A filtering approach to estimation of the epidemic exponential growth rate r(t) from incidence timeseries

Matthew So (somatthewc@gmail.com), Jonathan Dushoff

April 2021

Contents

1 Introduction

Outbreaks of infectious diseases such as the ongoing COVID-19 pandemic pose a threat to population health. In order to combat these outbreaks, it is important for public health authorities to understand how quickly an epidemic is spreading. One metric for studying epidemic spread is the exponential growth rate r(t). [[*Not exactly true. Maybe better to say something like methods have not been well studied:*]] [[*Fixed.*]] Methods for estimating this metric using incidence data have not been well-studied. We propose an estimation method that is conceptually simple, easily implemented, and requires few assumptions about infection dynamics. This method involves filtering the raw observed incidence data with a Savitzky-Golay filter to remove periodicity and reduce noise, then using another Savitzky-Golay filter on the logarithm of the filtered incidence data to perform logarithmic differentiation. This estimation method has been evaluated on a custom modified discrete-time SEIR model incorporating a separate periodic observation process resulting in a greater number of observations on certain days of the week. The proposed estimation method is effective at estimating r(t) in simulated data, even given periodic and noisy signals similar to those found in real-world data. Using this estimation method on real-world data results in qualitatively stable estimates. The proposed estimation method

may make r(t) estimation quicker and more accurate, which could allow for better management of infectious disease outbreaks. Our modified SEIR model may also be useful for generating timeseries representative of real-world incidence data, which can be used to improve other epidemiological parameter estimation methods.

2 Modelling

2.1 Rationale

We need to obtain a ground-truth r(t) estimate before evaluating a potential r(t) estimation method. An SEIR-like model was developed which contained some dynamical features found in real-world COVID-19 data, such as noise and periodicity.

[[*I'm not going to have time to micro-edit; you can work on making your writing sharper, shorted and more direct.*]] [[*Fixed.*]]

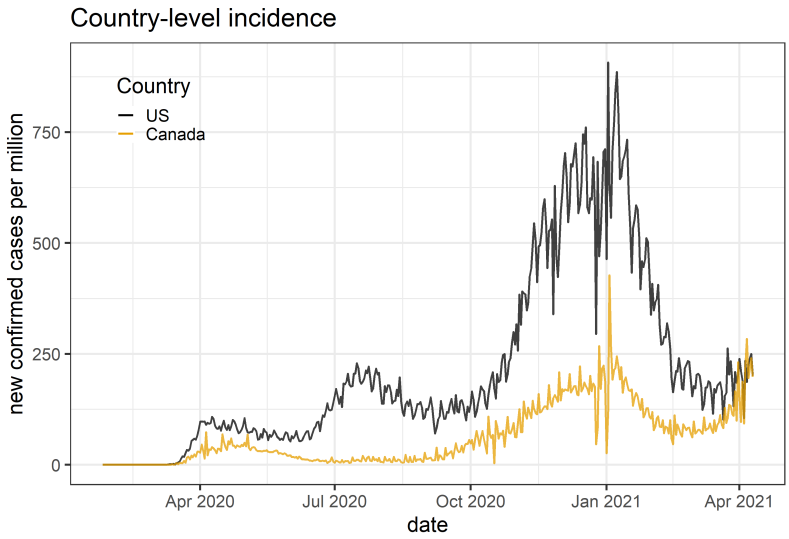

Figure 1: Real-world COVID-19 incidence data contains noise and "spiky" periodicity. [1]

[[*You can probably get better incidence data and avoid that weird anomaly. If I have time I'll send you a link.*]] [[*I can change Canada to a different country that doesn't have this anomaly? Otherwise, I await the link to the different data source.*]]

It is hypothesized in this work that this "spikiness" primarily reflects daily changes in observations. Any trends in infection, such as changes in r(t), would be heavily smoothed by the incubation period; however, this is inconsistent with the consistent single-day spikes found in real data.

2.2 Model Explanation

2.3 About the model

The current model is a discrete-time model based on the SEIR framework, which also models weekday trends in observation data. The main idea behind the model is that the probability density function for an infectee's observation time is modified such that observation on a Monday (for example) is more likely than the other days. Parameters were chosen to represent COVID-19 as closely as possible.

2.4 Explanations for features in the model

2.4.1 Conventions

The sum of two probability distributions X and Y is denoted as $X + Y$, and is computed as the convolution of their probability density functions. The expected value is denoted as $E[X]$. Temporary variables have self-explanatory names.

2.4.2 State variable explanations

All state variables are initialized to 0 except $S \leftarrow 10,000,000$. On day 0, 10 infections occur (see Simulation for details). [[*It would be good to mention the initial value of incidence here – and probably even better to just start with a non-zero value of I.*]] [[*A nonzero value of I doesn't work because observations and recoveries are modified when a new person is infected. I now mention the initial value of incidence.*]]

- S: susceptible.
- E: exposed, but non-infectious
- I: infectious
- R: recovered (or dead)
- O: cumulative infectious individuals who have been observed
- t: time since epidemic start.

2.4.3 Parameter/Function explanations

- $\beta(t)$: Same meaning as in typical SEIR models.
- $\mathcal{R}_0$: The basic reproductive number (that is, the reproductive number not scaled by S/N. Implicitly used to define $\beta(t)$. See results for details.
- `baseObservationDist`: Probability density function of time for an infection to be observed. This PDF is modified based on the day of the week. Currently set to discrete lognormal distribution with μ=1.7, log-SD=0.5
- `incubationDist`: Probability density function of time to go from $E \rightarrow I$. Currently set to discrete lognormal distribution with μ=1.63, log-SD=0.5. [2]
- `infectiousDist`: Probability density function of time to go from $I \rightarrow R$. (time spent infectious). Currently set to exponential distribution with mean=13 days. (Note: Non-exponential distributions are supported for $r(t)$ estimation, but an exponential distribution has been chosen to support $\mathcal{R}(t)$ estimation in *Appendix: $\mathcal{R}(t)$ Estimation*. It is unlikely that this parameter is truly exponentially distributed in COVID-19, but the mean is consistent with existing literature. [3])

- κ: 1/(dispersion parameter) in an alternatively parameterized negative binomial distribution [4]. negBinom(mean, 0) is implemented as the Poisson. I chose 0 as the value of κ, implicitly making all instances of negBinom actually Poisson.
- negBinom: Negative binomial distribution parameterized by (mean, κ).
- `dayScalers`: On each weekday, the probability of observation is multiplied by this value. Set to 1.1 for Monday and Tuesday and 1 otherwise.
- `observationProb`: The total probability that an infected individual will be observed. Set to 0.8.
- $t_{\max}$: Maximal value of t for simulation. Set to 401.

[[*You should avoid writing words as math products. Use mathrm or something. This is probably not worth fixing for now unless you have extra time but I wanted to say it. Like do you* not *think your tmax looks ugly?*]] [[*Fewer words should be written as math products now. I may not have caught them all, though.*]]

2.4.4 Derived variable explanations

- `incidence`: The number of new individuals infected today that weren't infected yesterday. At t=0, incidence is set to 10.
- μ: The reciprocal of E[`incubationDist`], analogous to the $E \to I$ controlling parameter in SEIR model.

2.5 Explanation of model logic

2.5.1 Key assumptions

1. The infectious disease being modeled follows SEIR dynamics.
2. When an individual is infected, the time that they will be detected at and the time they will become infectious at are defined by two independent distributions.
3. Dynamical noise does not exist; that is, the true number of people infected each day is deterministic.
4. However, not every person who is infected will be observed. Observation noise can result in a non-deterministic number of infections being observed each day.

2.5.2 Pre-simulation setup

1. Compute an observation distribution for every day in the week. For each weekday, modify a copy of `baseObservationDist` such that each weekday in the distribution is multiplied by the corresponding item in dayScalers. Then, renormalize this distribution to have a sum of 1.

2. Set values for each state variable. Set $N \leftarrow \text{sum}(S, E, I, R)$. Additionally, set $t \leftarrow -1$ for setup purposes.

2.5.3 Simulation

These steps are executed for each desired value of t between 0 and $t_{\max}$.

1. Compute the weekday, $t \pmod 7$.

2. If t=0, then initialize `incidence` to some number. Else, compute `incidence` $= SI\beta(t)/N$.

3. Transition event times are put into a separate list with each element at each index representing the number of events at (index) days from now.

