Supplementary details on methodology and experiments of effective reproduction number estimation with trend filtering

Jiaping Liu, Zhenglun Cai, Paul Gustafson, and Daniel J. McDonald

2023-11-20

1 Numerical optimization of the $\mathcal{R}_t$ estimator

Recall that the Poisson trend filtering problem is

$$\widehat{\mathcal{R}} = \exp(\widehat{\theta}) \quad \text{where} \quad \widehat{\theta} = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \, \eta^\mathsf{T} \exp(\theta) - \mathbf{y}^\mathsf{T}\theta + \lambda \left\| D^{(k+1)}\theta \right\|_1 .$$

The proximal Newton method is a second-order algorithm solving a proximal optimization iteratively followed by a line search algorithm adjusting the step size at each iteration for faster convergence. The proximal Newton method for Poisson trend filtering solves an approximate problem iteratively — specifically, it takes a second-order Taylor expansion of the Poisson loss, which results in a proximal optimization, i.e., trend filtering with squared ℓ_2 loss, with dynamic weights during iteration, and solves it iteratively until convergence to the objective.

Let $g(\theta) = \eta^\mathsf{T} \exp(\theta) - \mathbf{y}^\mathsf{T}\theta$ be the Poisson loss and $h(\theta) = \lambda \|D^{(k+1)}\theta\|_1$ be the regularization. At iterate $j+1$, consider the following approximation of $g(\theta)$ using the second-order Taylor expansion around θ^j,

$$g(\theta) = g(\theta^j) + (\theta - \theta^j)^\mathsf{T} \nabla_\theta^{(1)} g(\theta^j) + \frac{1}{2}(\theta - \theta^j)^\mathsf{T} \nabla_\theta^{(2)} g(\theta^j)(\theta - \theta^j),$$

where $\nabla_\theta^{(1)} g(\theta^j) = \frac{1}{n}\eta^\mathsf{T} \exp(\theta^j) - y \in \mathbb{R}^n$ is the gradient of $g(\theta)$ evaluated at θ^j, and $\nabla_\theta^{(2)} g(\theta^j) = \frac{1}{n} \operatorname{diag}\left(\eta \circ \exp(\theta^j)\right) \in \mathbb{R}^{n \times n}$ is the Hessian matrix of $g(\theta)$ evaluated at θ^j and $\circ$ means elementwise product. Define the proximal operator as $\operatorname{prox}_{W,D}(\mathbf{x}) := \underset{\mathbf{z} \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2n} \|\mathbf{z} - \mathbf{x}\|_W^2 + \lambda \|D\theta\|_1$, where $\|\mathbf{a}\|_W^2 := \mathbf{a}^\mathsf{T} W \mathbf{a}$. The proximal optimization problem at iterate $j+1$ given θ^j becomes

$$\begin{aligned} \theta^{j+} :&= \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \, (\theta - \theta^j)^\mathsf{T} \nabla_\theta^{(1)} g(\theta^j) + \frac{1}{2}(\theta - \theta^j)^\mathsf{T} \nabla_\theta^{(2)} g(\theta^j)(\theta - \theta^j) + h(\theta), \\ &= \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \, \frac{1}{2n} \left\|\theta - \mathbf{c}^j\right\|_{W^j}^2 + \lambda \left\| D^{(k+1)}\theta \right\|_1 , \\ &= \operatorname{prox}_{W^j, D^{(k+1)}}(\mathbf{c}^j), \end{aligned} \tag{1}$$

where $W^j := \operatorname{diag}\left(\eta \circ \exp(\theta^j)\right)$ is the weighted (Hessian) matrix multiplied by n and

$$\mathbf{c}^j := \theta^j - n \left(W^j\right)^{-1} \nabla_\theta^{(1)} g(\theta^j) = \mathbf{y} \circ \eta^{-1} \circ \exp(-\theta^j) - \mathbf{1} + \theta^j \circ \eta^{-1}.$$

This is just univariate Gaussian trend filtering with weights W^t [Tibshirani(2014)].

We solve the trend filtering problem in Equation (1) using the specialized ADMM, proposed by [Ramdas and Tibshirani(2016)], with the primal θ step solved in closed-form and the auxiliary step solved by the dynamic programming algorithm for fused lasso proposed by [Johnson(2013)]. Let the auxiliary variable $\mathbf{z} := D^{(k)}\theta$. The scaled augmented Lagrangian is

$$\mathcal{L}_{\lambda,\rho}(\theta, \mathbf{z}, \mathbf{u}) = \frac{1}{2n} \left\|\theta - \mathbf{c}^j\right\|_{W^j}^2 + \lambda \left\|D^{(1)}\mathbf{z}\right\|_1 + \frac{\rho}{2} \left\|D^{(k)}\theta - \mathbf{z} + \mathbf{u}\right\|^2 - \frac{\rho}{2} \|\mathbf{u}\|^2,$$

where ρ is a scaled dual parameter and $\mathbf{u}$ is the dual variable. At the $(j+1)^{\text{th}}$ Newton step, the specialized ADMM solves the following subproblems, at ADMM iteration $l+1$:

$$\begin{aligned}
\theta^{l+1} &:= \underset{\theta}{\operatorname{argmin}} \frac{1}{2n} \left\|\theta - \mathbf{c}^l\right\|_{W^j}^2 + \frac{\rho}{2} \left\|D^{(k+1)}\theta - \mathbf{z}^l + \mathbf{u}^l\right\|_2^2, \\
\mathbf{z}^{l+1} &:= \underset{\mathbf{z}}{\operatorname{argmin}} \frac{\lambda}{\rho} \left\|D^{(1)}\mathbf{z}\right\|_1 + \frac{1}{2} \left\|D^{(k+1)}\theta^{l+1} - \mathbf{z} + \mathbf{u}^l\right\|_2^2, \\
\mathbf{u}^{l+1} &\leftarrow \mathbf{u}^l + D^{(k+1)}\theta^{l+1} - \mathbf{z}^{l+1}.
\end{aligned} \tag{2}$$

Finally, the step size $\gamma^{j+1} \in (0, 1]$ at iterate $j+1$ is adjusted by a backtracking line search algorithm to solve for θ^{j+1}, i.e.,

$$\theta^{j+1} \leftarrow \theta^j + \gamma^{j+1}(\theta^{j+} - \theta^j).$$

The proximal Newton algorithm iterates until convergence of the objective.

For $k = 0$, the trend filtering penalty is equivalent to fused LASSO. In this case, we use Johnson's dynamic programming algorithm [Johnson(2013)], which solves effective reproduction number of our model exactly and in linear time.

2 Experimental design

2.1 Compute λ_{max}

Let $\alpha := D\theta$, and then $\theta = D^\dagger\alpha$, where $D^\dagger D = I_n$. The PTF objective to be minimized can be written correspondingly as

$$\mathcal{L}_\lambda(\alpha) := \eta^\mathsf{T} \exp\left(D^\dagger\alpha\right) - \mathbf{y}^\mathsf{T} D^\dagger\alpha + \lambda \|\alpha\|_1.$$

Solve the following problem $\lambda_{max} := \sup_{\alpha \neq \mathbf{0}} \lambda$ through solving $\sup_\alpha \frac{\partial \mathcal{L}_\lambda(\alpha)}{\pm \partial \alpha_i} = \sup_{\alpha_i} \left(D^\dagger\right)^\mathsf{T} \left(e^{D^\dagger\alpha}\eta - y\right) = \max_i \left|\left(D^\dagger\right)^\mathsf{T}(\eta - y)\right|_i$ as $D^\dagger\alpha \to \mathbf{0}$ for $i = 1, \cdots, n$.

2.2 Cross validation of RtEstim in simulation

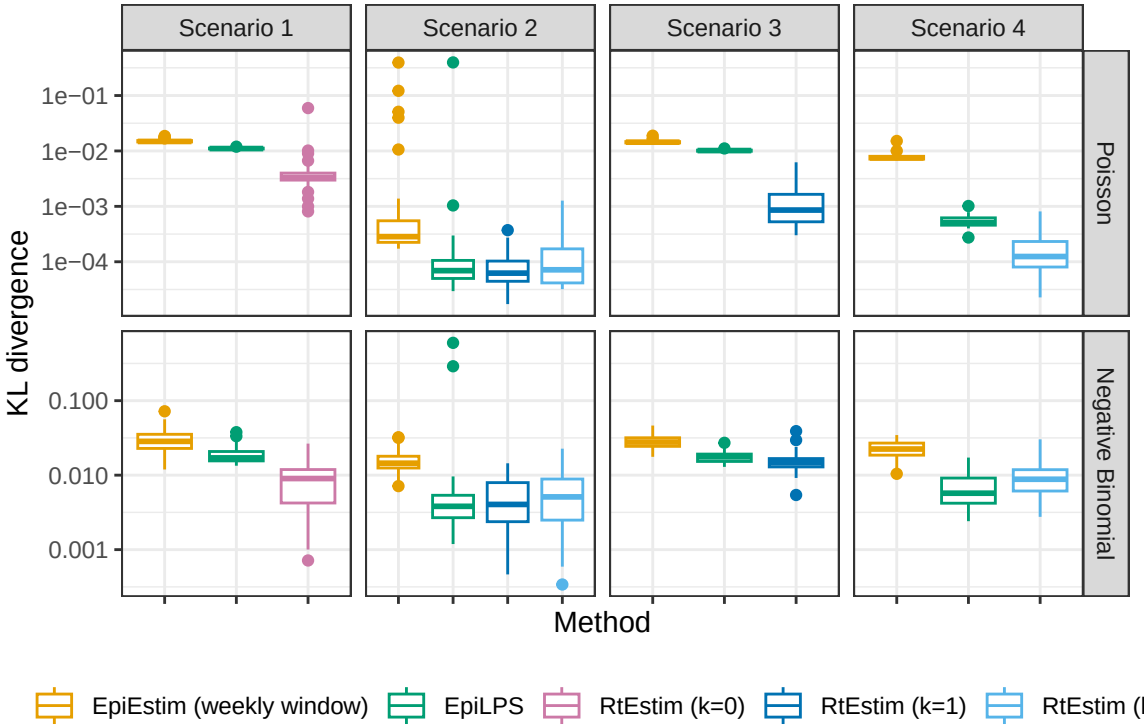
We run 10-fold cross validation (CV) to choose the best tuning parameter from the candidate set of size 50, i.e., $\boldsymbol{\lambda} = \{\lambda_1, \cdots, \lambda_{50}\}$. Specifically, we divide the all samples (except the first and last entries) into ten folds evenly and randomly, and build models on each sample set by leaving a fold out across all hyperparameters. We select the tuning parameter that gives the lowest averaged ***deviance*** between the estimated reproduction

numbers and the observed samples averaged over all folds.

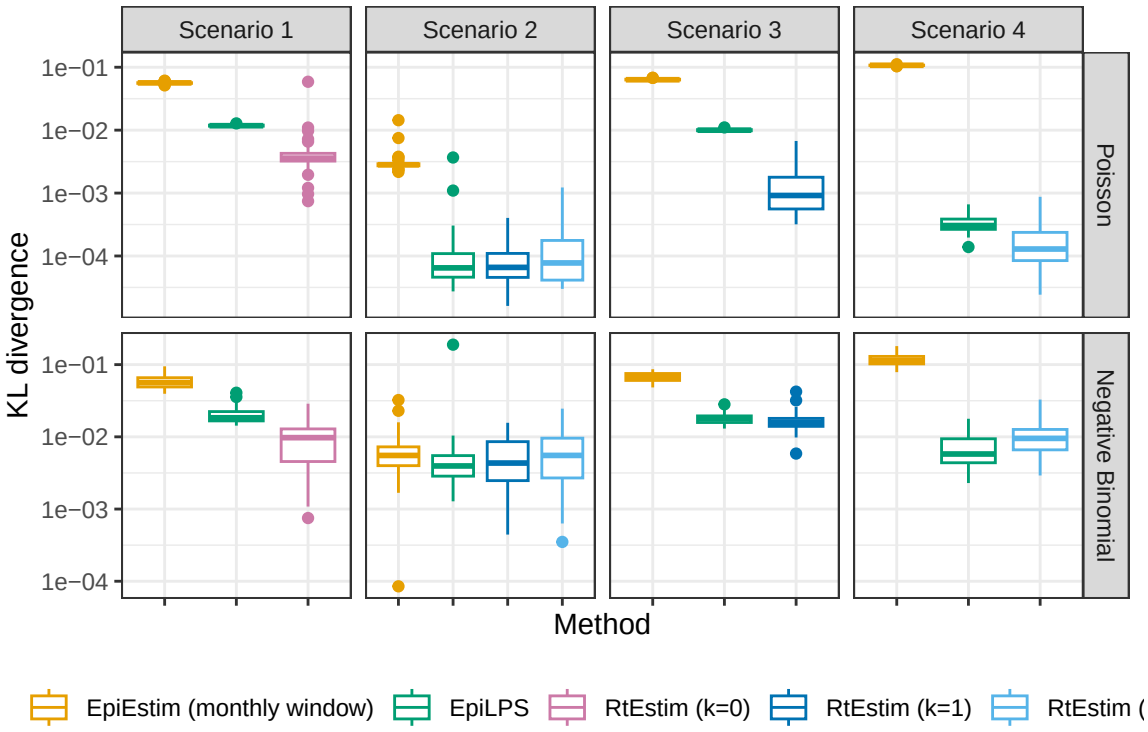
2.3 Supplementary results of experiments

The full KL divergence values for Poisson and negative Binomial incidence without excluding outliers are shown in the following figures. Both y-axes are in logarithmic scales for a better visualization, since there are extreme values. The baseline KL values is the rolling average of 7 past days of the true Rt values for comparison of weekly sliding windows, and 30 past days for monthly windows.

Compare EpiLPS, RtEstim and EpiEstim with ***weekly*** sliding windows. KL values computation excludes the first week of Rt estimates for all approaches.



