Report COVID19 Model

Måns Magnusson

2020-05-05

Contents

Summary / Take Away 1
Covariate effects 1
R0 for different countries 3
Infections, deaths and Rt by country and model 4
Stan code 15

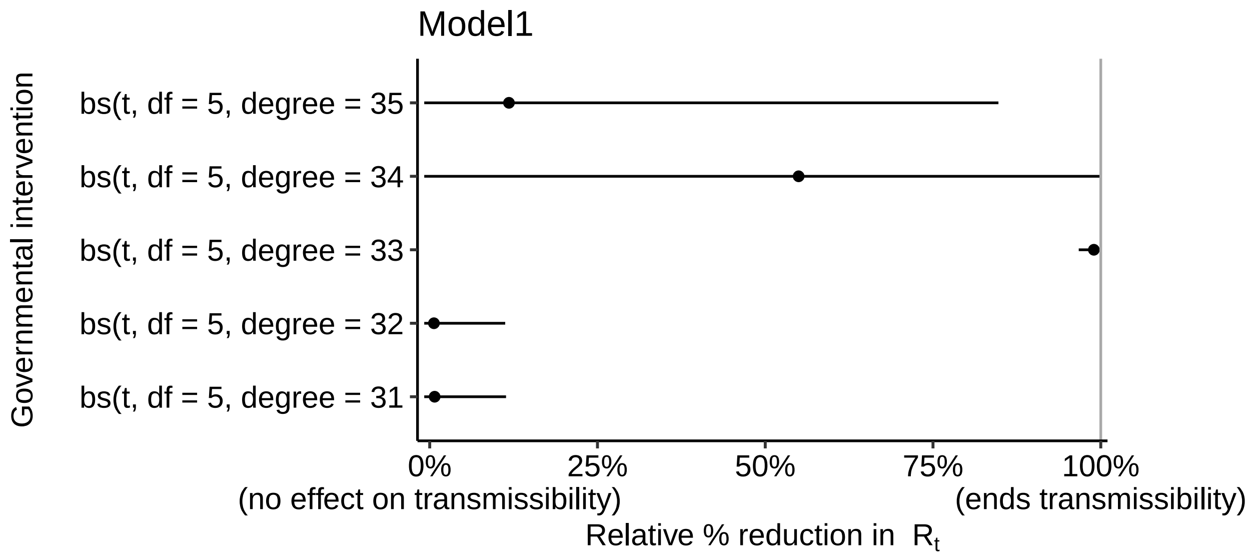
Summary / Take Away

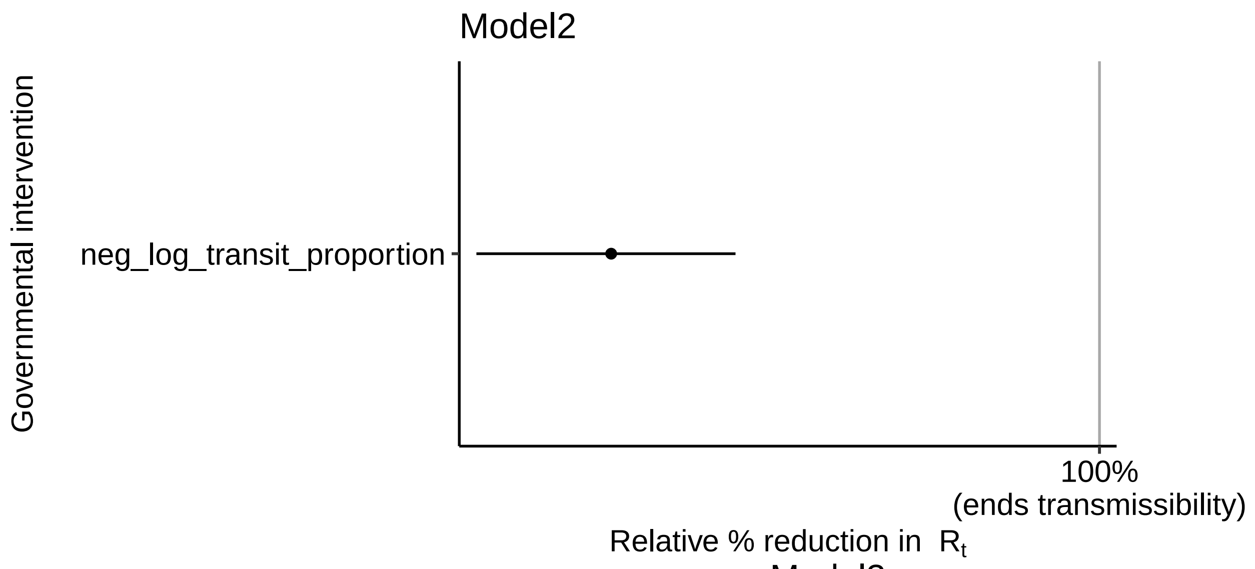
- It is possible to include b-splines in the model.
- It does not seem to give a lot of gains with using b-splines compare to linear effects.
- Using splines for t is working well to capture the underlying change in Rt, although, this should be needed to be different for different countries.