4. Transition events are distributed amongst all future days with a value equal to their respective probability density functions. $E \to I$ events are proportional to `incubationDist`, $I \to R$ is proportional to `incubationDist + infectiousDist`, and $E \to I$ is proportional to the appropriate `weekdayObservationDist` scaled by `obervationProb`. [[*Unclear. You are using E-to-I to represent two different things it looks like (progression to infectiousness, and observation).*]]

5. Execute all other transition events occurring today. For $S \to E, S = S - 1, E = E + 1$. For $E \to I, E = E - 1, I = I + 1$. For $I \to R, I = I - 1, R = R + 1$. For $E \to O$, compute n_{obs}, the number of $E \to O$ transitions that would happen today. Then, $O = O + negBinom(n_{obs}, \kappa)$.

6. Store all state variables as well as `incidence` from today.

2.5.4 Post-simulation

1. Compute `scaledIncidence` by multiplying `incidence` by `observationProb`.

2. Compute the ground truth r(t) by taking the first differences of log(`incidence`). (The r(t) for a given day is equal to $\log(t+1) - \log(t)$). The ground truth is undefined for days on which `incidence` $= 0$ or $t = t_{\max}$.

[[*It's good practice to say* $\log(x)$ *instead of* $log(x)$.]] [[*Fixed.*]]

3 r(t) Estimation

3.1 The Savitzky-Golay filter

The Savitzky-Golay filter is a low-pass filter typically used for signal processing applications. The principle behind the Savitzky-Golay filter is that, for each time t, a polynomial regression model is fit from the points surrounding it [5]. For this work, three additional parameters are considered: window size, polynomial order, and derivative order.

For each time t, a polynomial regression model is fit using the incidence data from the times $[t - \texttt{windowWidth}, t + \texttt{windowWidth}]$, where $\texttt{windowWidth} = \text{floor}(\texttt{windowSize}/2)$. For this work, polynomial order is set to 1 for all filters (making each window a linear regression). Then, the value of the filter at each time t is equal to the `derivativeOrder`th derivative of the best-fit polynomial at time t. For the left `windowWidth` points, the regression model is fit to the first `windowSize` points in the timeseries, and the value of the filter at each of the first `windowSize` points is equal to the regression model's prediction at each of these points. The analogous method is applied to the right `windowSize` points, using the last `windowWidth` points to train the regression model. [[*The floor is imprecise (since not every windowSize is possible as written). Better to say* $s = 2w + 1$.]]

While confidence interval estimation for this filter is not standard in signal processing libraries, it can be done for derivative order 0 by taking the confidence interval on each predicted filter value [6]. (Other combinations may also be possible; for example, for derivative order 1 and polynomial order 1 by taking the confidence interval on the slope of the linear regression.)

3.2 Estimation algorithm

1. The observed incidence timeseries is filtered using a Savitzky-Golay filter of window size 7, polynomial order 1, and derivative order 0, saving both the mean value of the filter and the confidence interval at each point. This is intended as an initial low-pass filter on the incidence data, which acts to reduce periodicity and noise. (Note: There are other types of initial low-pass filters that could conceivably be used if confidence intervals are not important, see Appendix: Filtering for other possibilities). [[*You don't use the word 'week' in this ¶. Be more explanatory.*]]

2. Define a function `estim(incidence, windowSize, shiftAmt)`.

   (a) `logIncidence` $\leftarrow \log($`incidence`$)$.

   (b) Filter `logIncidence` using a Savitzky-Golay filter of length `windowSize`, polynomial order 1, and derivative order 1. NaNs are ignored (not used as points in the linear regression). Only the mean estimate at each point needs to be considered. [[*The NaN thing should be explained and maybe changed. It's a bias, and you're going to say it's unimportant because we can't get decent estimates for those windows*

anyway – so maybe just refuse to do estimates for those windows (return NA).]]

(c) Shift the resulting curve backwards by `shiftAmt`, which should be set to the rounded expected value of the time between infection and observation.

(d) Return the shifted curve, which is the mean r(t) estimate given the input incidence timeseries.

3. Sample a new plausible incidence timeseries. That is, for `nBootstrap` iterations, for each point in the filtered timeseries obtained from step 1, randomly sample a value from the confidence interval of the point and append it to a new timeseries. Store all new plausible incidence timeseries.

4. Call `estim` on each plausible incidence timeseries to obtain an r(t) estimation.

5. Call `estim` on the mean value of the filtered timeseries obtained from step 1. This is the mean r(t) estimate.

6. For each time t, compute the lower 0.025 and upper 0.975 quantiles of the r(t) estimated from the plausible incidence timeseries. This is the lower and upper confidence interval bounds on the r(t) estimates. (For example, if the plausible incidence timeseries was stored in a `length(incidence)` $\times$ `nBootstrap` matrix where `length(incidence)` is the number of rows, then compute quantiles row-wise to obtain a `length(incidence)` $\times 1$ matrix.)

`windowSize` can be chosen as any odd number, but 7 and 15 have been tested. Lower numbers should react to changes more quickly but should also result in noisier estimates, and vice versa.

3.3 Caveats and potential improvements

- Confidence intervals assume a normally distributed confidence interval; this may be changed to a t-distributed confidence interval later on).
- Confidence intervals are also too wide for smoothing windows of 7 days, but sometimes too narrow for smoothing windows of 15 days. Confidence interval estimates should be improved in the future.
- Using deconvolution rather than shifting is more theoretically sound, but deconvolution amplifies noise inherent in the data [7] [8]. See Appendix: Deconvolution for details.

4 Results

4.1 Simulation - Smooth transitions

4.1.1 Modelling

Varying $\mathcal{R}_0(t)$ was used to define model behavior. These parameters were chosen to simulate a rise, peak, and decline in COVID-19 incidence.

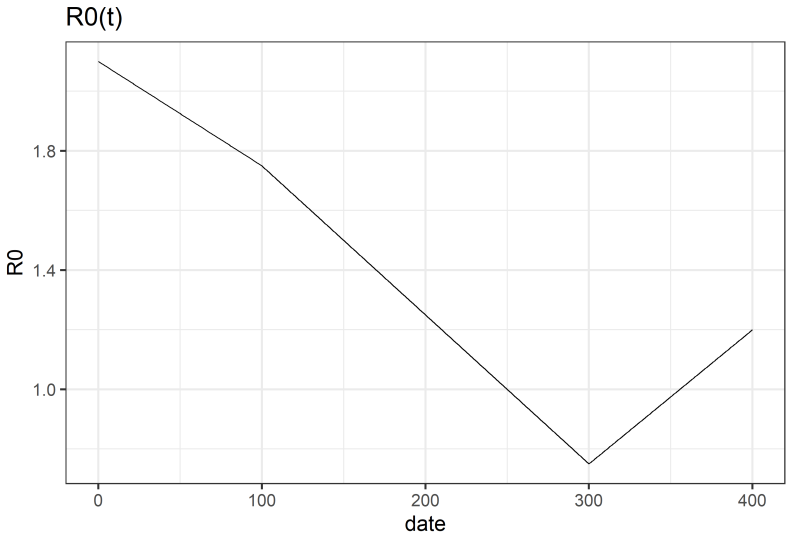

Figure 2: A piecewise linear function was used to define gradual changes in $\mathcal{R}_0(t)$, which drives the incidence in this model.

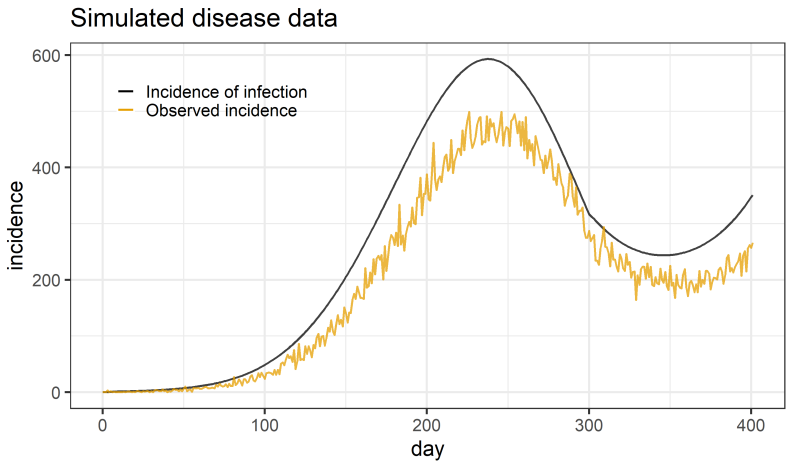

Figure 3: A plot of modeled incidence over time. The incidence of infection is perfectly deterministic, and increases exactly when an individual is infected. The observed incidence is periodic, has observation noise, is delayed by the time delay distribution between infection and observation. Only 80% of infections are observed.