Compare EpiLPS, RtEstim and EpiEstim with ***monthly*** sliding windows. KL values computation excludes the first month of Rt estimates for all approaches.

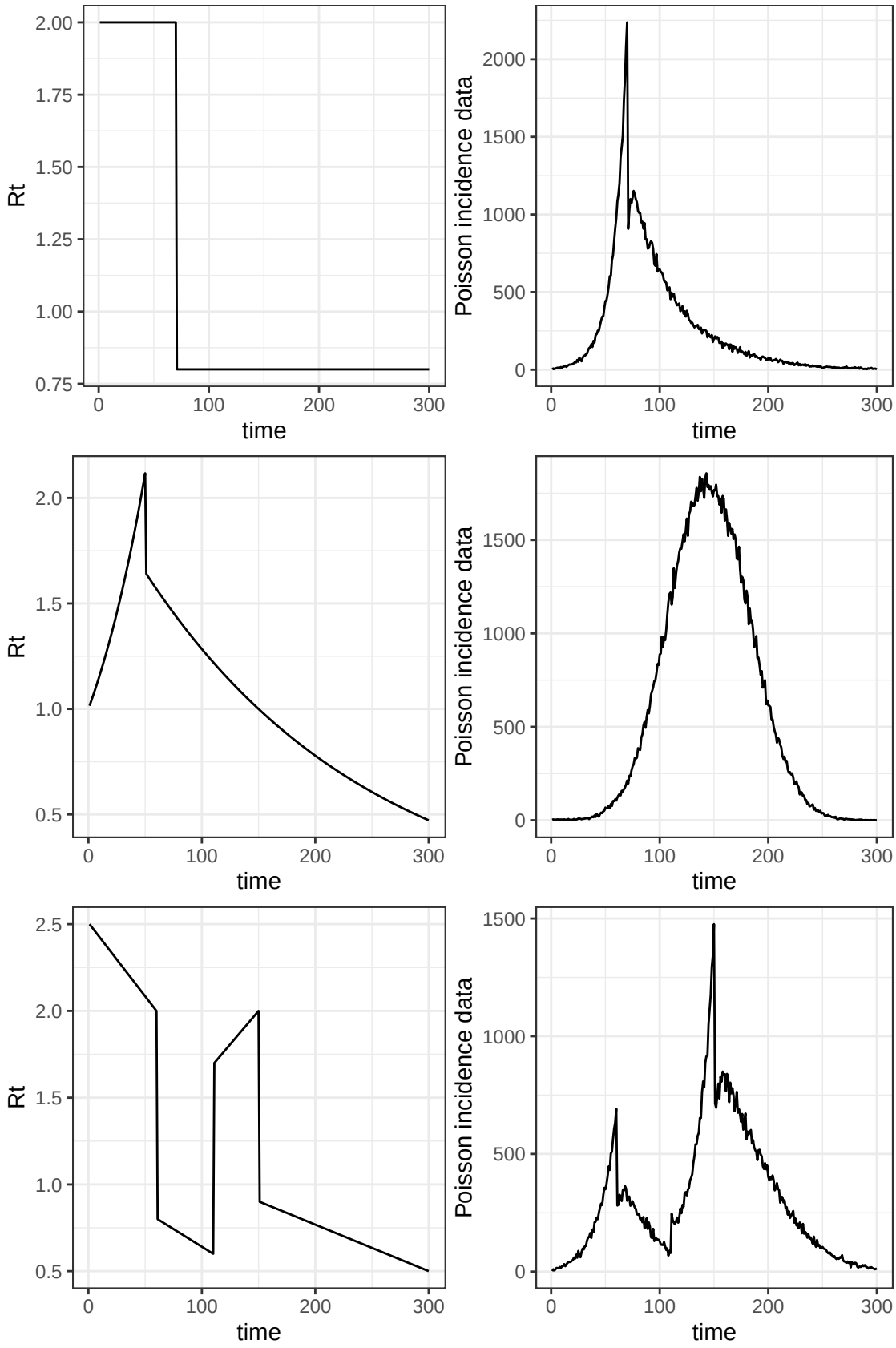


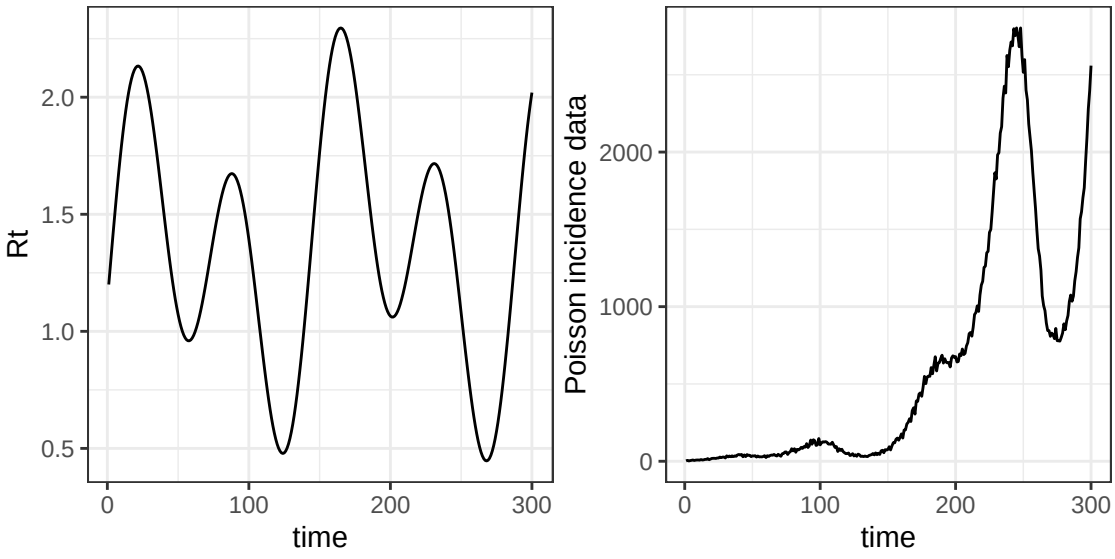
2.4 Synthetic effective reproduction numbers and Poisson incidence examples

Overall, we argue our estimator is accurate, robust in model misspecification and computationally efficient. We can do a series of tests for each property. We may consider the following curvature of efficient reproduction numbers, and test the accuracy of our estimators compared to EpiEstim and EpiLPS.

We may consider arbitrary reproduction numbers in a few scenarios: a) piecewise-constant epidemics with a drop at a certain time point to measure the effect of control measures, b) exponentially rising and falling epidemics with a change point, c) piecewise-constant with multiple segments to measure the initially controlled and resurged and the suppressed epidemics, d) periodic waves.

We may simulate the epidemics (with length T=300) 10 times for each scenario, estimate Rt, and compute the averaged KL divergence.


Rt
time
Poisson incidence data
time
Rt
time
Poisson incidence data
time
Rt
time
Poisson incidence data
time



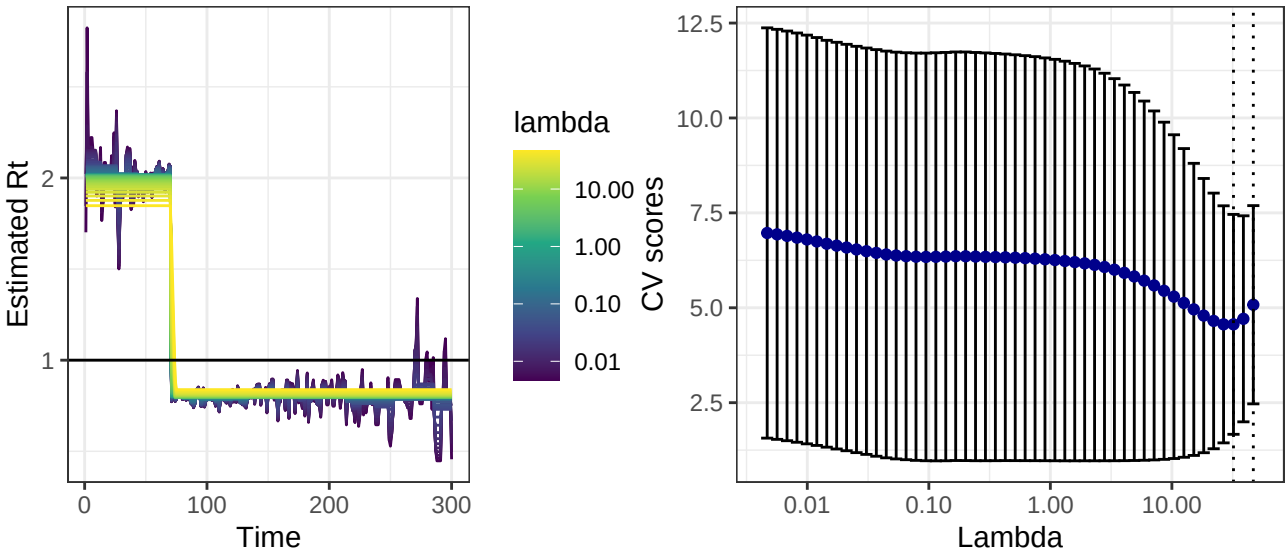
2.4.1 Serial interval distributions

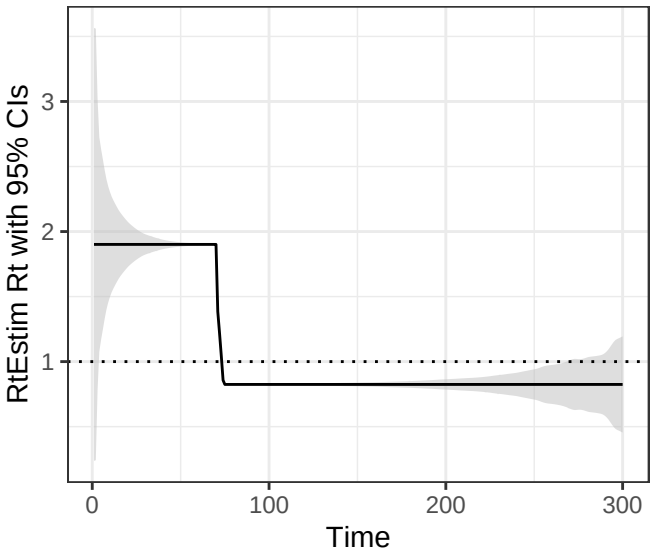
Parameters of the serial interval distribution, i.e., shapes and scale/rates of Gamma distribution, can significantly influence the peak values of incidence and the smoothness of incidence curves. Here is a comparison of the densities of Gamma distribution with shape 2.5 and scale 2.5 and Gamma distribution with shape 5 and scale 5.

2.4.2 Rt estimates for Poisson incidence using RtEstim

Fit our poisson trend filtering using `RtEstim`. See an example of the first scenario. We first see an demonstration of estimates using 50 hyperparameters, and use cross validation to choose the "best" hyperparameter with the lowest score. The last plot of this scenario displays the Rt estimates using the chosen hyperparameter.

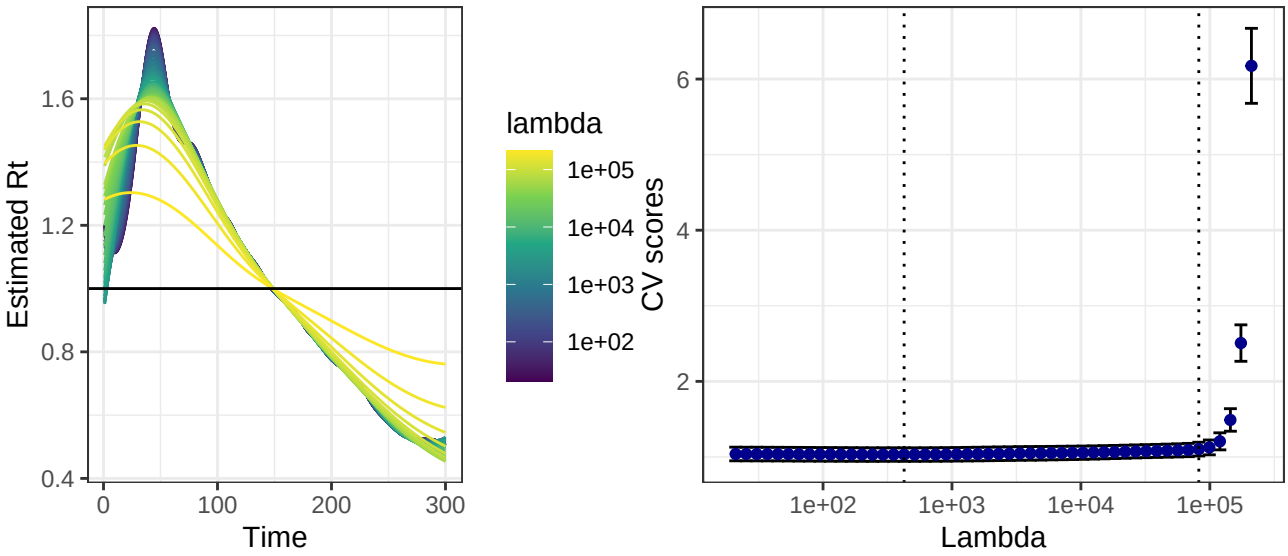
```
## [1] TRUE
```

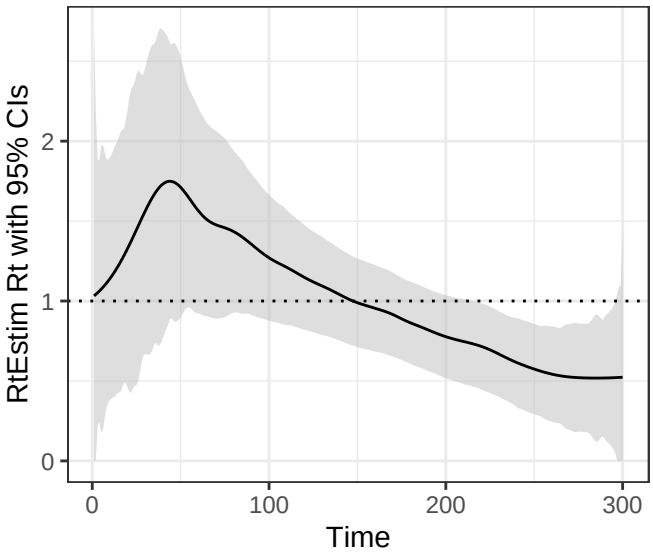




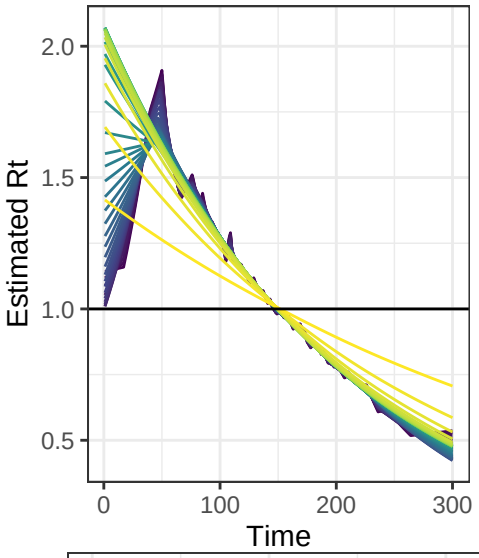
The Rt estimates for the other three scenarios are given in the following figures.