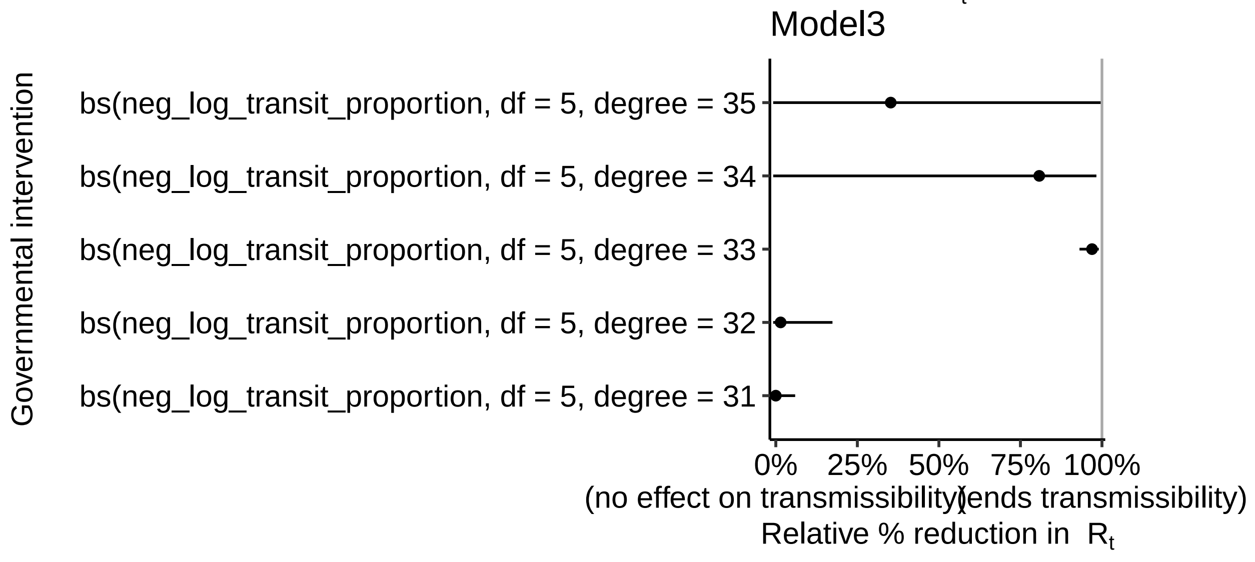
Model descriptions

- Model 1: Splines on t, the time point
- Model 2: Linear neg_log_transit_proportion
- Model 3: Splines on neg_log_transit_proportion

