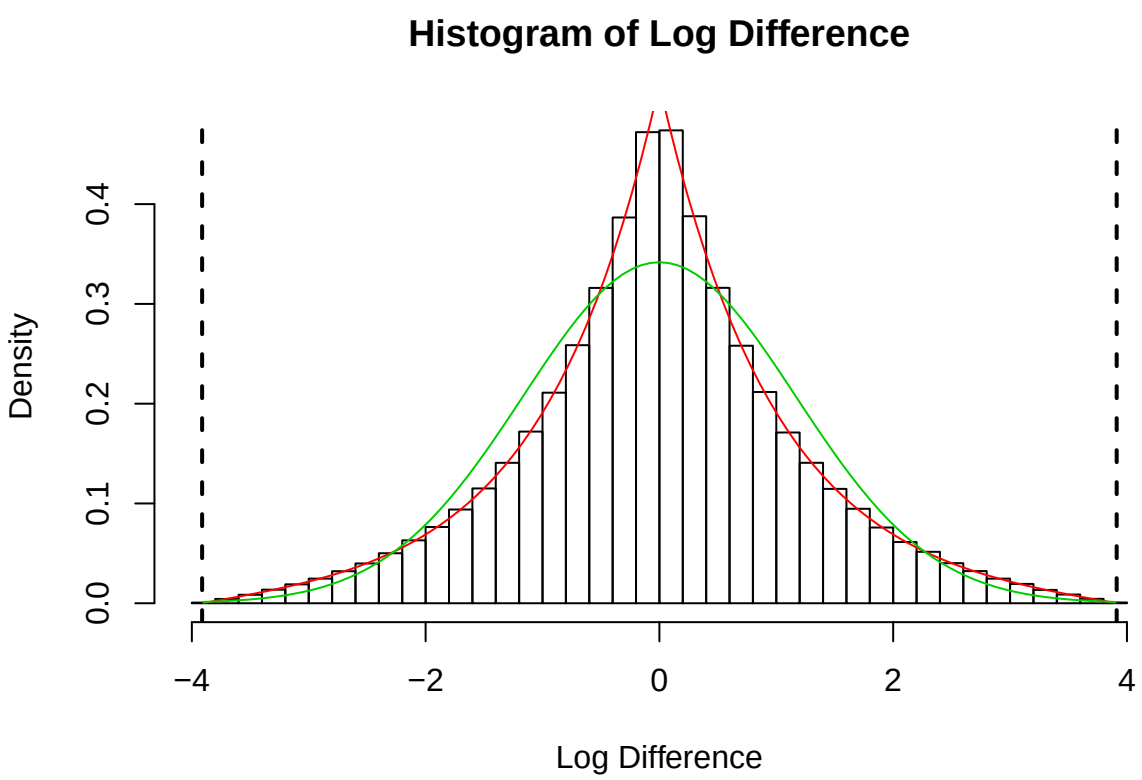
Every percentage increase in the infection rate should result in at least a 0.1% increase in the total number of people tested as a percentage of the population, though the upper bound is unknown (some states may test many more than are actually infected). It is difficult to impose this constraint directly in the model; however, we can consider adding it as prior information if we can define a prior density over the ratio. First, for computational simplicity, we can consider a distribution over the log differences of the parameters:

$$\log q_{ct} - \log I_{ct}. \tag{16}$$

By simulating from uniform distributions of possible rates for $\log q_{ct}$ and $\log I_{ct}$, it is possible to tell that the a realistic distribution of log differences with small density less than 0.1 is in fact very close a standard normal distribution:

```
# this code generates a log distribution of a ratio of two probabilities.
f_y <- function(y, a, b, log = FALSE){
  lc <-  log(b-a)
  ans <- y  - lc
  if(!log) ans <- exp(ans)
  return(ans)
}
#
get_Y_var <- function(a, b){
  EXsq <- ( b*(log(b)^2 - 2*log(b) + 2) -  a*(log(a)^2 - 2*log(a) + 2) )/(b-a)
  EX <- (b*(log(b) - 1) - a*(log(a) - 1) ) /(b-a)
  ans <- EXsq - (EX)^2
  return(ans)
}
##
analytic_W <- function(w, a, b){  ## assumes i.i.d.
  c0 <- all(w > a/b, w < b/a)
  k <- (b-a)*(b-a)
  m <- max(a, a/w)
  M <- min(b, b/w)
  soln <- function(L, U) ((U *abs(U))- (L *abs(L)))/(2*k)
  d0 <- soln(L = m, U = M)
  dens <- c0 * d0
  return(dens)
}
analytic_W <- Vectorize(analytic_W)
##
f_z <- function(z, a, b, log = FALSE){
  ans <- log(analytic_W(exp(z), a, b)) + z
  if(!log) ans <- exp(ans)
```

```r
  return(ans)
}
f_z <- Vectorize(f_z)
###############

M <- 1E6
a <-  .01
b <-  .5
Y <- log(runif(M, min = a, max = b))

#hist(Y, probability = TRUE)
# curve(f_y(x, a, b), min(Y), max(Y), add = TRUE, lwd = 2)
# abline(v = c(log(a), log(b)), lwd = 2, lty = 2)
# integrate(function(x) f_y(x, a, b), log(a), log(b))

###

Y1 <- log(runif(M, min = a, max = b))
Y2 <- log(runif(M, min = a, max = b))

`Log Difference` <- Y1 - Y2

# integrate(function(x) f_z(x, a, b), log(a)-log(b), log(b)-log(a))
# integrate(function(x) x^2* f_z(x, a, b), log(a)-log(b), log(b)-log(a))
#
# 2*get_Y_var(a, b)
# Var(`Log Difference`)
#
hist(`Log Difference`, breaks = 50, probability = TRUE)
curve(f_z(x, a, b), min(`Log Difference`), max(`Log Difference`), col = 2, add = TRUE)
curve(dnorm(x, 0, sqrt(2*get_Y_var(a, b))), min(`Log Difference`), max(`Log Difference`), col = 3, add =
abline(v = c(log(a)-log(b), log(b)-log(a)), lwd = 2, lty = 2)
```



If one desires to choose different parameters a and $b > a$ for the distribution of $\log q_{ct}$ and $\log I_{ct}$, one may either (i) use the code above (`get_Y_var`) to obtain an approximating Gaussian – with variance given by `2*get_Y_var(a, b)` – or (ii) use the exact expression for the log of the ratio of two i.i.d. uniform random variables (`f_z`). Here, we will employ the standard Gaussian as it is good and tractable approximation of the exact distribution.

We can see in the figure above that the standard normal provides an approximate fit to the density of two probabilities where there is limited density on values below log -2 or 0.14. On the other hand, the prior puts substantial mass on ratios of tests to infected that are quite high, such as log2, 7.38. This informative prior, which still allows for a range of possible ratios, allows me to use SEIR/SIR model conclusions while still permitting uncertainty over the underlying relationship.

For these reasons, we add this term to the joint posterior for each time point t and state c:

$$\log q_{ct} - \log I_{ct} \sim \text{Normal}(0, 1). \tag{17}$$

We also add a log Jacobian adjustment to the posterior density to reflect the fact that this prior is a non-linear

function of parameters we have already assigned priors to:

$$\log(1 - q_{ct}) + \log(1 - I_{ct}). \tag{18}$$

The Jacobian adjustment is the derivative of the non-linear functions with respect to the underlying parameters, which in this case are the natural log and the inverse logit function.

References

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