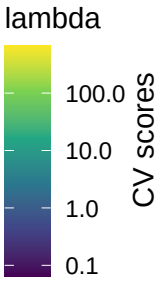
```
## [1] TRUE
```

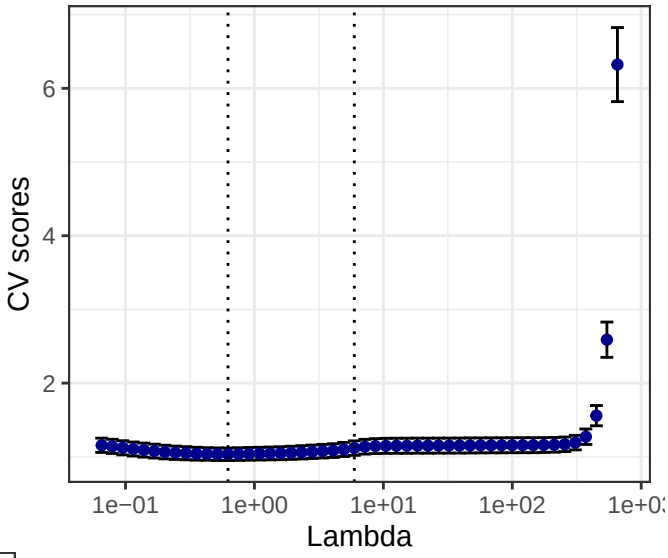


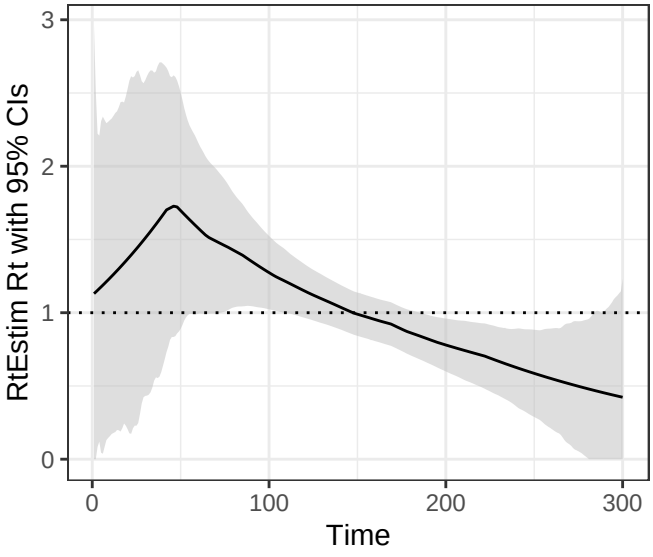

RtEstim Rt with 95% CIs
0
1
2
0
100
200
300
Time

```
## [1] TRUE
```

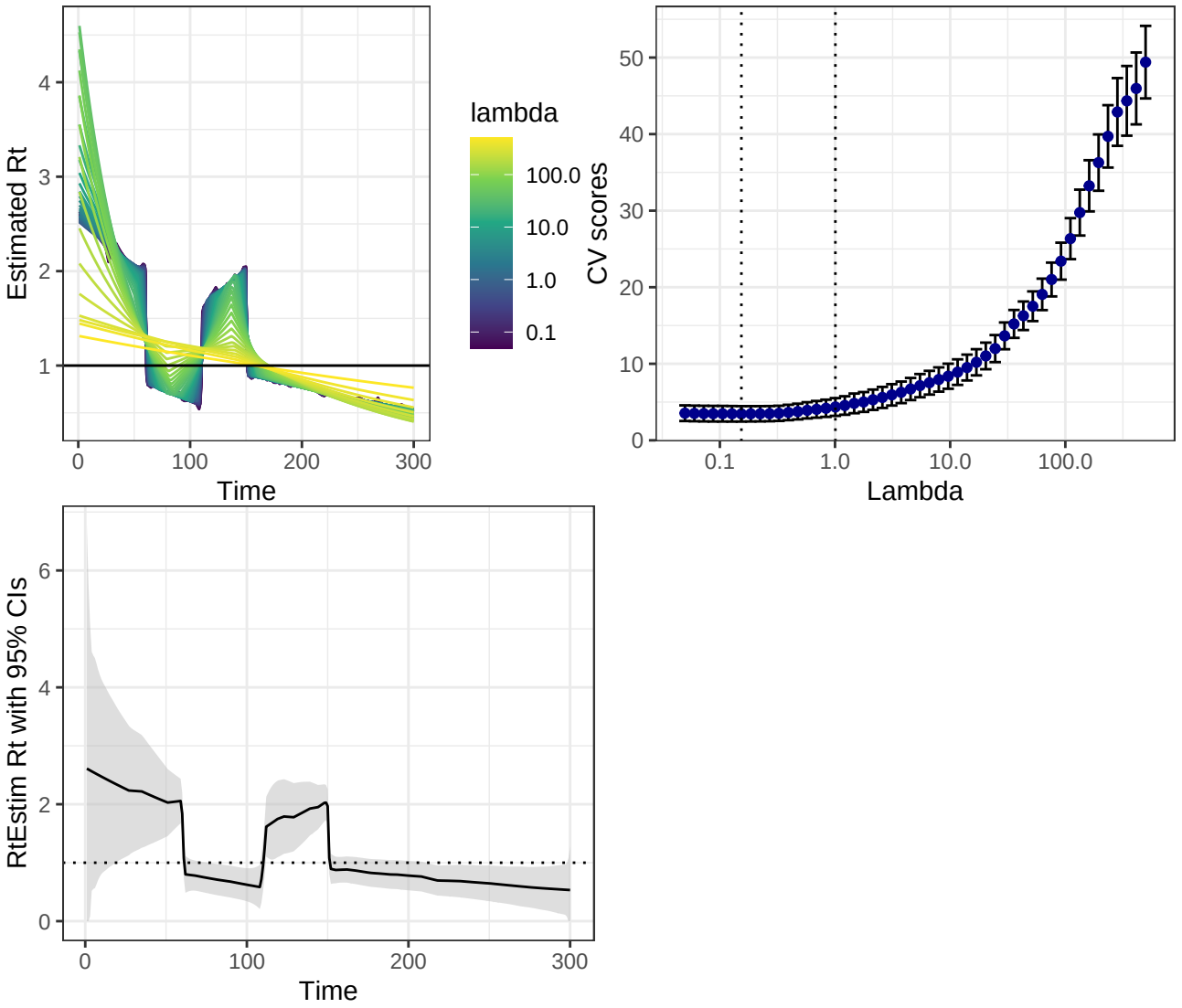

Estimated Rt
0.5
1.0
1.5
2.0
0
100
200
300
Time


lambda
100.0
10.0
1.0
0.1

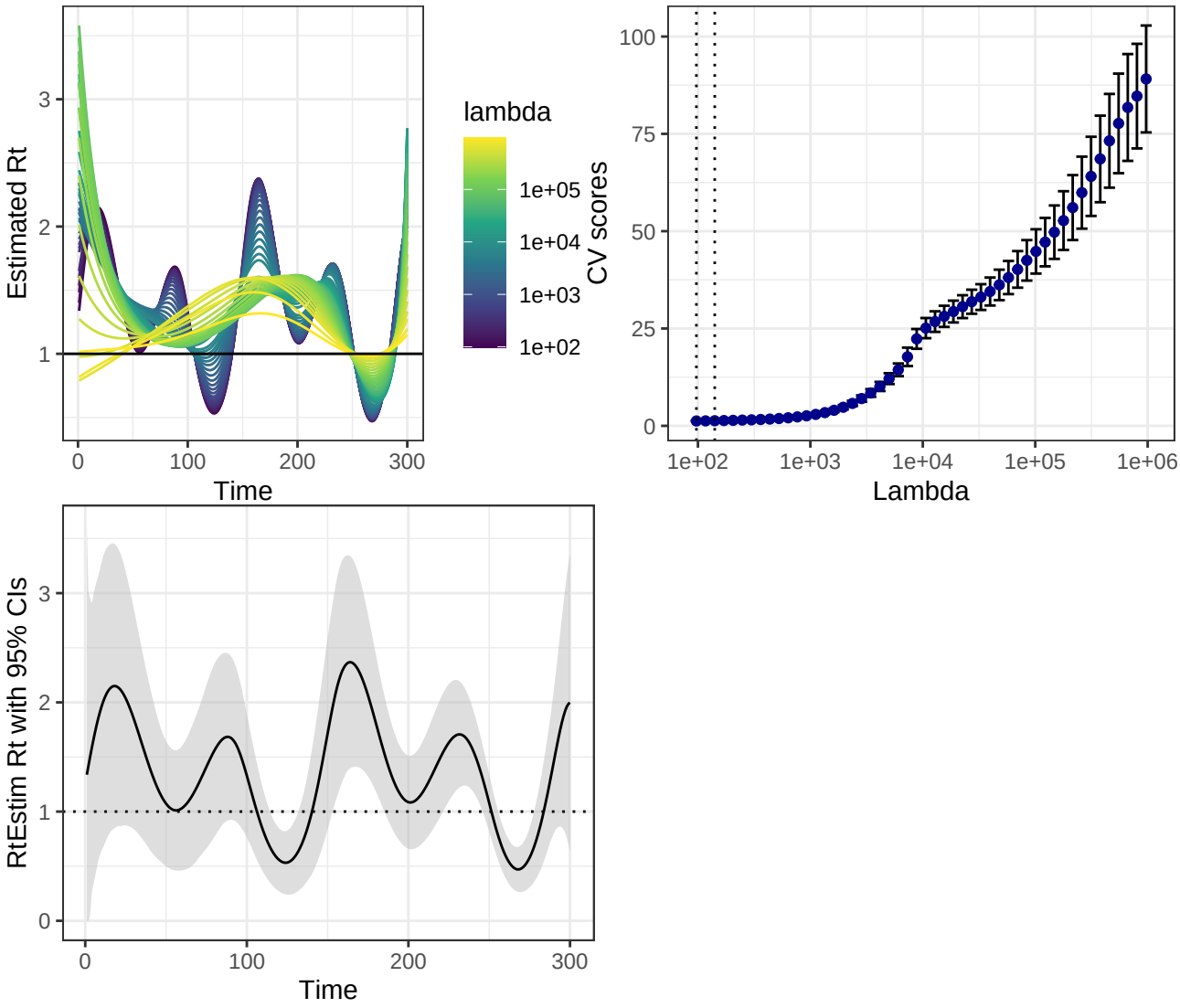

CV scores
2
4
6
1e−01
1e+00
1e+01
1e+02
Lambda


RtEstim Rt with 95% CIs
0
1
2
3
0
100
200
300
Time

```
## [1] TRUE
```


Estimated Rt
Time
lambda
100.0
10.0
1.0
0.1
CV scores
Lambda
RtEstim Rt with 95% CIs
Time

```
## [1] TRUE
```