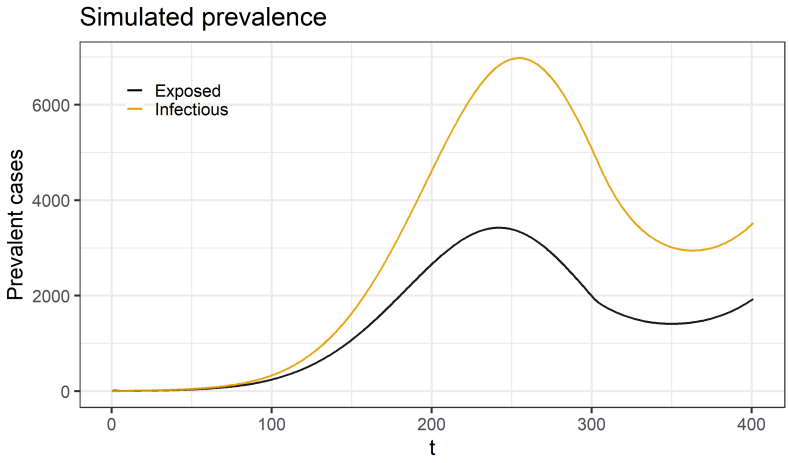

Figure 4: A plot of modeled prevalence over time. As all transitions aside from observations are deterministic, the number of infectious and exposed (non-infectious/in incubation period) individuals are also deterministic.

4.1.2 Smoothing and Estimation

The first step in the estimation algorithm is an initial filtered of the incidence data to reduce noise and periodicity. See the r(t) estimation section for more details.

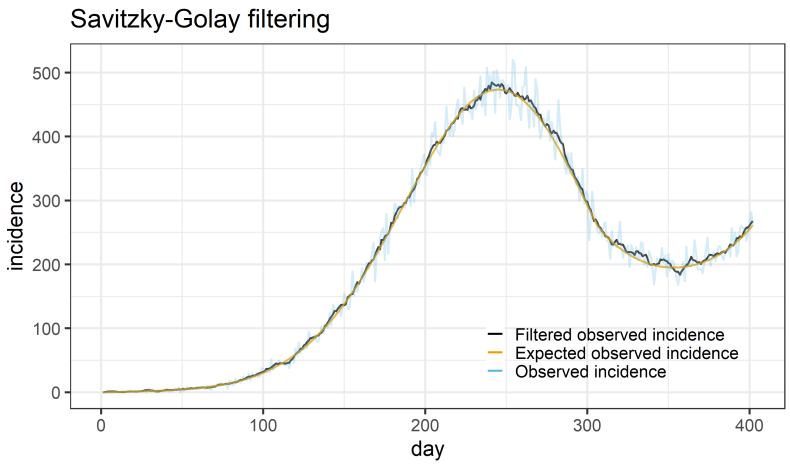

Figure 5: Application of a first-pass Savitzky-Golay filter substantially reduces noise and periodicity. The "expected observed" data is shown as a comparison, and is the incidence of infection forward-convolved by the mean infection-observation delay distribution.

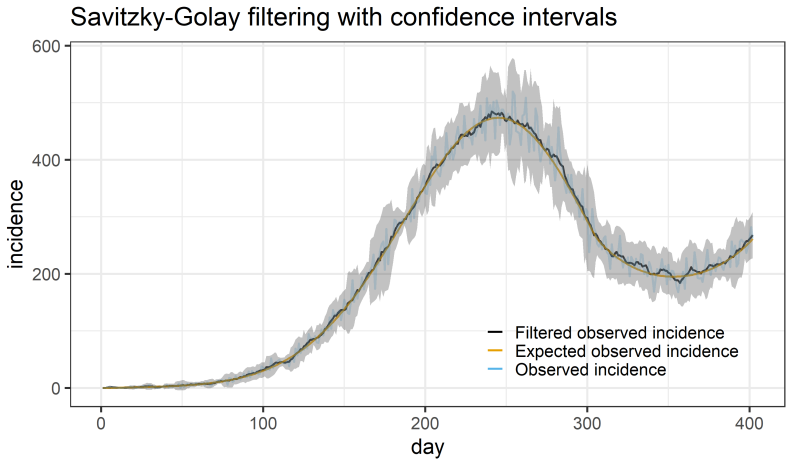

Figure 6: The same first-pass Savitzky-Golay filter as shown in Figure 5, with the confidence intervals shown.

The next step in the estimation algorithm produces r(t) estimates from the filtered data. Window sizes can be chosen arbitrarily, and have been set to 7 and 15 days in the following figures.

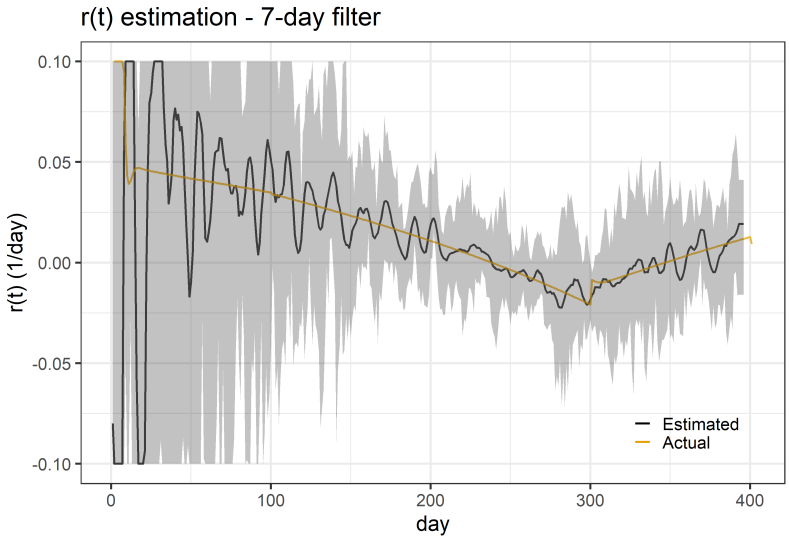

Figure 7: r(t) estimation using simulated data, with a 7-day post-filtering Savitzky-Golay window size. All values are clipped between -0.1 and 0.1 for visualization purposes. The true r(t) is within the 95% confidence interval 99% of the time.

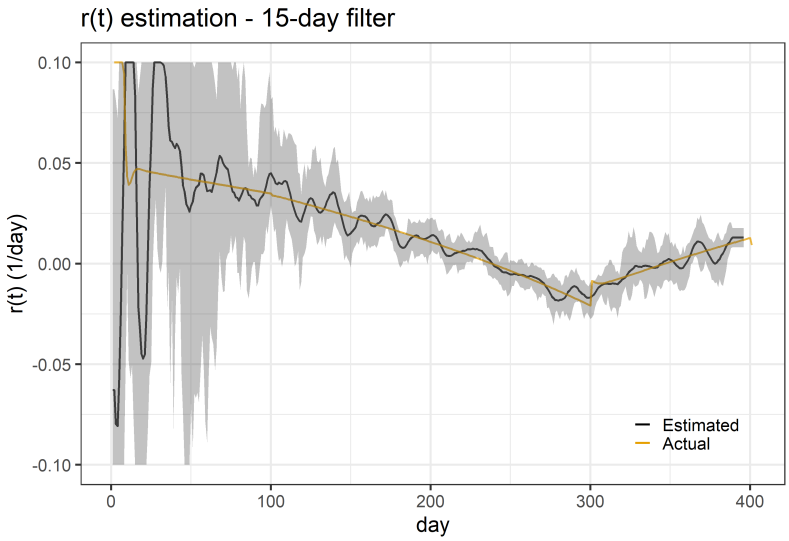

Figure 8: r(t) estimation using simulated data, with a 15-day post-filtering Savitzky-Golay window size. All values are clipped between -0.1 and 0.1 for visualization purposes. The true r(t) is within the 95% confidence interval 97% of the time.

4.2 Simulation - Instantaneous transitions

4.2.1 Modelling

Varying $\mathcal{R}_0(t)$ was used to define model behavior. Once again, parameters were chosen to simulate a rise, peak, and decline in COVID-19 incidence.

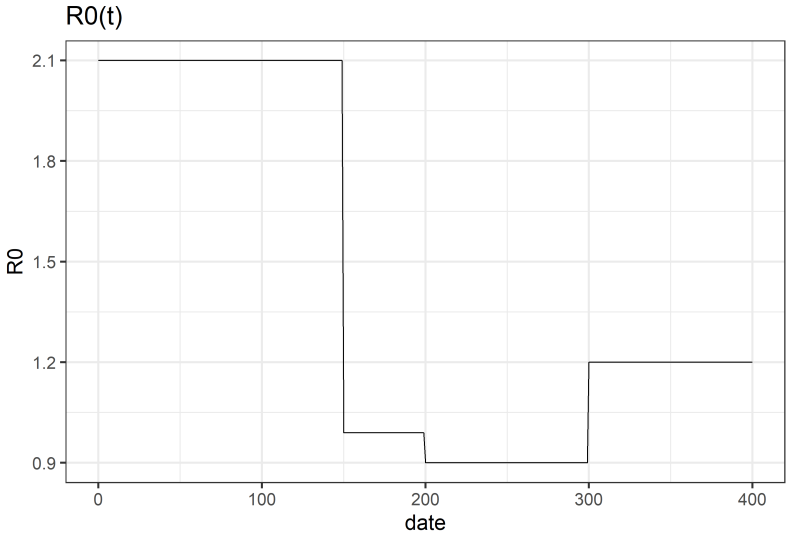

Figure 9: A more *blocky* piecewise linear function was used to define gradual changes in $\mathcal{R}_0(t)$, which drives the incidence in this model.

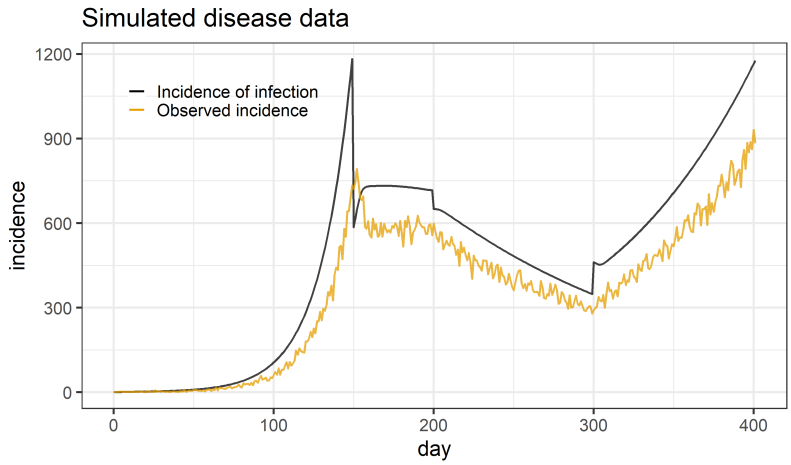

Figure 10: A plot of modeled incidence over time. The incidence of infection is perfectly deterministic, and increases exactly when an individual is infected. The observed incidence is periodic, has observation noise, is delayed by the time delay distribution between infection and observation, and reflects the 80% of observed infections.

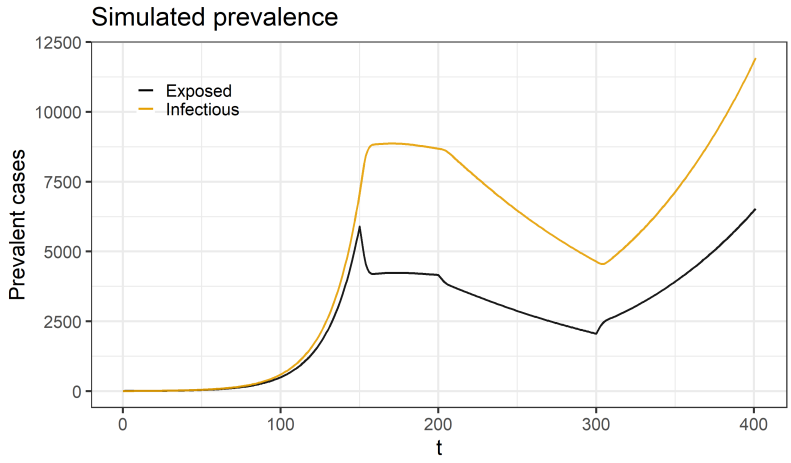

Figure 11: A plot of modeled prevalence over time. As all transitions aside from observations are deterministic, the number of infectious and exposed (non-infectious/in incubation period) individuals are also deterministic.

4.2.2 Smoothing and Estimation

[[*Avoid this sort of repetition.*]] The first step in the estimation algorithm is an initial filtered of the incidence data to reduce noise and periodicity. See the r(t) estimation section for more details.

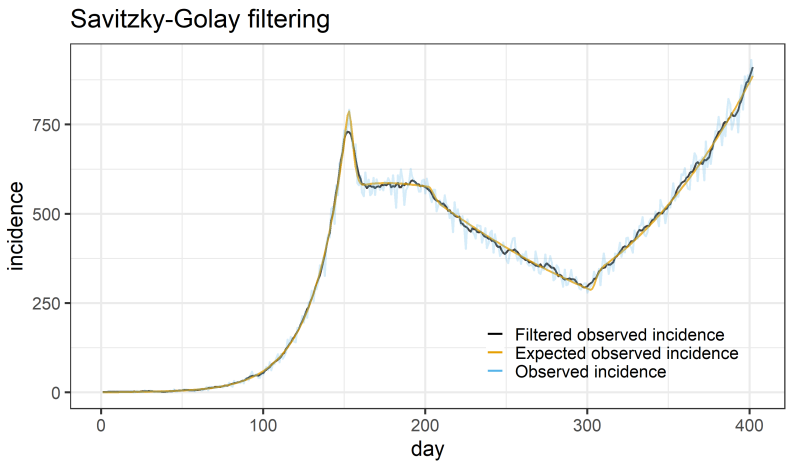

Figure 12: Application of a first-pass Savitzky-Golay filter substantially reduces noise and periodicity. The "expected observed" data is shown as a comparison, and is the incidence of infection forward-convolved by the mean infection-observation delay distribution.

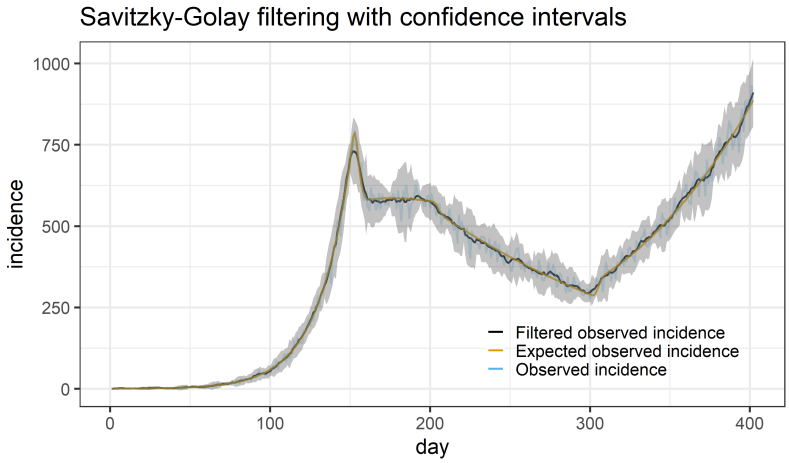

Figure 13: The same first-pass Savitzky-Golay filter as shown in Figure 12, with the confidence intervals shown.

Window sizes have been set to 7 and 15 days in the following figures. In the following figures, we see that this method is not capable of providing precise estimates immediately before and after sharp transitions (due to the filtering used), but can pinpoint the location of the transition.

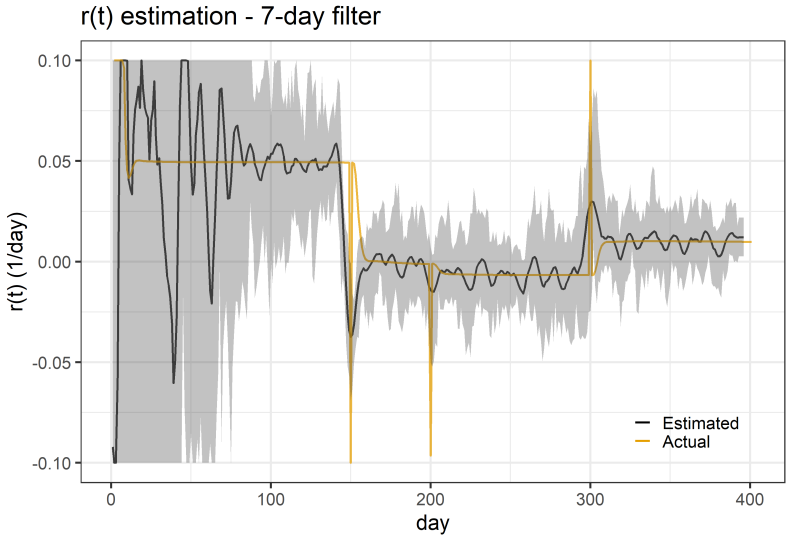

Figure 14: r(t) estimation using simulated data, with a 7-day post-filtering Savitzky-Golay window size. All values are clipped between -0.1 and 0.1 for visualization purposes. The true r(t) is within the 95% confidence interval 96% of the time.