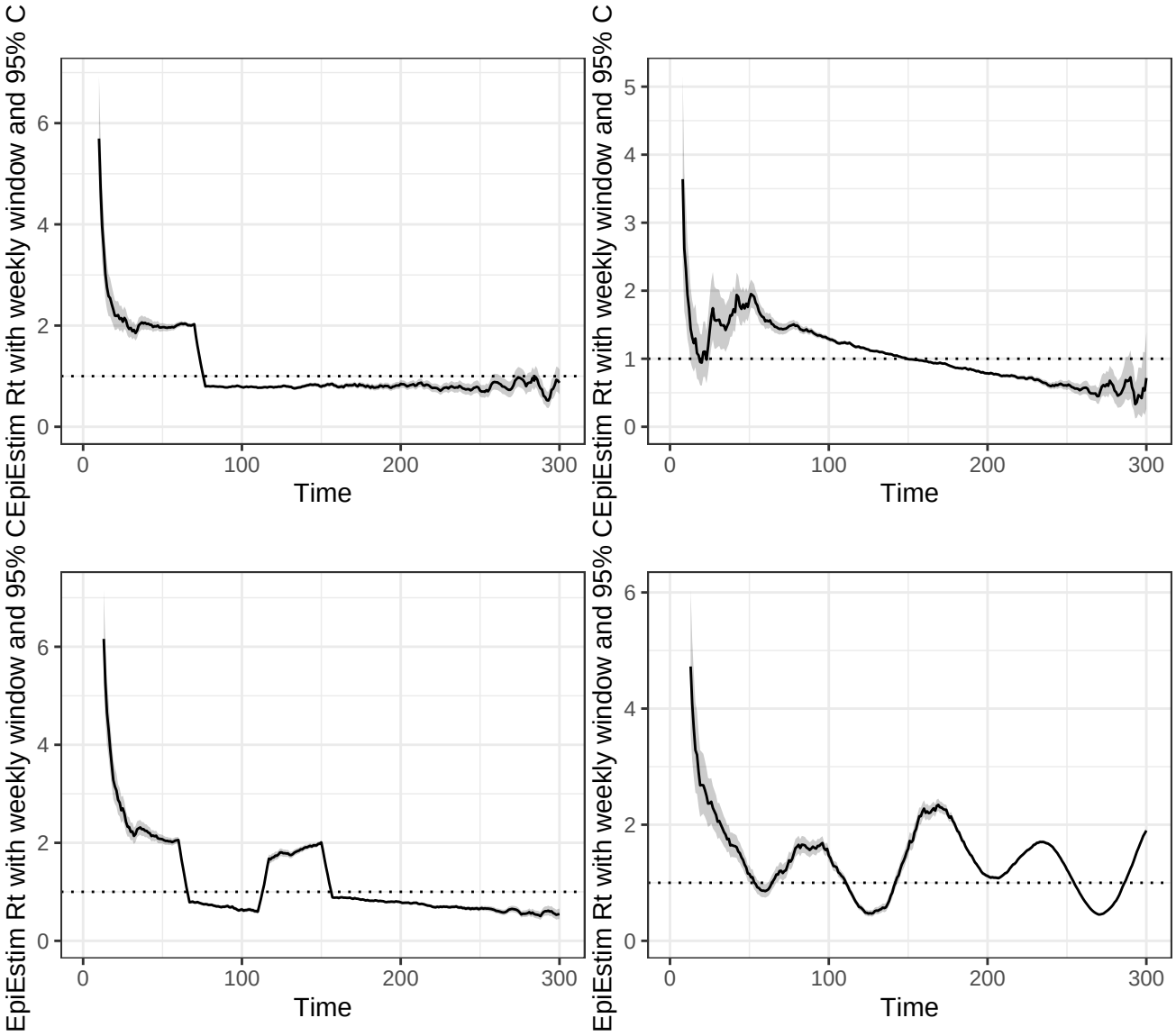
2.4.3 Rt estimates for Poisson incidence using EpiEstim

Fit EpiEstim with weekly sliding windows using "true" serial interval.

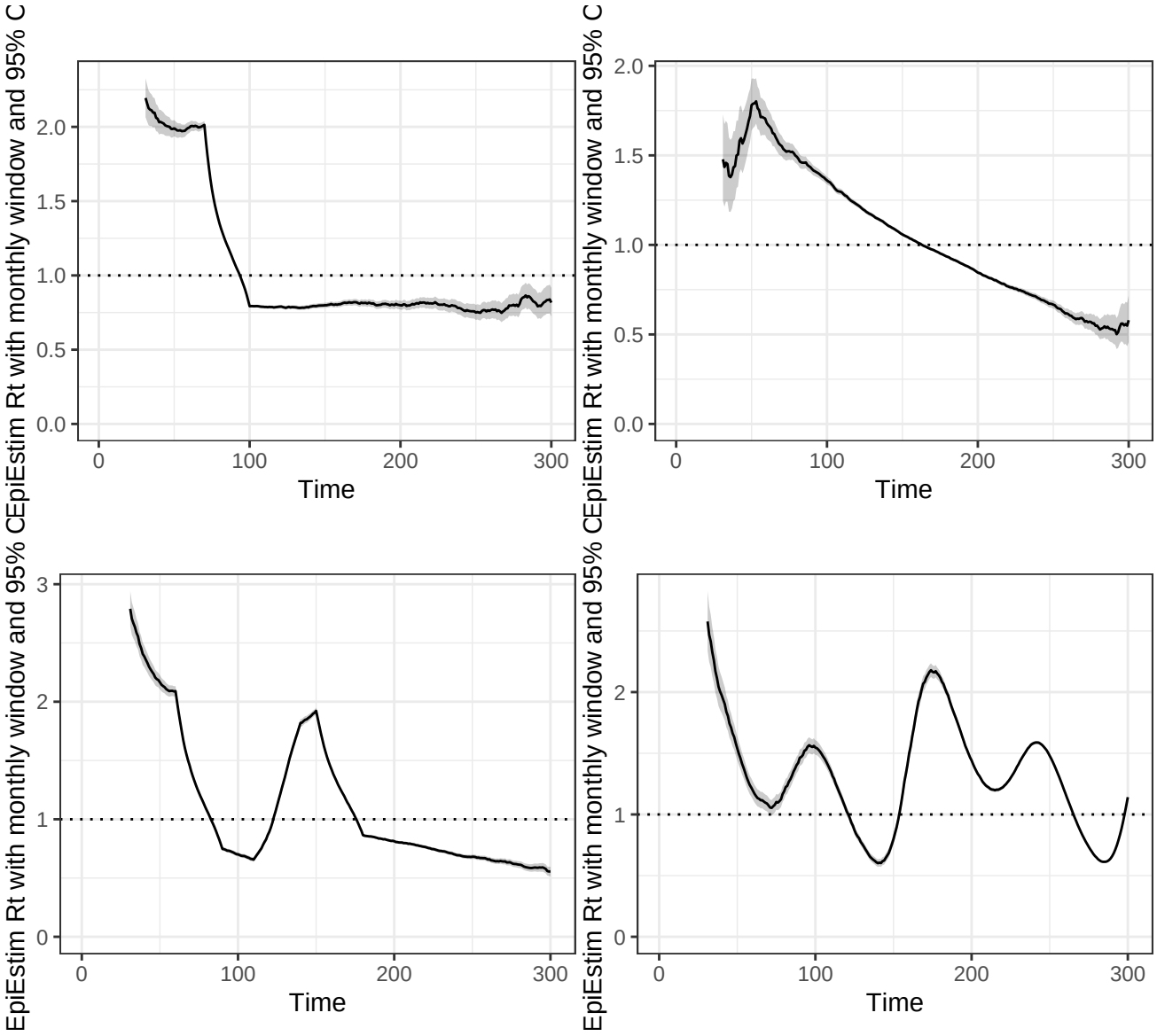
```
## [1] TRUE
```

```
## [1] TRUE
```

```
## [1] TRUE
```

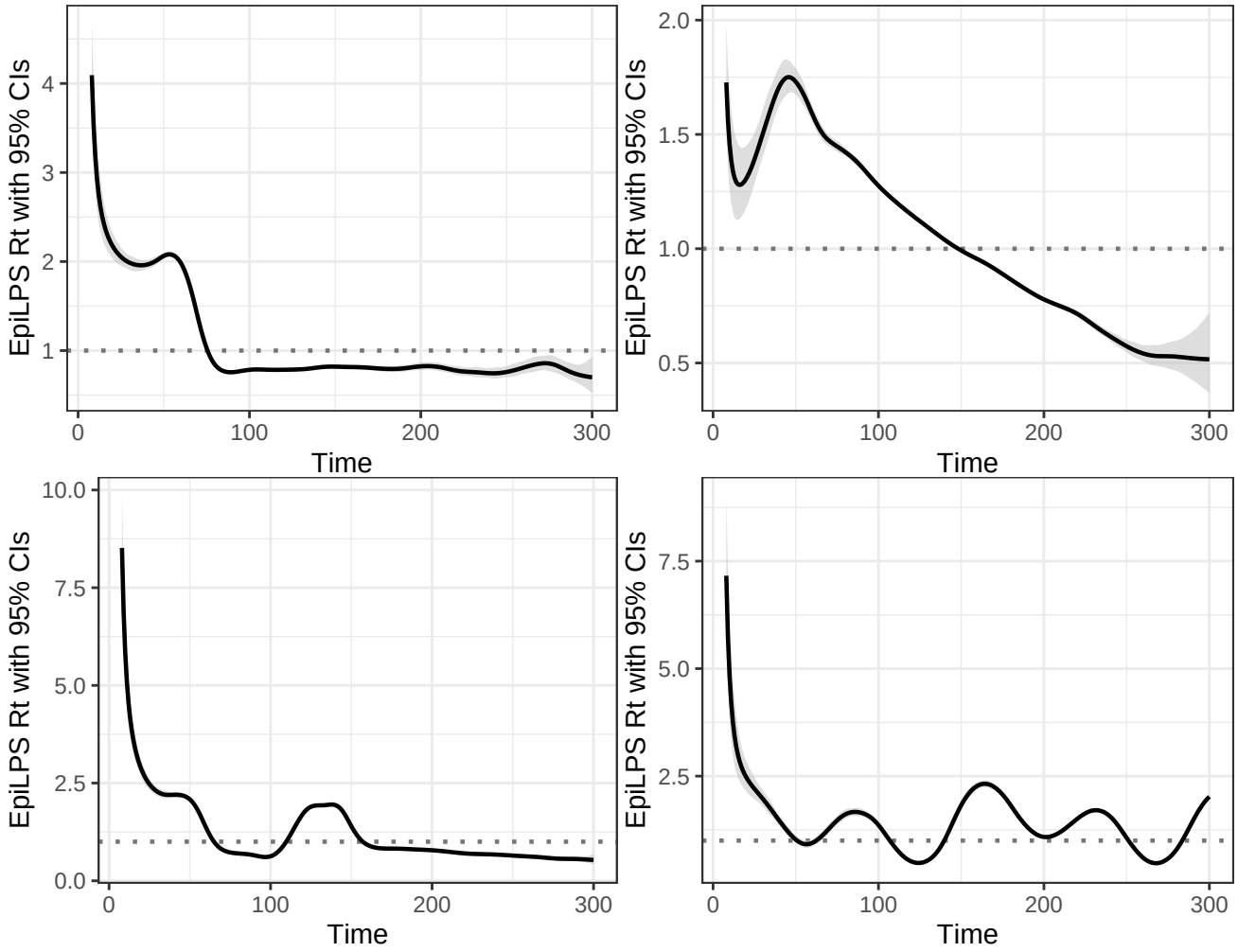


Fit EpiEstim with monthly sliding windows using "true" serial interval.