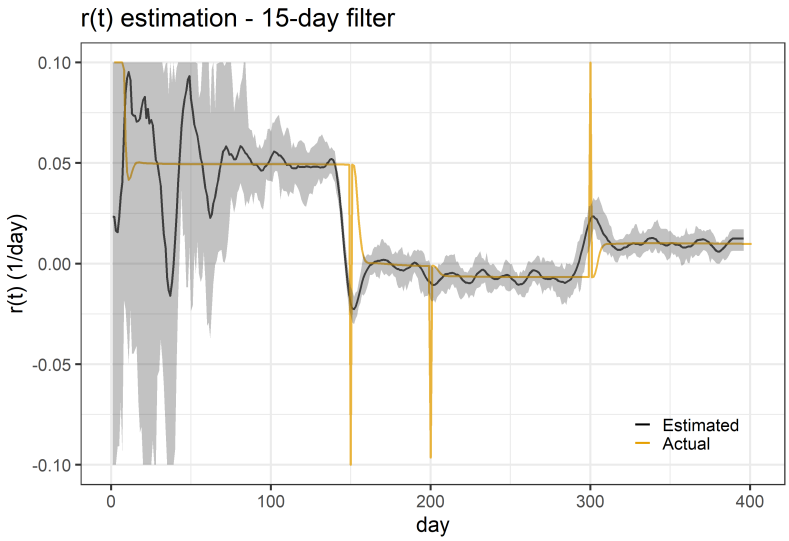

Figure 15: r(t) estimation using simulated data, with a 15-day post-filtering Savitzky-Golay window size. All values are clipped between -0.1 and 0.1 for visualization purposes. The true r(t) is within the 95% confidence interval 91% of the time.

4.3 Real-world data

While ground-truth r(t) values are unavailable for real-world data, this method can still be evaluated for qualitative stability using real-world data.

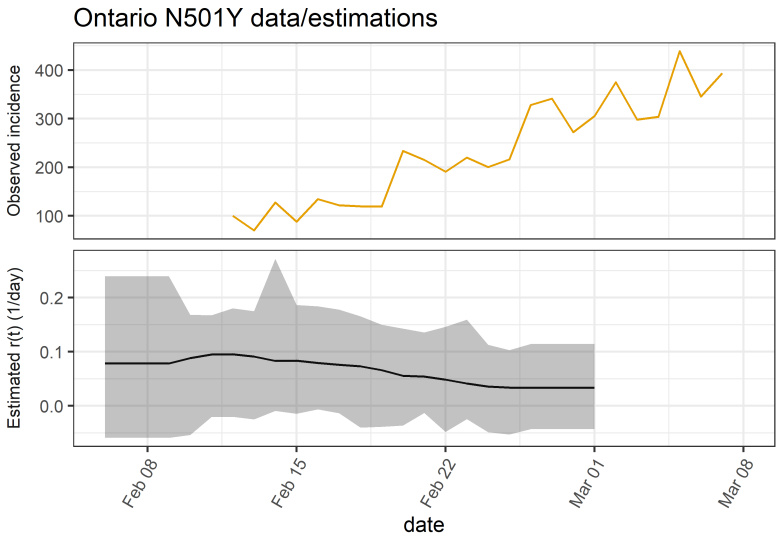

Figure 16: r(t) estimation using estimated N501Y variant data from Ontario, courtesy of Michael Li [9]. A 7-day post-filtering method and mean observation delay of 6 days was used.

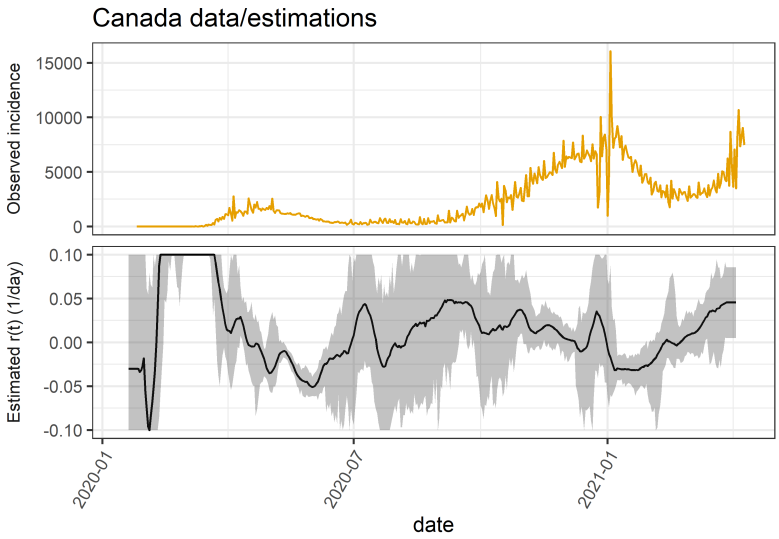

Figure 17: r(t) estimation using Canadian incidence data from OWID [1]. A 15-day post-filtering method and mean observation delay of 6 days was used. r(t) estimates are clipped between -0.1 and 0.1 for visualization purposes. (Note that during the period where estimates are cut off, the confidence interval is cut off as well.)

5 Conclusion

This work details an r(t) estimation method based on Savitzky-Golay filtering that provides reasonable estimates of r(t) on simulated data, which also effectively quantifies uncertainty. This work also presents an SEIR-based model with realistic dynamical features related to observation dynamics which can be useful for other parameter estimation tasks, such as $\mathcal{R}(t)$ estimation. The proposed r(t) parameter estimation method may be useful in the management of pandemics, but also in other fields where the exponential growth rate must be estimated from timeseries data. [[*Avoid vague comparisons.*]]

5.1 Code availability and use-cases

The code for this project is primarily written in R, with some minor code for handling real-world data written in Python. This project can be found at https://github.com/Apeirogons/rt_estimation_thesis. Please follow the instructions in the README to replicate this project. Among other things, code from this project can be used to:

- **Create a new discrete-time SEIR-based simulation with realistic observation dynamics. (ts_utils/deterministic_simulation.R)**

- **Perform a derivative order 0, polynomial order 1 Savitzky-Golay filtering with confidence interval estimates. (ts_utils/filter.R)**
- **Estimate the exponential growth rate from timeseries data using the previously described filtering-based approach. (ts_utils/filter.R and ts_utils/rt.R)**
- Perform regularized Richardson-Lucy deconvolution on timeseries using code modified from the Cobey lab. (ts_utils/rl_cobey.R) [7].
- Pull and split worldwide incidence data from OWID. (data_splitter.py) [1]
- Pull Ontario Variant of Concern timeseries data (variants_mli.R) [9].
- Easily create timeseries plots in ggplot2 with individual transparencies (create_plot in ggplot_params.R).
- Perform wavelet filtering of incidence data. (ts_utils/deconvolution_and_smoothing.py)

6 Appendix: Filtering

6.1 Rationale and figure explanations

Real-world data is noisy and periodic, and various smoothing methods were hypothesized to be effective at removing these effects. It was hoped that removing noise and periodic effects would improve the quality of $r(t)$ estimation. While Savitzky-Golay filtering is used in the main text, it is not the only type of filtering that may be applied to reduce noise in data. The next sections will detail several other filtering and smoothing methods that were explored in this project.

For each of the next figures, the *observed* curve is the number of observation events recorded per day. The *smoothed observed* curve is the 7-day smoothed *observed* curve, and the *expected* curve is the true incidence of infection scaled by `observationProb` convolved by `baseObservationDist`. We consider the *expected* curve to be the ground truth, although stochasticity can cause the number of observations to diverge from the expected curve.

6.2 7-Day Smoothing

A simple rolling mean is a simple method of smoothing, and is a special case of the Savitzky-Golay filter (window 7 days, polynomial order 0, derivative order 0).

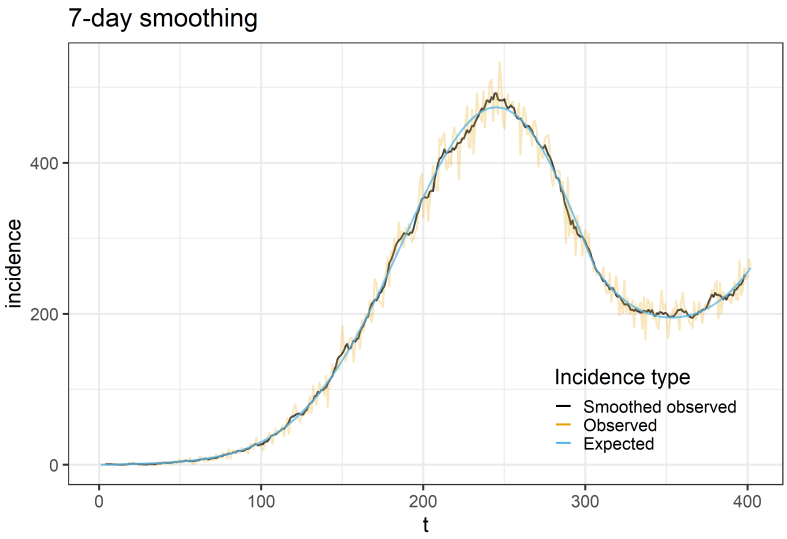

Figure 18: 7-day smoothing of simulated incidence. The smoothed observed curve is the smoothed version of the number of daily incidence observations, and the expected curve is the theoretical number of observations expected on a given day. See the beginning of this section for more detailed explanations.

6.3 Wavelet transform low-pass filter

The discrete wavelet transform converts a timeseries into the wavelet domain, for which there are high-frequency and low-frequency wavelets . Wavelets are periodic functions localized in space, allowing for modelling of periodicity and spatial features that can change over time. [10] All wavelets above a certain level (high-frequency wavelets) were set to zero in order to perform filtering, although this seems like a crude approach. This works relatively well for certain wavelet parameters, such as using the db4 wavelet and removing all levels above 3. This has the benefit of being highly flexible, allowing for various levels of smoothing and different choices of wavelets.

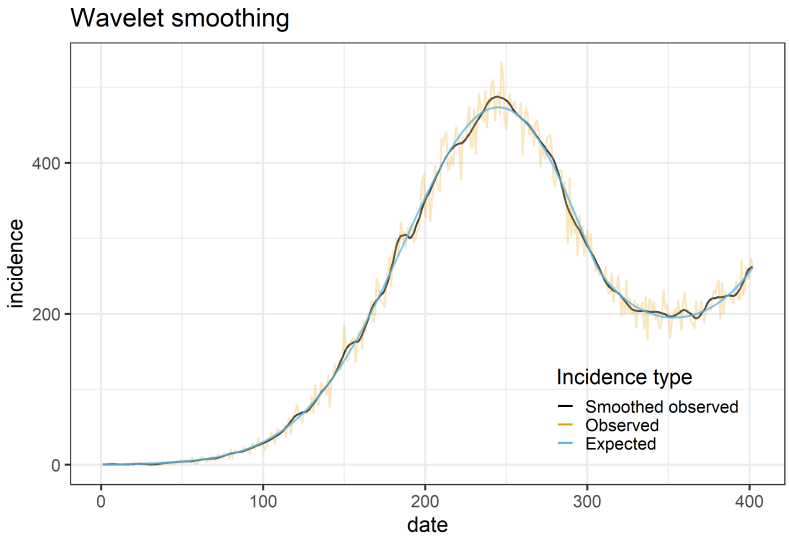

Figure 19: Wavelet filtering of simulated incidence. The smoothed observed curve is the filtered version of the number of daily incidence observations, and the expected curve is the theoretical number of observations expected on a given day. See the beginning of this section for more detailed explanations.

6.4 Butterworth low-pass filter

Like the wavelet transform, the Fourier transform converts a timeseries to the frequency domain. One can set all frequencies above a threshold to zero to perform filtering [11]. However, this suffers the unique issue of not being very accurate at the end of the timeseries, and has therefore been largely ignored for the rest of the project. The following filter is a Butterworth filter of order 5 and critical frequency of 0.1.

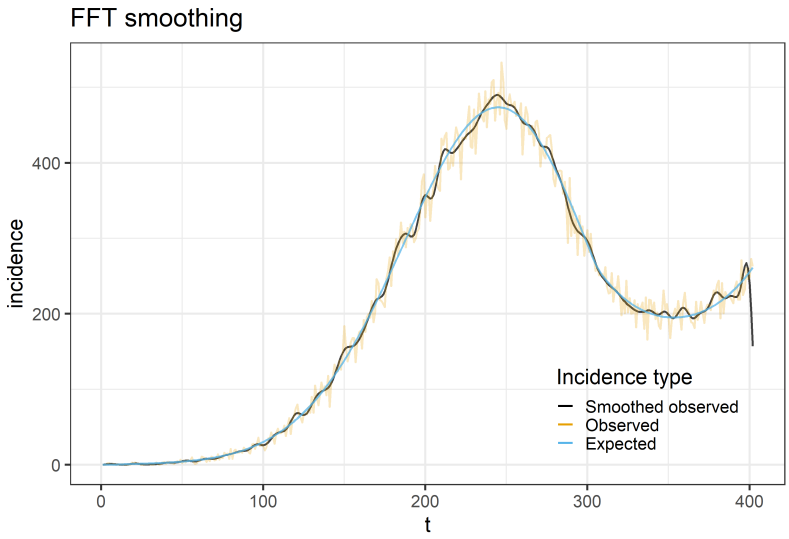

Figure 20: Butterworth filtering of simulated incidence. The smoothed observed curve is the filtered version of the number of daily incidence observations, and the expected curve is the theoretical number of observations expected on a given day. See the beginning of this section for more detailed explanations.

7 Appendix: Deconvolution

7.1 Rationale

Deconvolution is the inverse process of convolution. In the absence of noise and with a constant delay distribution, the observed incidence should be equal to the true incidence of infection curve convolved by the observation delay distribution, therefore blurring spikes in the true incidence of infection [7]. Therefore, in an ideal world, deconvolving the observed incidence curve would be more effective than shifting to obtain the true incidence of infection curve. However, in the real world and in the model used in this work, delay distributions may not be constant and observation noise may be present, a violation of both conditions. I investigated whether or not deconvolution would still be more effective than shifting for r(t) estimation given the presence of heavy noise in the data.

7.2 Methods

7.2.1 Introduction to deconvolution

The goal of deconvolution is to reconstruct an original image given an image convolved by some kernel, and the probability density function of this kernel. Richardson-Lucy deconvolution is a common iterative algorithm for performing

deconvolution by maximizing a Poisson log-likelihood [8]. For this work, an implementation of the Richardson-Lucy algorithm developed by the Cobey Lab for right-censored incidence timeseries was used [7].

7.2.2 Deconvolution priors

Unlike convolution, deconvolution does not have a unique solution. As a result, many deconvolution solutions are mathematically possible despite being biologically infeasible. As a result, different priors are used to obtain solutions that are considered feasible.

1. Early stopping. The baseline Richardson-Lucy algorithm does not converge in most cases to a "reasonable" solution, therefore, the iterative process is stopped when a "good enough" solution is found. In the Cobey Lab's code, this is accomplished using a χ^2 metric computed between the convolved reconstructed image, and the given image.

2. Regularization. We can enforce the prior that solutions should be "smooth" by adding a penalty on the absolute value of the derivative of the reconstructed image. I modify the original Cobey lab code to allow for this regularization.

7.2.3 Description of the Richardson-Lucy algorithm

Total variation-regularized Richardson-Lucy deconvolution is an iterative method following the equation below: [8]

$$o_{k+1} = \frac{i}{o_k * h} * (-h) \frac{o_k}{1 - \alpha div(\frac{\nabla o_k}{|\nabla o_k|})} \tag{1}$$

where o represents the deconvolved image, i represents the observed image, h represents the point-spread function, div is divergence, α is a regularization parameter, $\int_z$ refers to integrating over all pixels of the image, and $*$ is the convolution operator.

In the epidemiological context, o represents the estimated incidence of infection, i represents the observed incidence, h represents the observation delay distribution, div reduces to the 1-dimensional derivative, $\int_z$ refers to integrating over all times in the timeseries, and ∇ also reduces to the 1-dimensional derivative.

As the image consists of discrete timesteps, the derivative is approximated by:

1. Taking the first differences of the timeseries.

2. Copying the first and last first differences and appending them to the beginning and end of the timeseries.

3. Applying an uncentered rolling mean of length 2 to the resulting timeseries.

While this equation looks complex, it can be derived from a simpler principle: the minimization of the following loss function.

$$J(o) = \underbrace{\int_z ((h * o)(x) - i(x)log(h * o)(x))dx}_{\text{Poisson log-likelihood}} + \underbrace{\alpha \int_z |\nabla o(x)|dx}_{\text{Total Variation regularization}} \tag{2}$$

Note that other forms of regularization exist (for example, TM regularization), but have not been explored in this work.