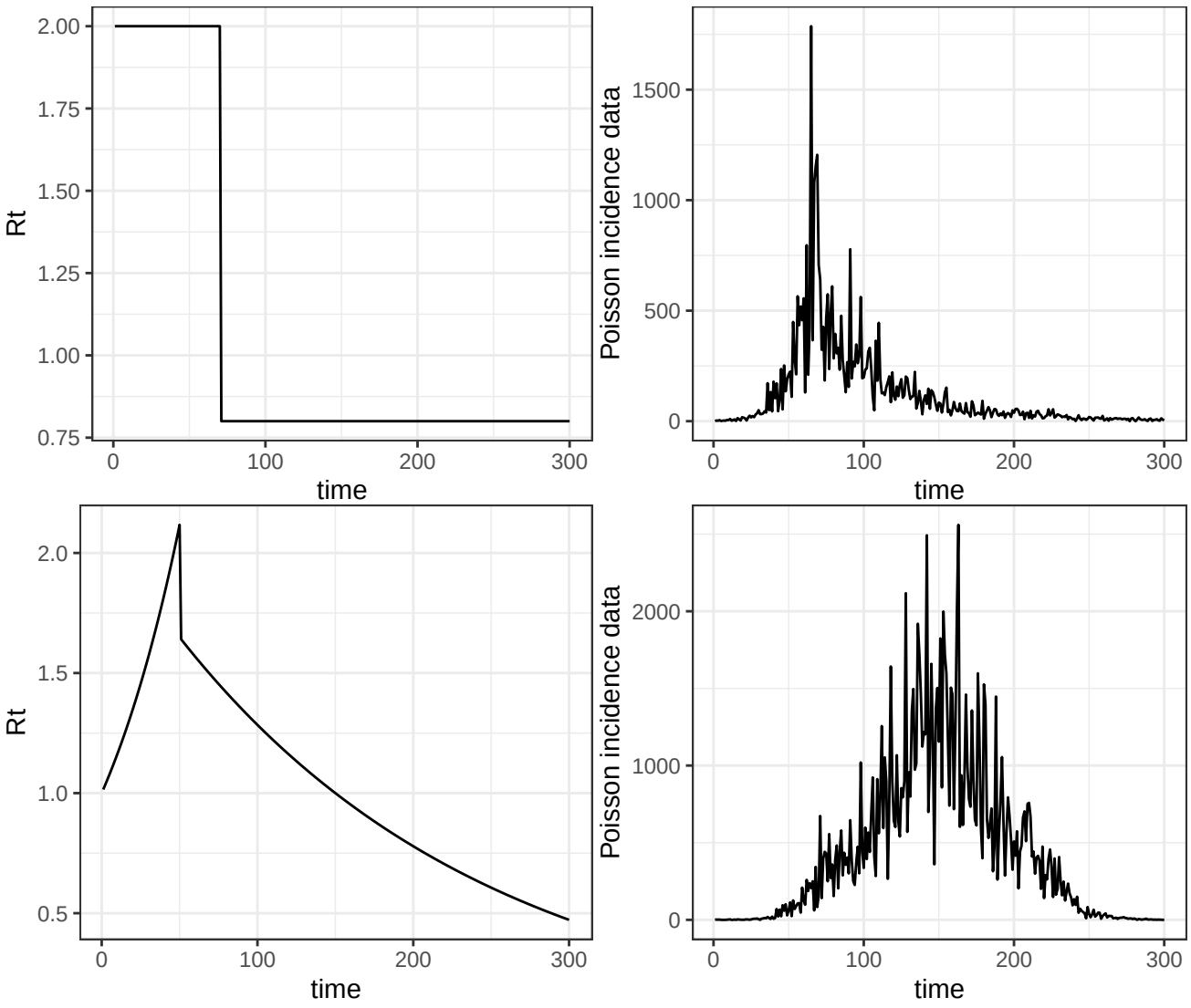


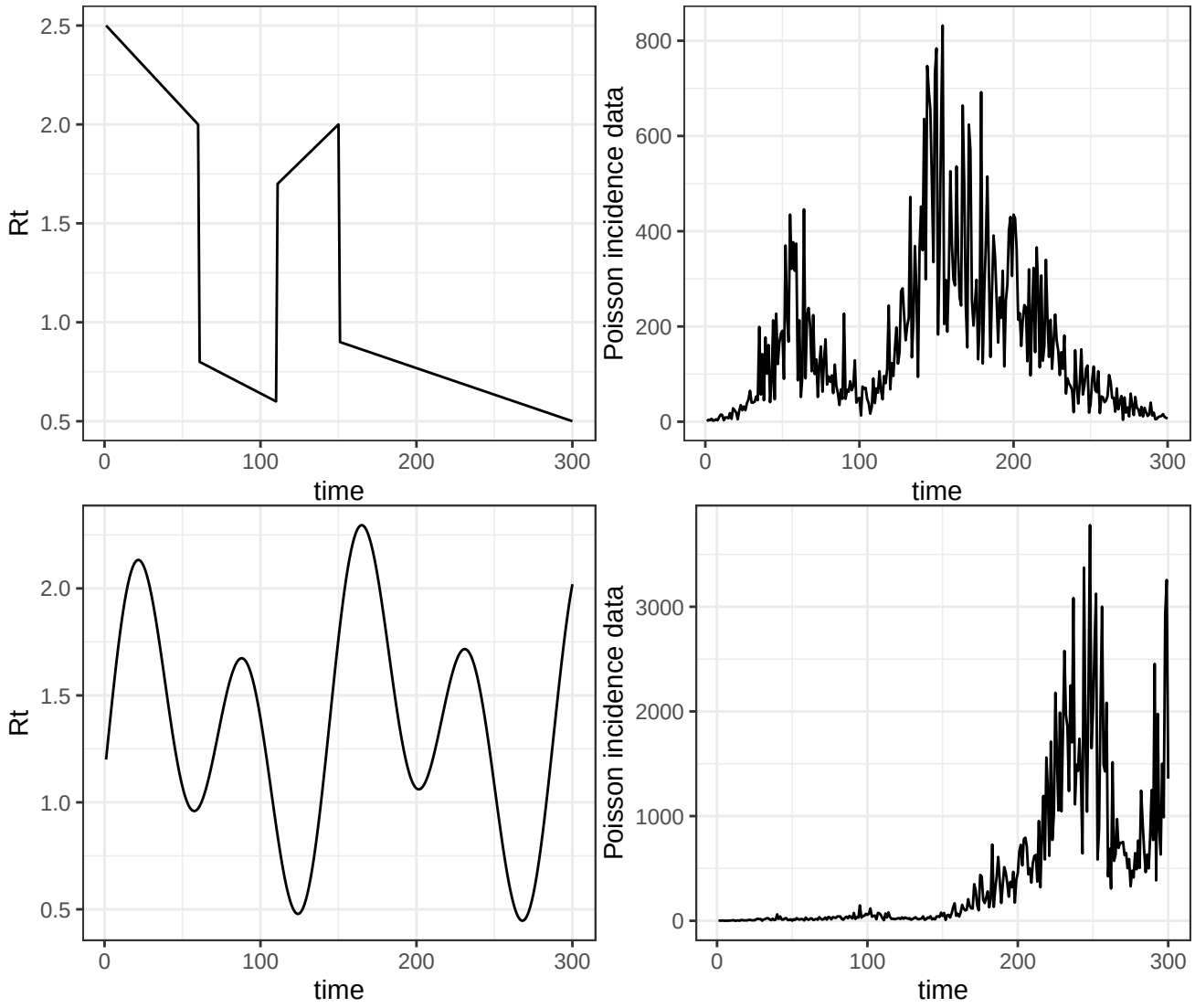
2.4.4 Rt estimates for Poisson incidence using EpiLPS

Fit EpiLPS using "true" serial interval.


EpiLPS Rt with 95% CIs
Time
0
100
200
300
1
2
3
4
0.5
1.0
1.5
2.0
0.0
2.5
5.0
7.5
10.0

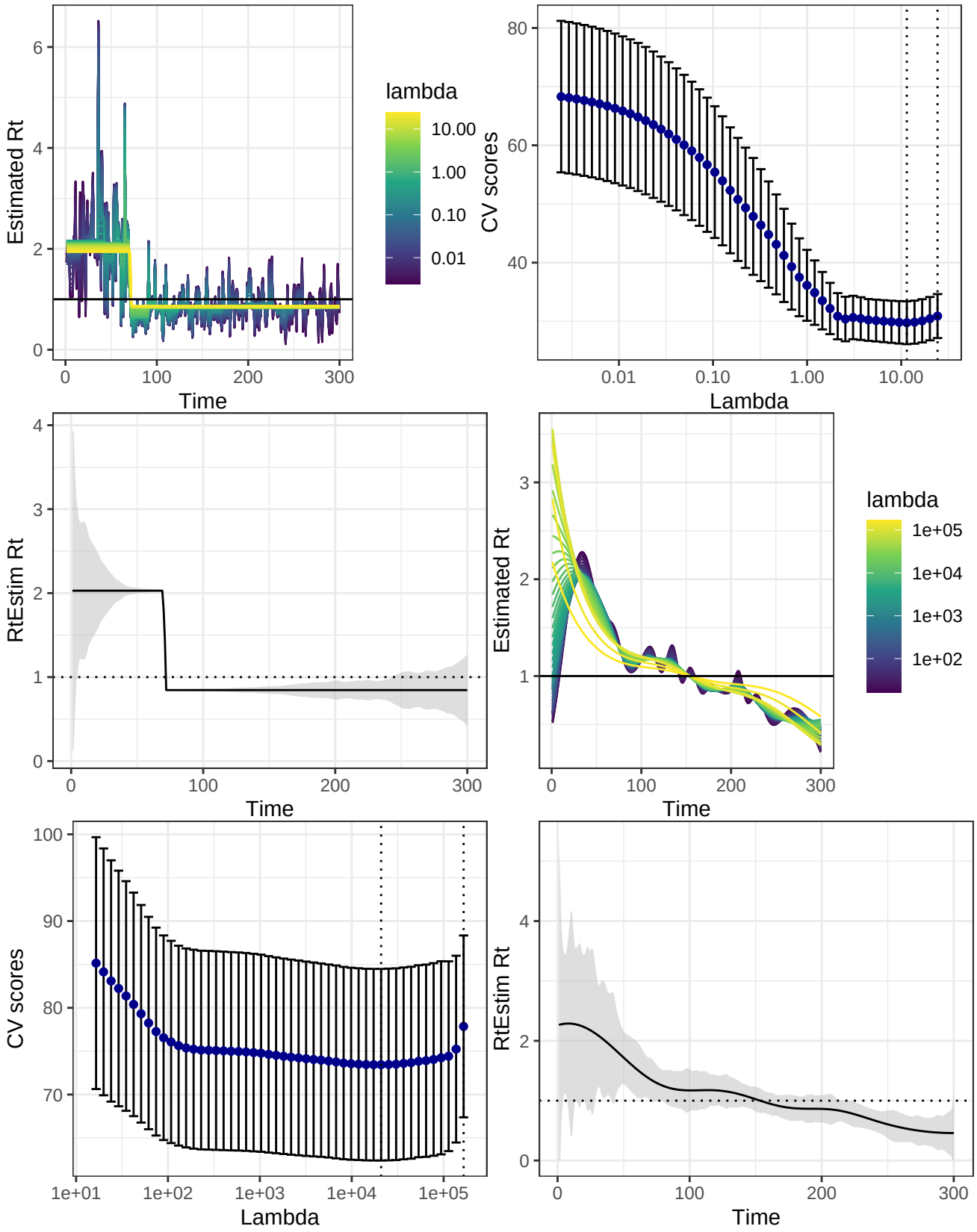
2.5 Synthetic effective reproduction numbers and negative Binomial incidence examples