7.3 Deconvolution results

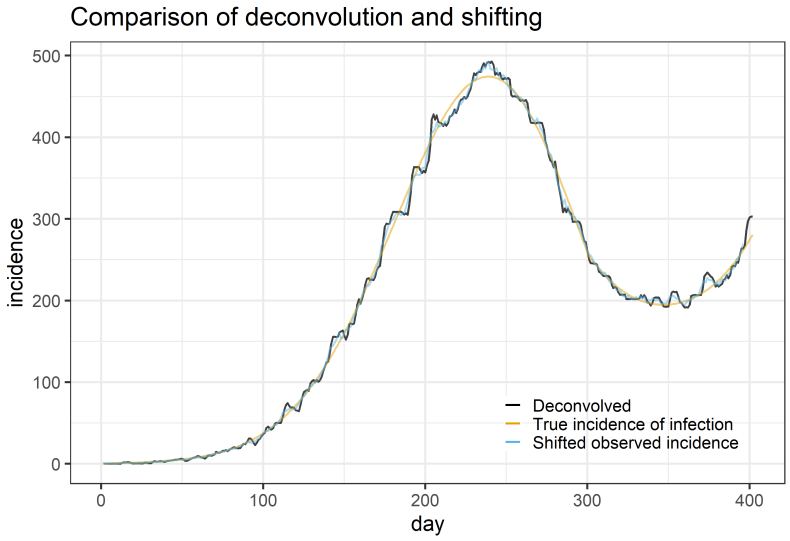

Figure 21: Comparison of deconvolution and shifting for reconstruction of true incidence of infection from filtered observation data.

Prior to deconvolution or shifting, a 7-day Savitzky-Golay filter was applied to the observation data. For this image, the regularization parameter was $\alpha = 0.001$ and 50 iterations of the deconvolution algorithm were applied. Note that qualitatively, the deconvolution seems to amplify noise.

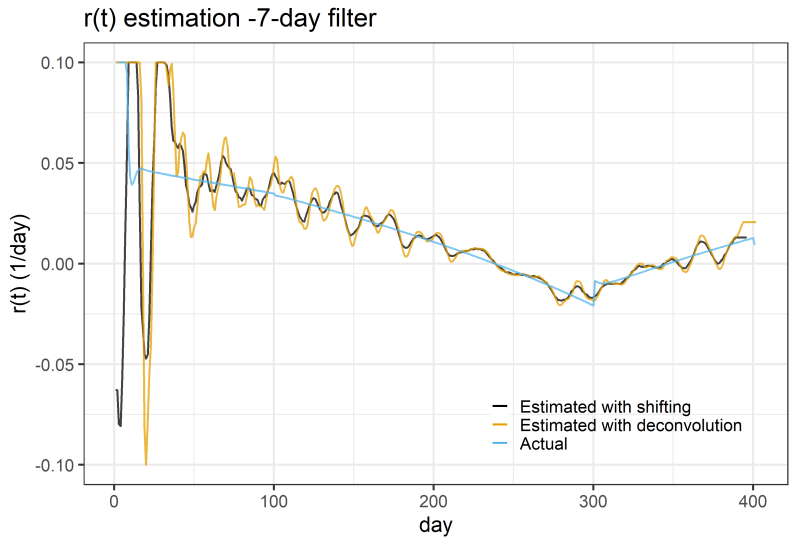

Figure 22: Comparison of deconvolution and shifting for r(t) estimation on simulated data with a 7-day post-smoothing window. Deconvolution clearly amplifies noise in the data, while not significantly sharpening any changes.

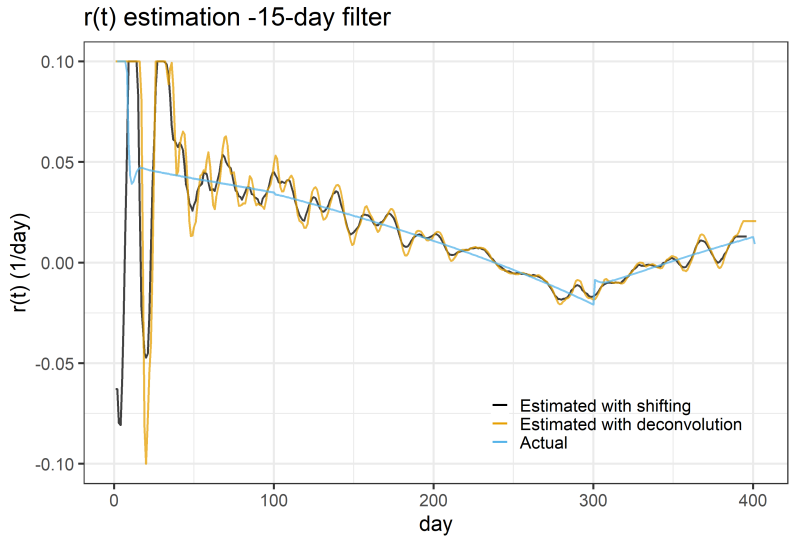

Figure 23: Comparison of deconvolution and shifting for r(t) estimation on simulated data with a 15-day post-smoothing window. Deconvolution clearly amplifies noise in the data, while not significantly sharpening any changes.

The previously described estimation algorithm was applied to simulated data. The "Estimated with shifting" curve was determined following the previously described method for r(t) estimation. The "Estimated with deconvolution" curve was determined by smoothing and deconvolving the observed incidence, and then applying the `estim` function with no shifting. Confidence intervals have not yet been implemented for r(t) estimation using deconvolution. It is our conclusion that deconvolution is not as effective as shifting for noisy incidence data for the purposes of r(t) estimation.

7.4 Other methods of deconvolution

7.4.1 Optimizer-based deconvolutions

Rather than using the Richardson-Lucy algorithm for deconvolution, one can also optimize the same underlying loss function using conventional optimizers. While this problem is too difficult for many gradient-based optimizers, it is possible to use the Powell (or potentially other non-gradient optimizers) for this task. One upside of this idea is that it is relatively easy to implement deconvolution based on arbitrary loss functions, for example, enabling the regularization term $\alpha \int_z |\nabla o(x)|^\phi dx$ to use an arbitrary value ϕ. In theory, this could be useful to implement deconvolution using new statistical models; for example, the negative binomial loglikelihood could be used in place of the Poisson log-likelihood. However, this idea was not pursued further in this work.

7.4.2 Wiener deconvolution

Wiener deconvolution performs deconvolution in the Fourier domain (deconvolutions are simple divisions in this domain), but adds an additional factor to the denominator. However, this suffered from being extremely inaccurate at the beginning and end of the timeseries, and was not pursued further in this work. See Appendix: Filtering - Butterworth low-pass filter for a similar example of poor trend-following at the edges of the timeseries. The implementation of this filter was provided in a public-domain Python script. [12]

8 Appendix: $\mathcal{R}(t)$ Estimation

8.1 Rationale

The proposed model can be used to evaluate other epidemiological parameter estimation methods. $\mathcal{R}(t)$ is the time-varying reproductive number, the mean number of infections each infectee will infect [7]. A common method of estimating the instantaneous $\mathcal{R}(t)$ is the method proposed by Cori et al [13], which will not be explained in detail in this work. However, it was initially hypothesized that filtering methods intended on reducing periodicity may improve Cori $\mathcal{R}(t)$ estimation. [[*Not clear what you mean by the proposed model here; the*

simulation model?]] The proposed model can be used to evaluate other epidemiological parameter estimation methods. $\mathcal{R}(t)$ is the time-varying reproductive number, the mean number of infections each infectee will infect [7]. A common method of estimating the instantaneous $\mathcal{R}(t)$ is the method proposed by Cori et al [13]. However, it was initially hypothesized that filtering methods intended on reducing periodicity may improve Cori $\mathcal{R}(t)$ estimation.

8.2 Evaluation of the Cori method on simulated data

The Cori method was used to estimate the instantaneous $\mathcal{R}(t)$ of simulated data (See Results for images of the simulations). Cori estimates using observations were shifted backwards by the mean observation delay. The timeseries was linearly extrapolated for 20 timesteps beyond the end of the timeseries based on the previous 50 observations to stabilize estimation near the end of the timeseries.

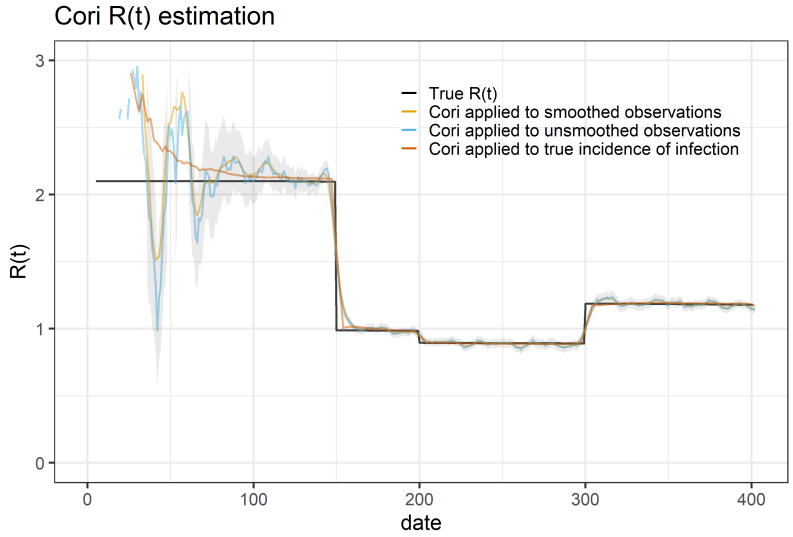

Figure 24: Cori estimation on simulated *smooth* data. After a warm-up period, all methods track the true $\mathcal{R}(t)$ exceptionally well. The Cori method applied to raw observation data is about the same as the Cori method applied to smoothed (7-day Savitzky-Golay filtered) data. The Cori method applied to the true incidence of infection fits the true $\mathcal{R}(t)$ nearly perfectly.