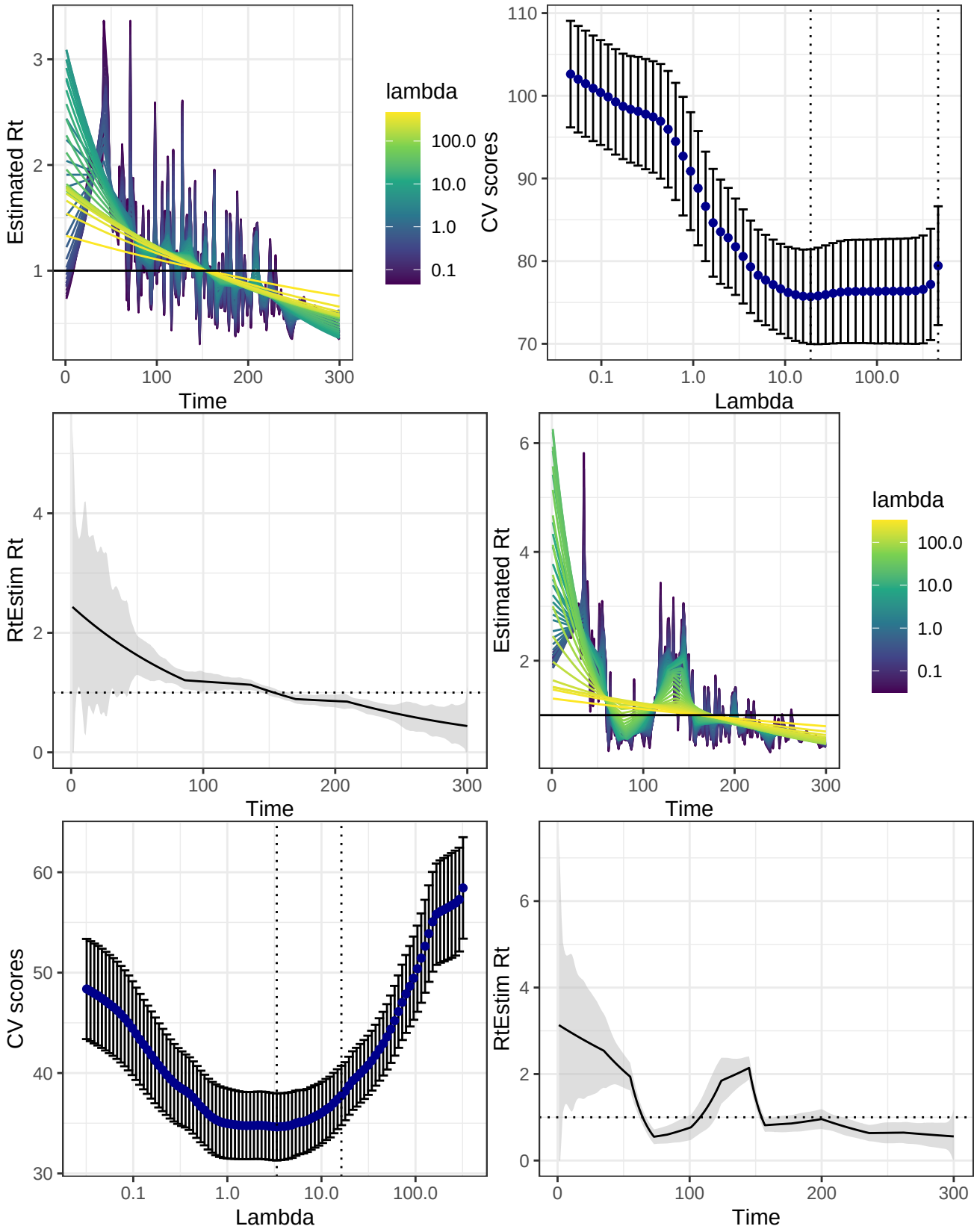


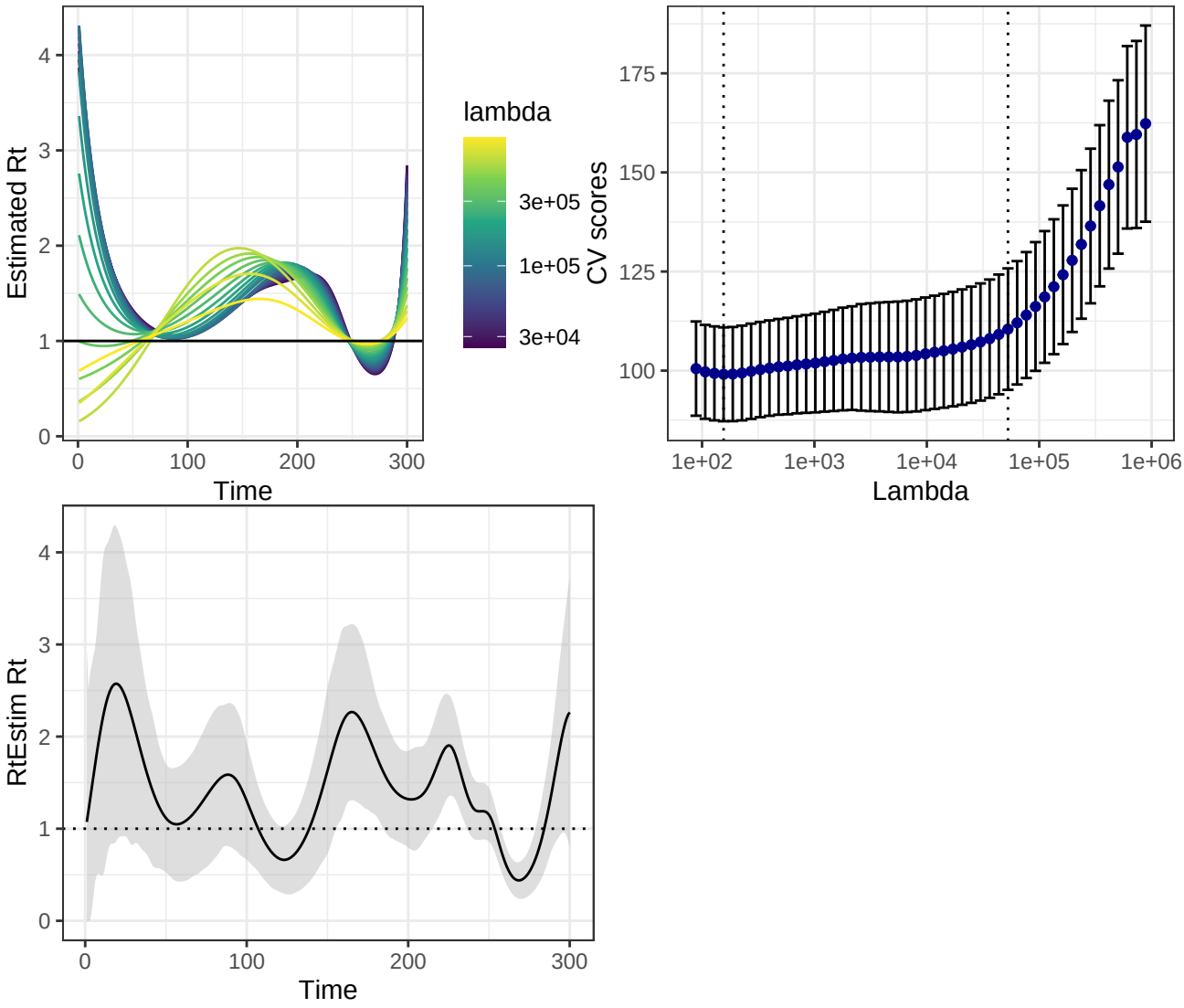


2.5.1 Rt estimates for negative Binomial incidence using RtEstim

Estimate Rt using our `RtEstim` with the same settings as for Poisson incidence.

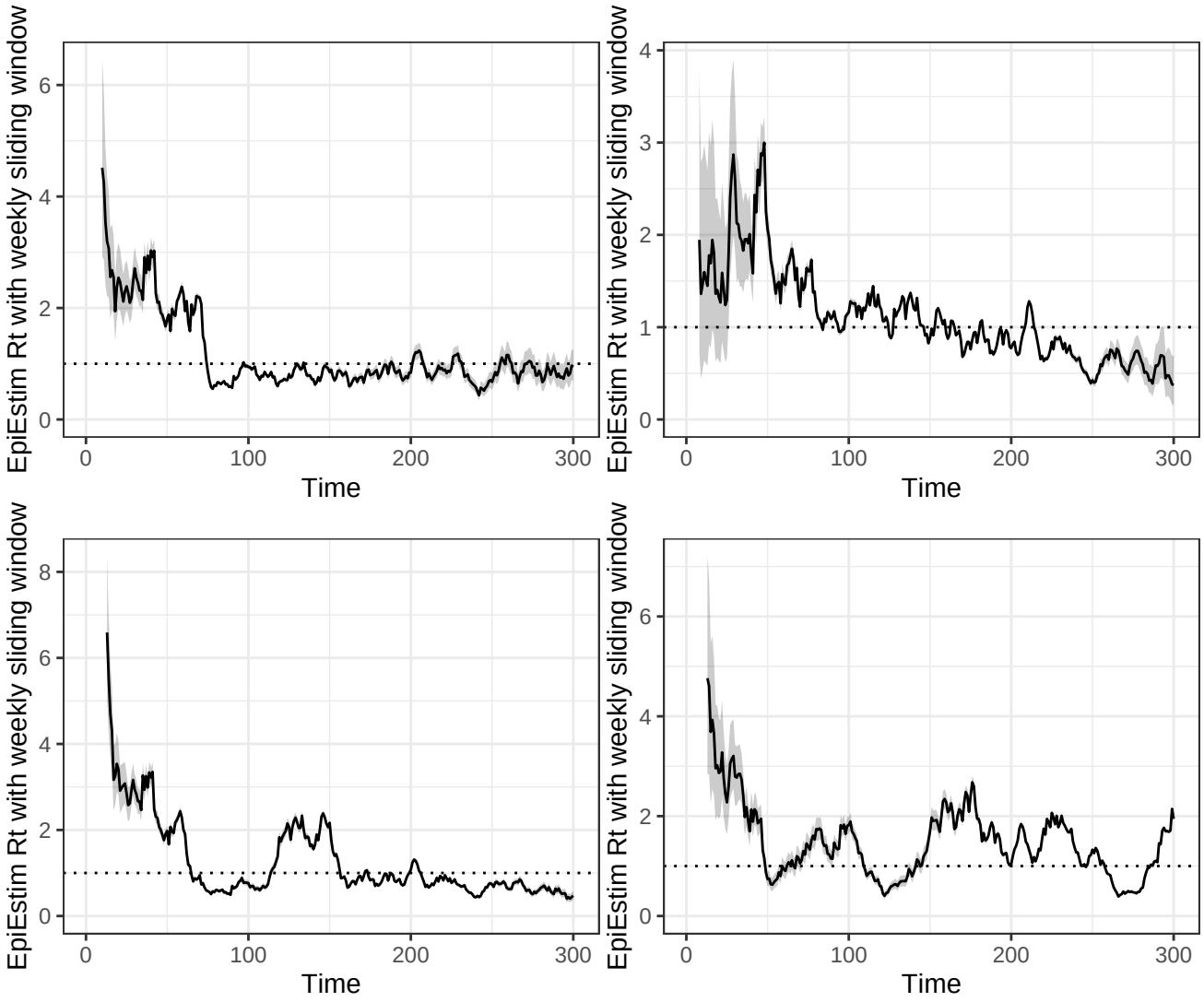

Estimated Rt
Time
lambda
10.00
1.00
0.10
0.01
CV scores
Lambda
RtEstim Rt
Time
Estimated Rt
Time
lambda
1e+05
1e+04
1e+03
1e+02
CV scores
Lambda
RtEstim Rt
Time


Estimated Rt
Time
lambda
100.0
10.0
1.0
0.1
CV scores
Lambda
RtEstim Rt
Time
Estimated Rt
Time
lambda
100.0
10.0
1.0
0.1
CV scores
Lambda
RtEstim Rt
Time

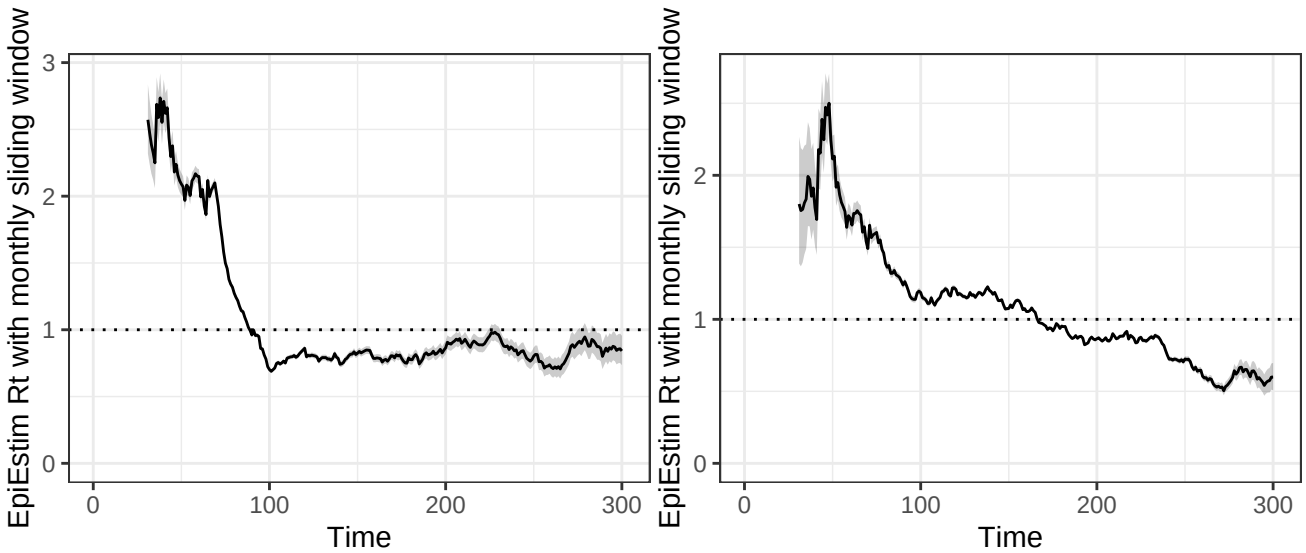


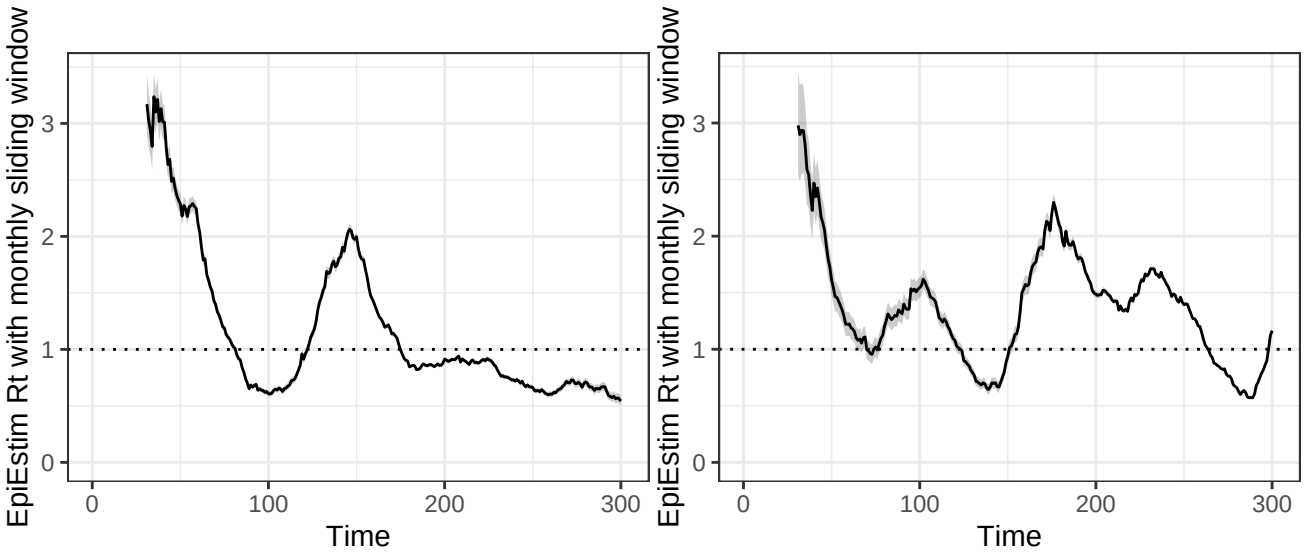
2.5.2 Rt estimates for negative Binomial incidence using EpiEstim

Estimate Rt for negative Binomial incidence using `EpiEstim` with weekly sliding windows.



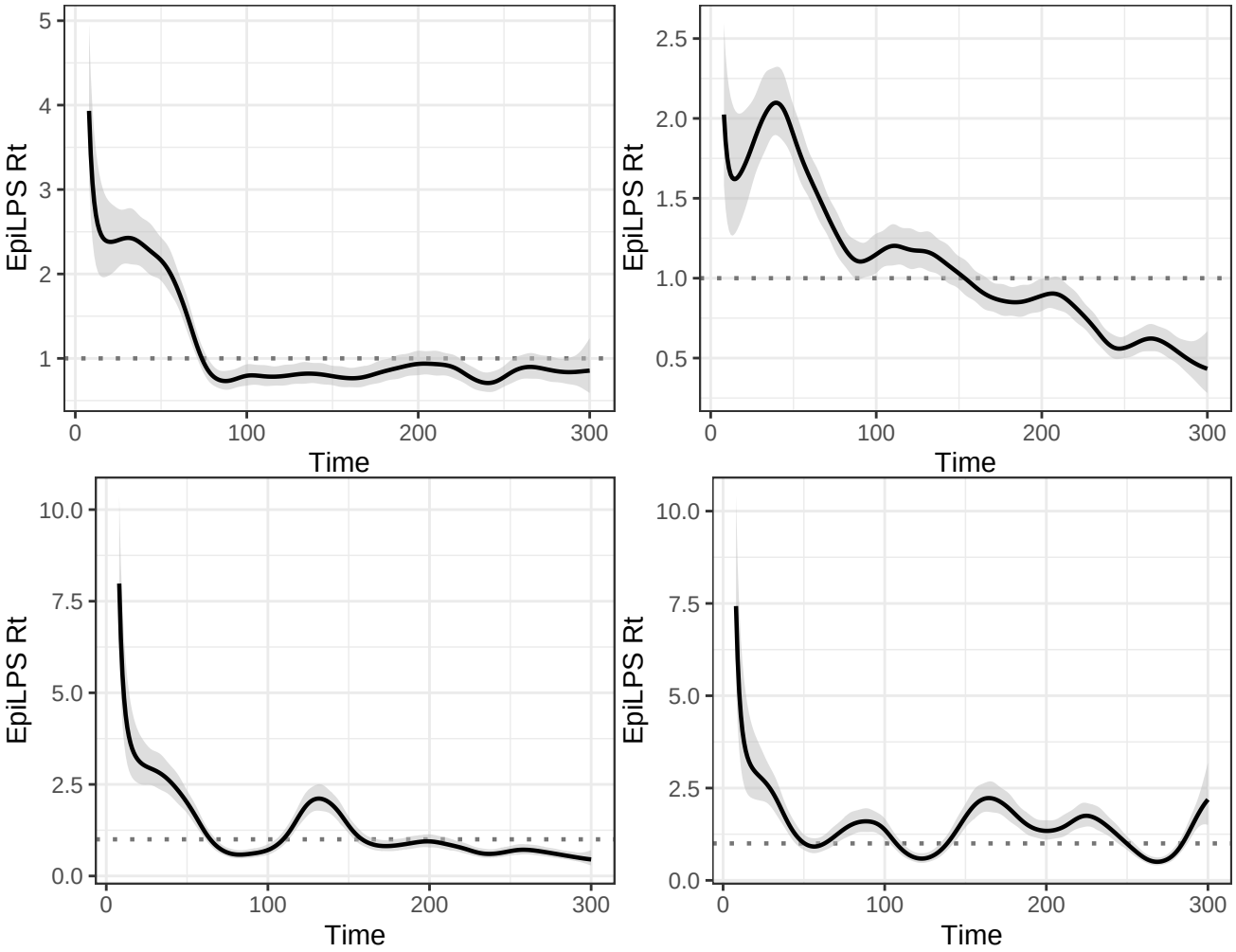
Estimate Rt for negative Binomial incidence using `EpiEstim` with monthly sliding windows.





2.5.3 Rt estimates for negative Binomial incidence using EpiLPS

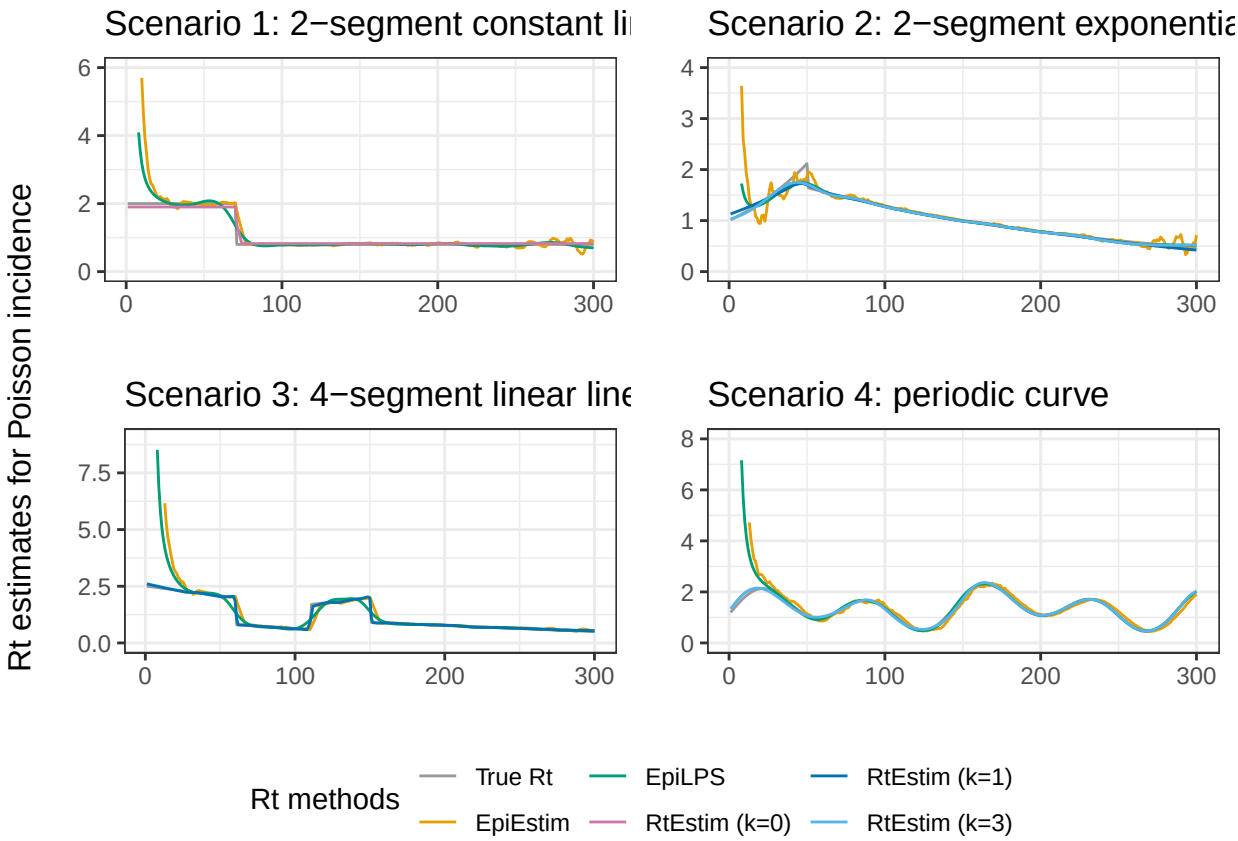
Estimate Rt using `EpiLPS` using the true serial intervals.



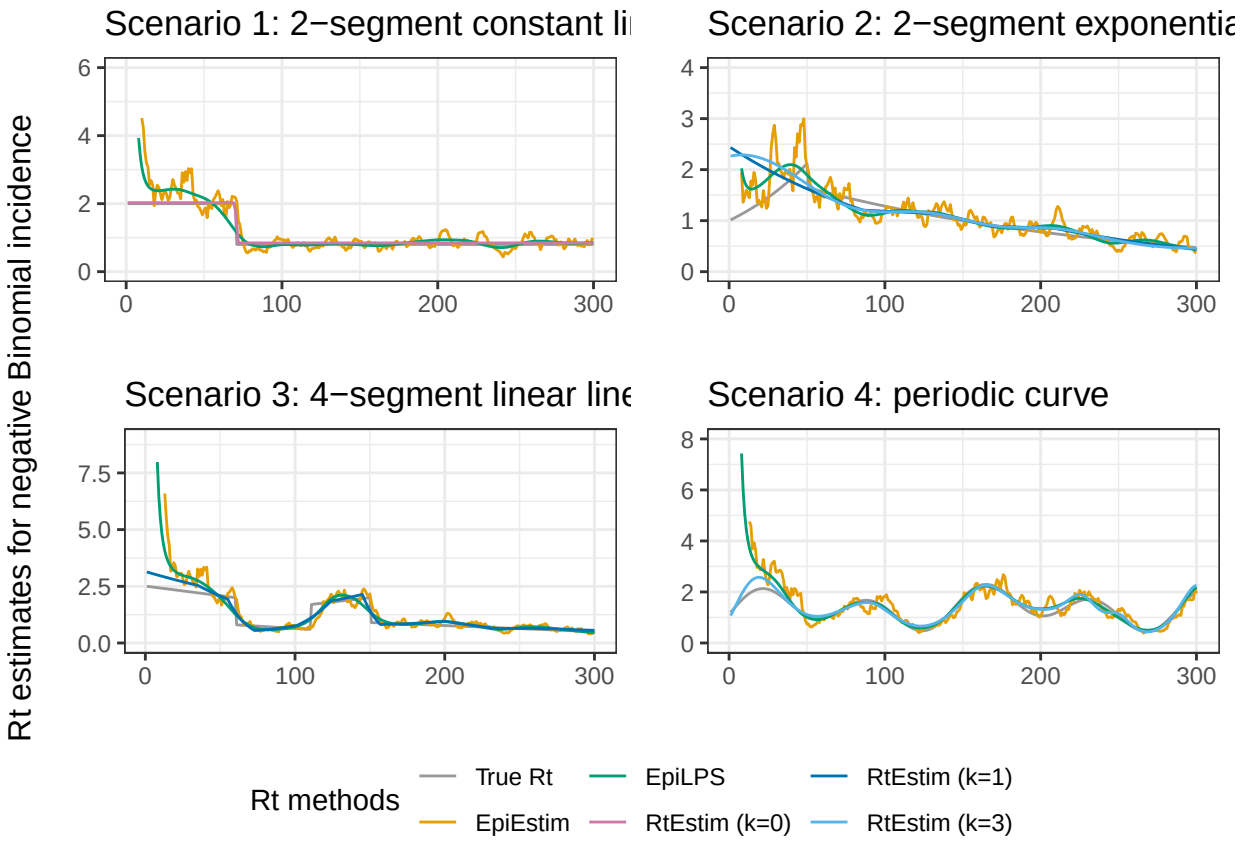
2.6 Display and save results for synthetic examples

Create a results table to save all Rt and incidence samples, and Rt estimates using three methods. Save the results table.

Display results of Poisson incidence.

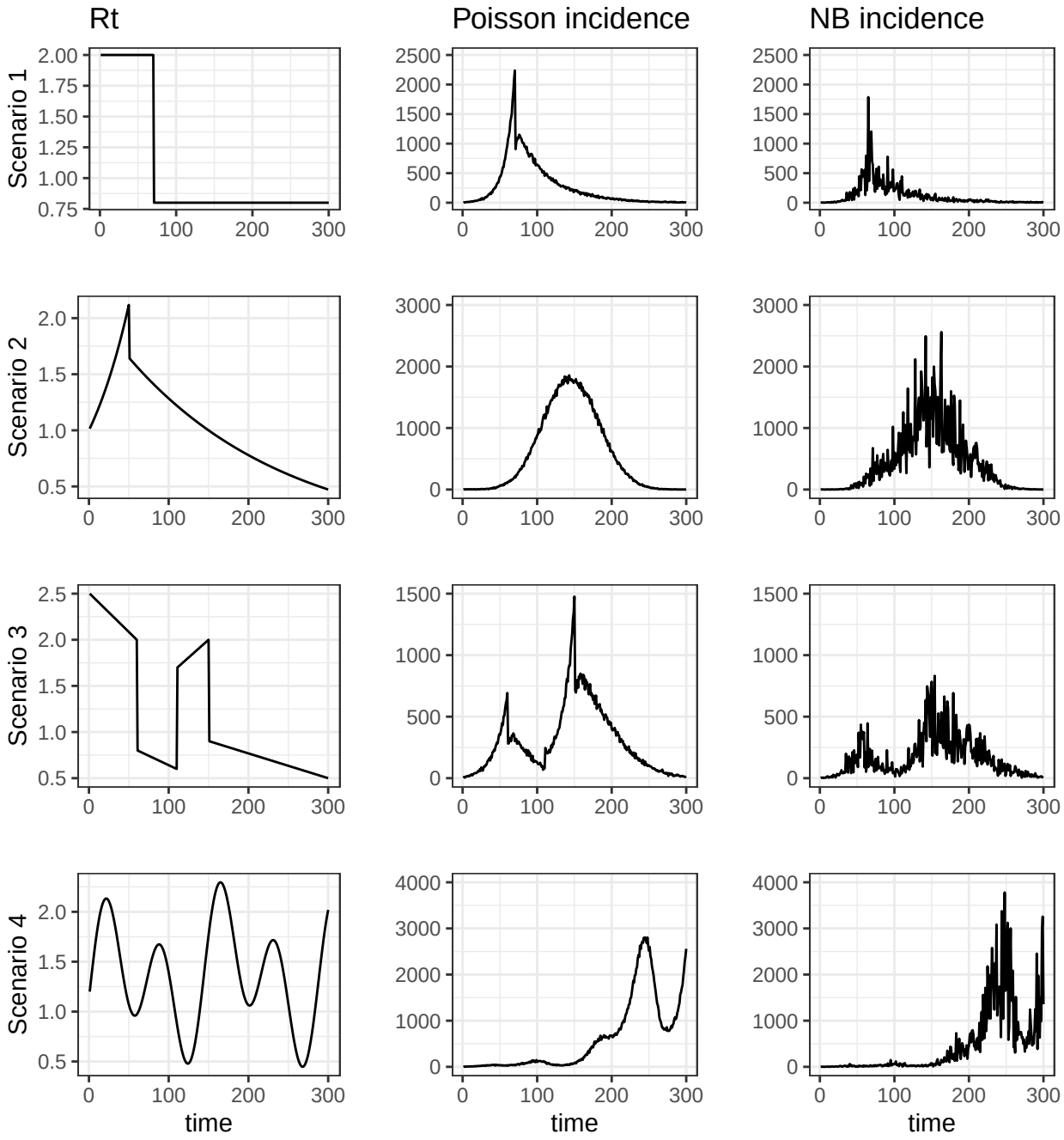


Display results of negative Binomial incidence.



Save graphical display of estimates across all Rt scenarios for both incidence distributional assumptions.