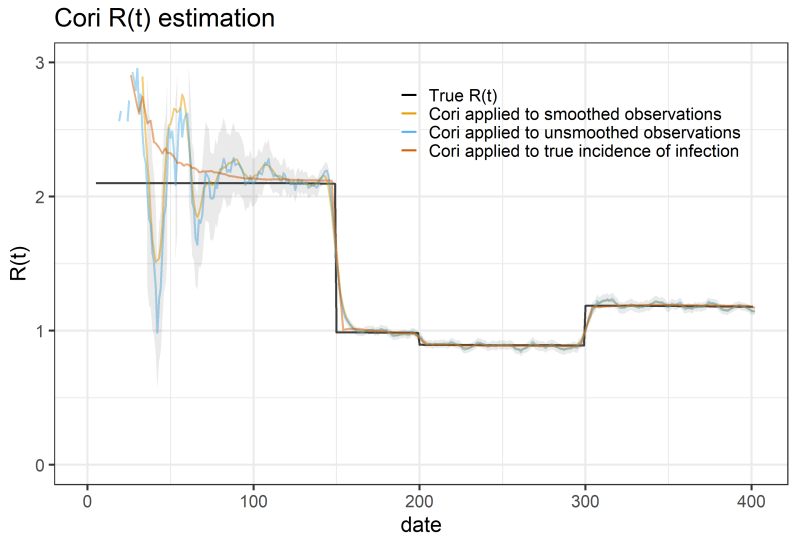

Figure 25: Cori estimation on simulated *blocky* data. After a warm-up period, all methods track the true $\mathcal{R}(t)$ exceptionally well. The Cori method applied to raw observation data is marginally less stable than the Cori method applied to smoothed (7-day Savitzky-Golay filtered) data. The Cori method applied to the true incidence of infection fits the true $\mathcal{R}(t)$ nearly perfectly.

Confidence intervals are the 95% confidence intervals of the Cori method applied to raw observation data.

8.3 Evaluation of *Shifts*

One hypothesis proposed by Jonathan Dushoff was that the Wallinga-Teunis method, designed for cohort $\mathcal{R}(t)$ estimation, could be naively applied to symptom onset data to estimate instantaneous $\mathcal{R}(t)$ [15].

Given that the case $\mathcal{R}(t)$ is equal to the instantaneous $\mathcal{R}(t)$ backwards-convolved by the generation interval, the generation interval = incubation period + infectious waiting time (which is equal to the infectious period for exponential infectious periods). Therefore, the case $\mathcal{R}(t)$ would be shifted backwards by incubation period + infectious period, while the symptomatic observations are shifted forwards by the incubation period - trailing the instantaneous $\mathcal{R}(t)$ by the infectious period.

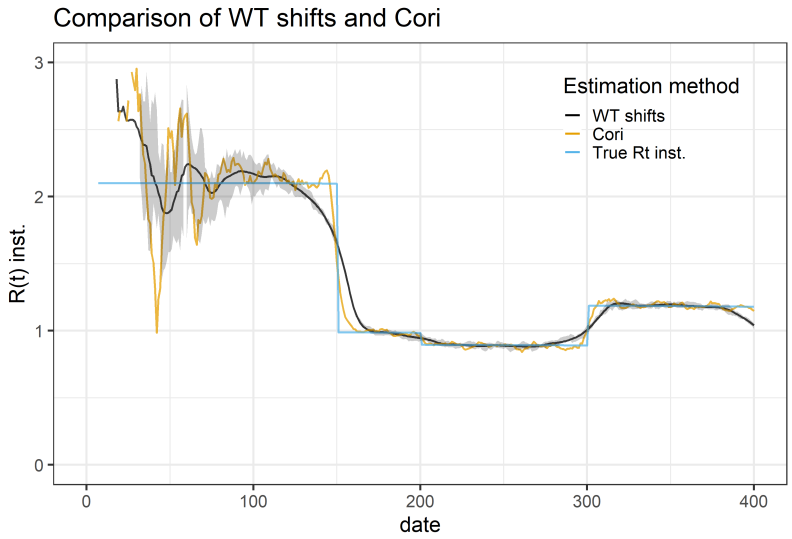

Figure 26: Comparison of the *shifts* method shifted forward by the mean infectious period, and the Cori method applied to observation data and shifted backwards. The Cori method tracks sharp changes in the true instantaneous $\mathcal{R}(t)$ much more effectively. The confidence interval shown is the *shifts* 95% confidence interval.

References

[1] Roser, M., Ritchie, H., Ortiz-Ospina, E., & Hasell, J. (2020). Coronavirus Pandemic (COVID-19). OurWorldInData.org. Retrieved 24 February 2021, from https://ourworldindata.org/coronavirus.

[2] McAloon, C., Collins, Á., Hunt, K., Barber, A., Byrne, A., & Butler, F. et al. (2020). Incubation period of COVID-19: a rapid systematic review and meta-analysis of observational research. BMJ Open, 10(8), e039652. https://doi.org/10.1136/bmjopen-2020-039652

[3] Byrne, A., McEvoy, D., Collins, A., Hunt, K., Casey, M., & Barber, A. et al. (2020). Inferred duration of infectious period of SARS-CoV-2: rapid scoping review and analysis of available evidence for asymptomatic and symptomatic COVID-19 cases. BMJ Open, 10(8), e039856. https://doi.org/10.1136/bmjopen-2020-039856

[4] R: The Negative Binomial Distribution. Stat.ethz.ch. (2021). Retrieved 24 February 2021, from https://stat.ethz.ch/R-manual/R-devel/library/stats/html/NegBinomial.html.

[5] Press, W., & Teukolsky, S. (1990). Savitzky-Golay Smoothing Filters. Computers In Physics, 4(6), 669. https://doi.org/10.1063/1.4822961

[6] scipy.signal.savgol_filter. docs.scipy.org. (2021). Retrieved 9 April 2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html.

[7] Gostic, K., McGough, L., Baskerville, E., Abbott, S., Joshi, K., & Tedijanto, C. et al. (2020). Practical considerations for measuring the effective reproductive number, Rt. PLOS Computational Biology, 16(12), e1008409. https://doi.org/10.1371/journal.pcbi.1008409

[8] Dey, N., Blanc-Féraud, L., Zimmer, C., Roux, P., Kam, Z., Olivo-Marin, J., & Zerubia, J. (2004). 3D Microscopy Deconvolution using Richardson-Lucy Algorithm with Total Variation Regularization. INRIA. Retrieved from https://hal.inria.fr/inria-00070726/document

[9] Li, M. (2021). COVID19-Canada. github.com. Retrieved 10 March 2021, from https://wzmli.github.io/COVID19-Canada/.

[10] Lee, G., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A Python package for wavelet analysis. Journal Of Open Source Software, 4(36), 1237. https://doi.org/10.21105/joss.01237

[11] Linear Circuit Design Handbook. (2008). https://doi.org/10.1016/b978-0-7506-8703-4.x0001-6

[12] Simple example of Wiener deconvolution in Python. github.com. Retrieved 11 April 2021, from https://gist.github.com/danstowell/f2d81a897df9e23cc1da.

[13] Cori, A., Ferguson, N., Fraser, C., & Cauchemez, S. (2013). A New Framework and Software to Estimate Time-Varying Reproduction Numbers During Epidemics. American Journal Of Epidemiology, 178(9), 1505-1512. https://doi.org/10.1093/aje/kwt133

[14] Wallinga, J. (2004). Different Epidemic Curves for Severe Acute Respiratory Syndrome Reveal Similar Impacts of Control Measures. American Journal Of Epidemiology, 160(6), 509-516. https://doi.org/10.1093/aje/kwh255

[15] Dushoff, J. (2021). R(t). Dushoff.github.io. Retrieved 11 April 2021, from http://dushoff.github.io/notebook/shifts.html.