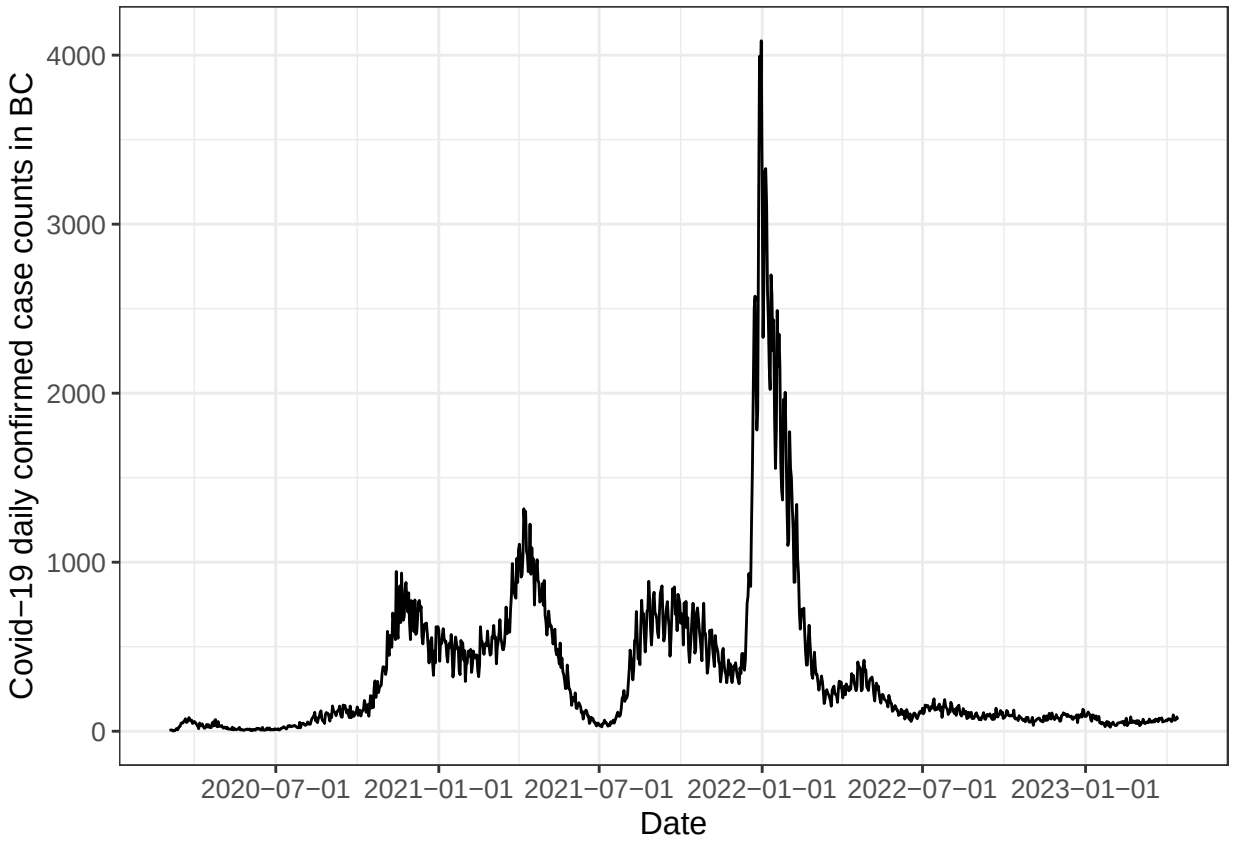
Save graphical display of all Rt and incidence samples across both incidence distributional assumptions.



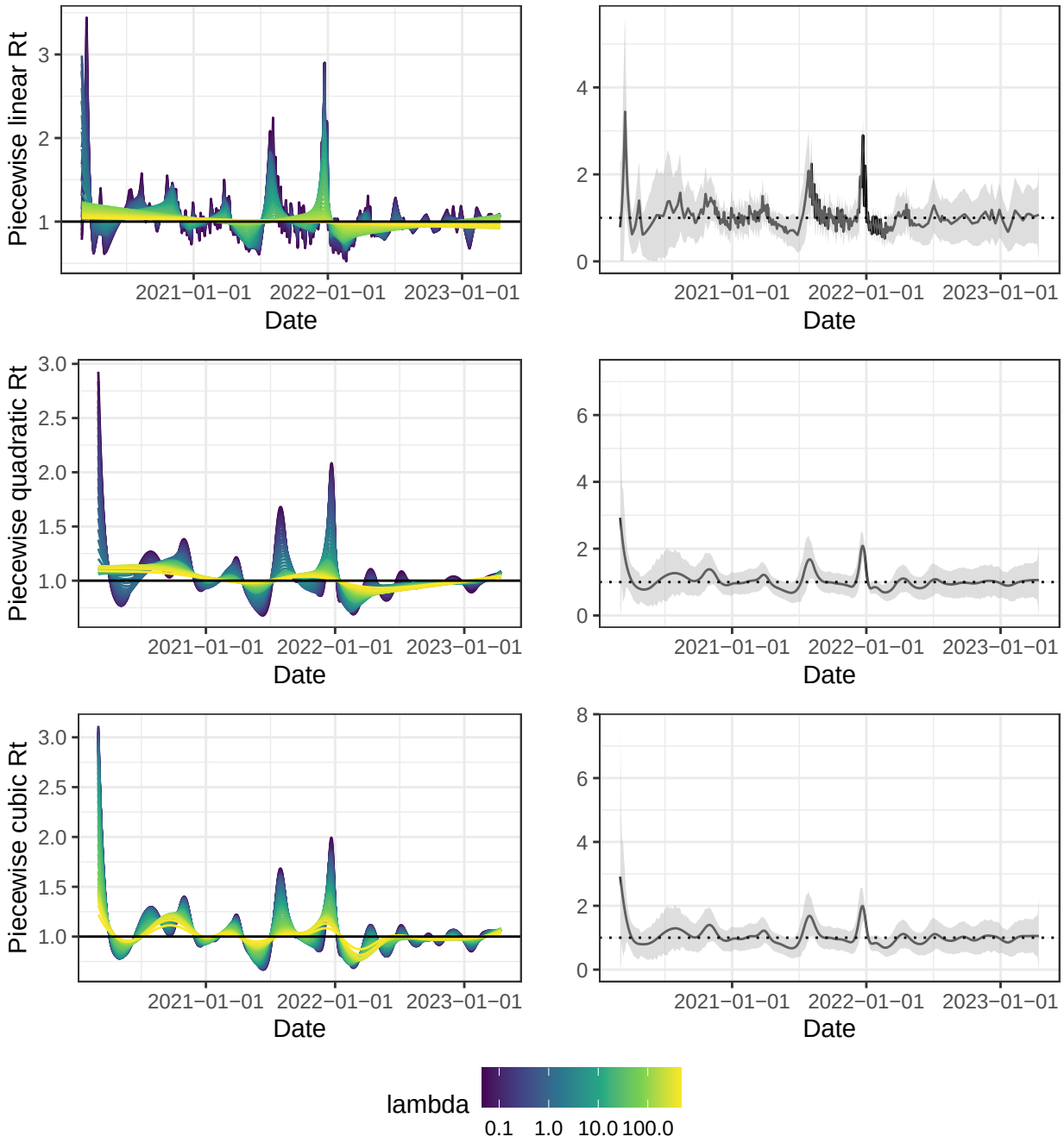
3 Real case applications

3.1 Covid-19 Canada

Get latest Covid-19 incidence data in Canada.

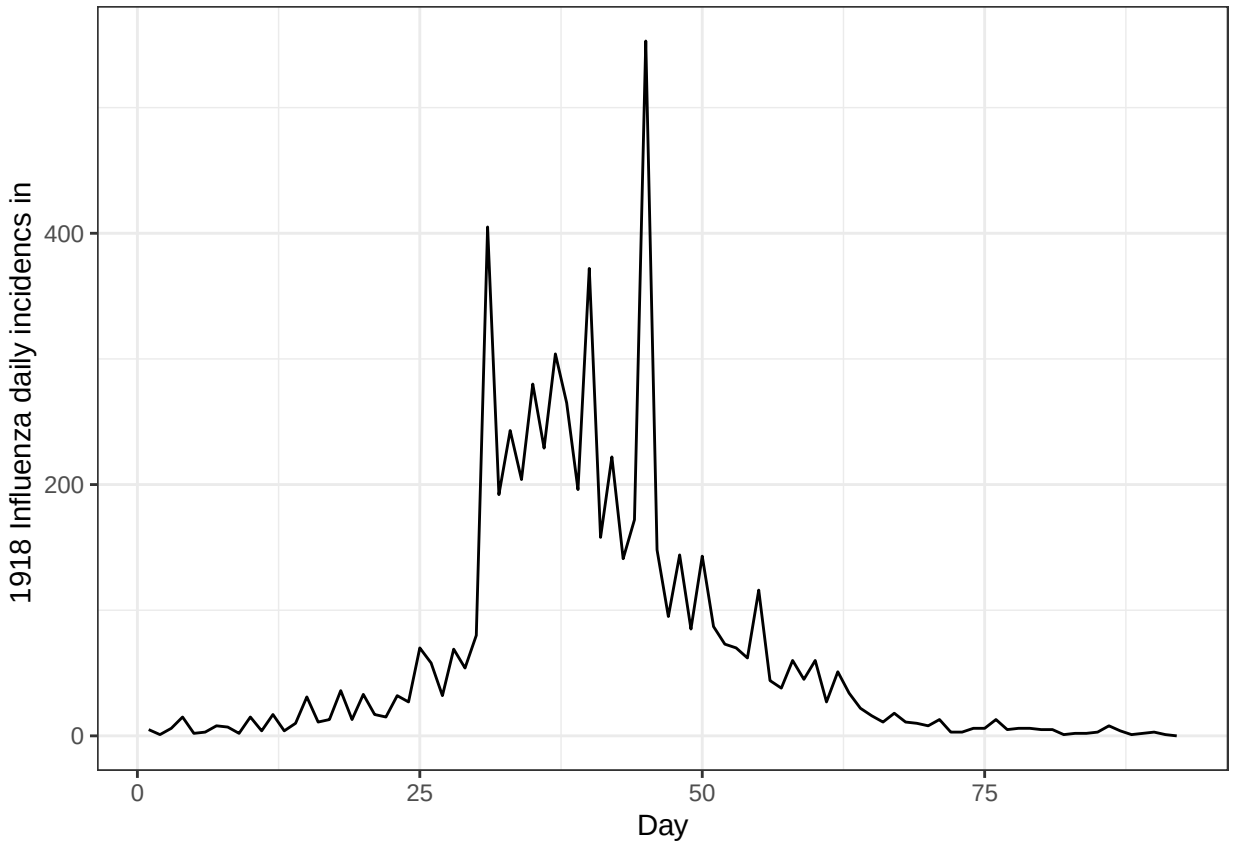


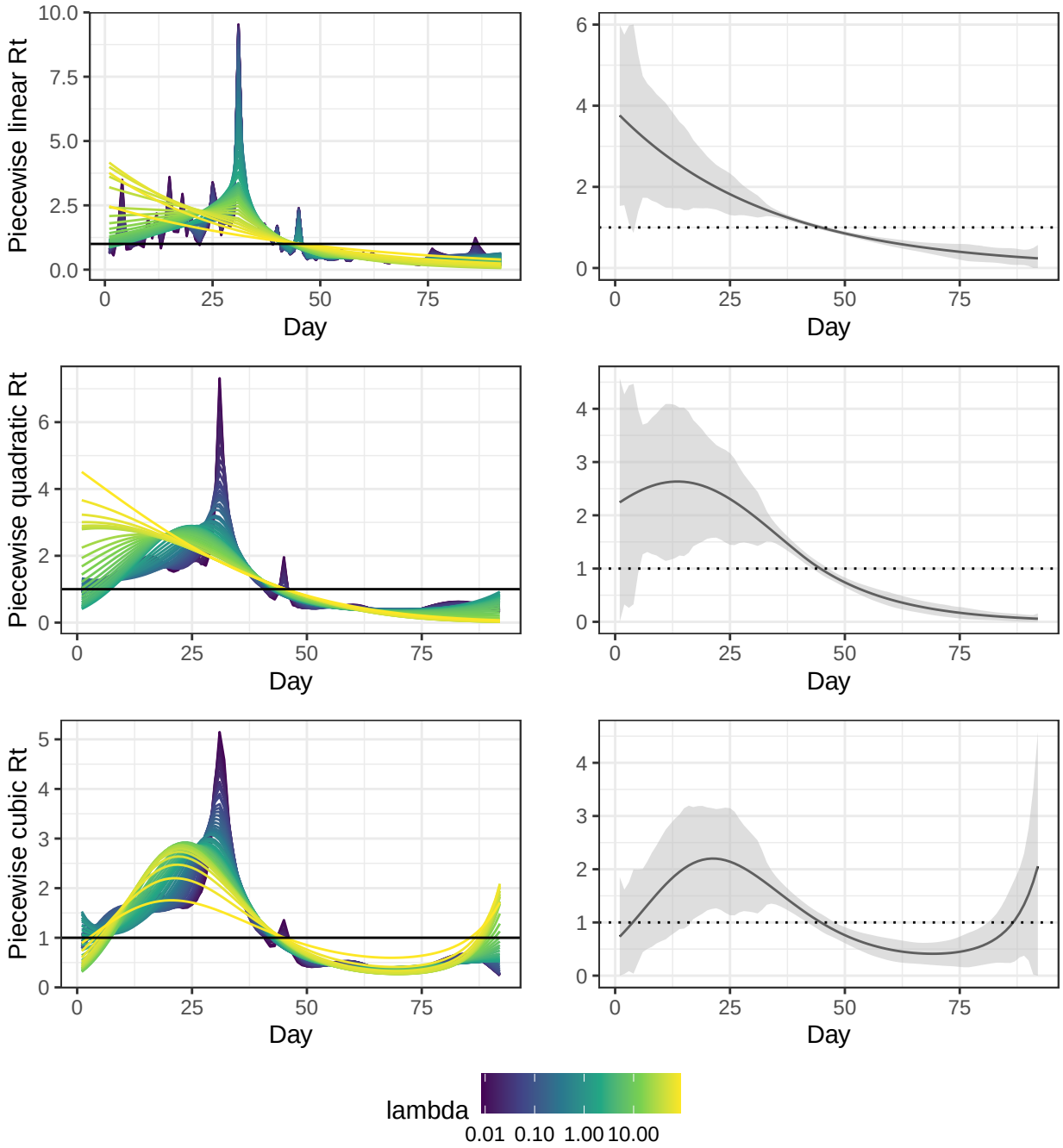
Estimate Rt using `RtEstim`.



3.2 1918 H1N1 influenza in the USA

Obtain 1918 H1N1 influenza data in Baltimore, Maryland from `EpiEtim` package.


1918 Influenza daily incidencs in
400
200
0
0
25
50
75
Day



References

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

[Ramdas and Tibshirani(2016)] A Ramdas and R J Tibshirani. Fast and flexible admm algorithms for trend filtering. *Journal of Computational and Graphical Statistics*. 25(3):839–858, 2016.

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