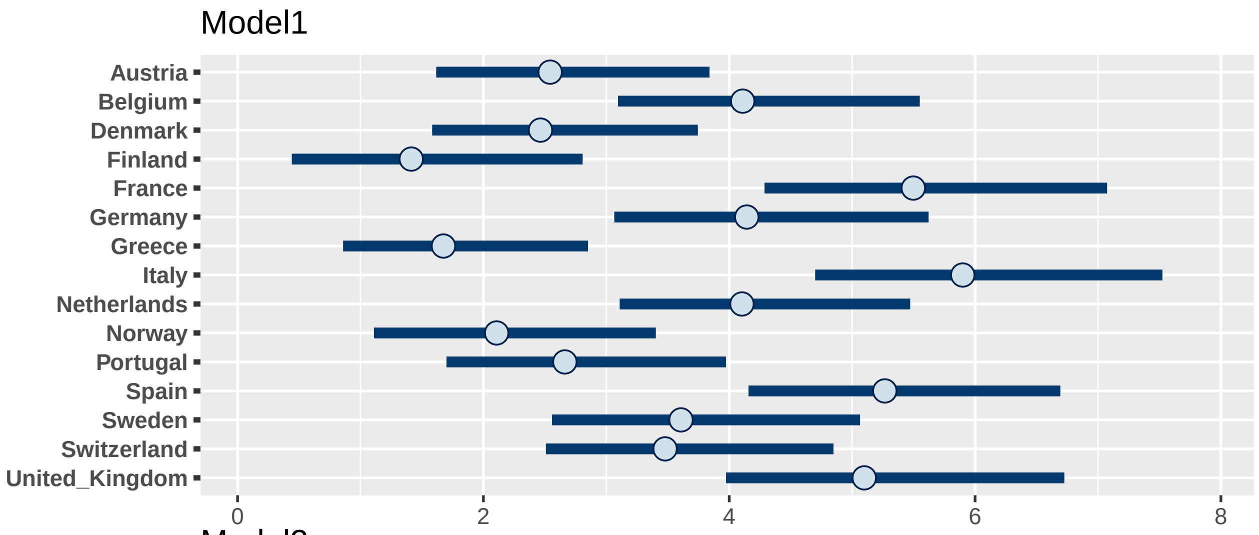
Covariate effects

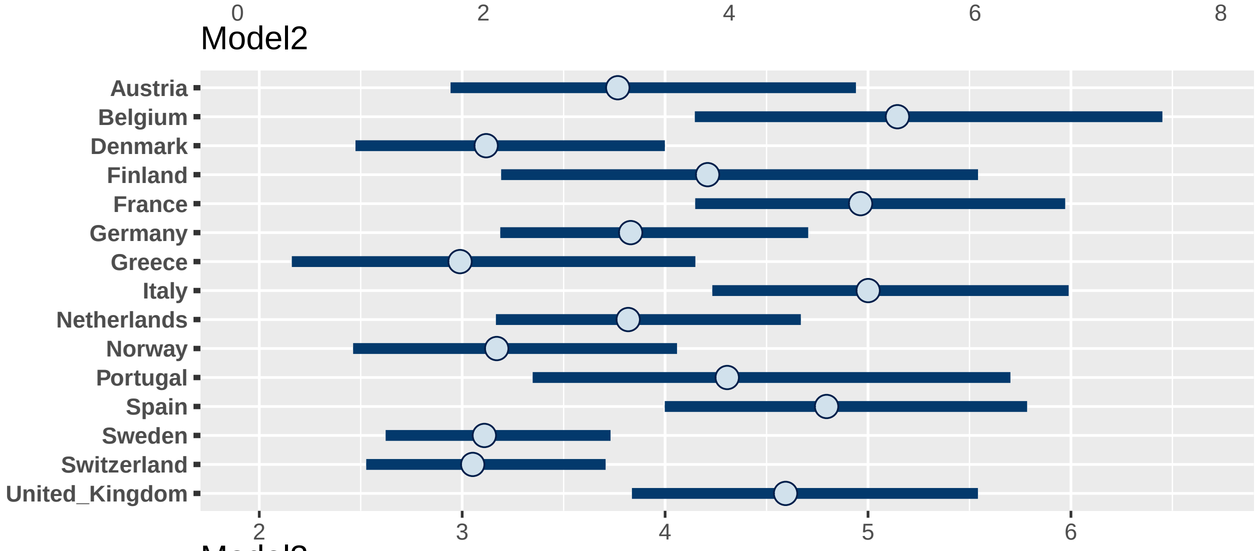


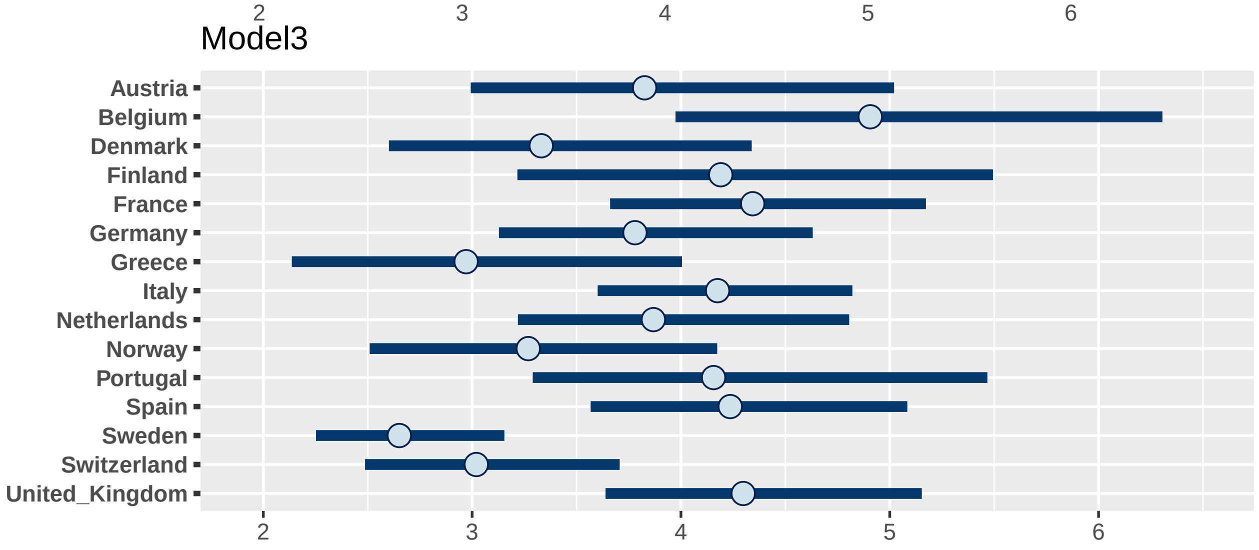

Model2
Governmental intervention
neg_log_transit_proportion
100%
(ends transmissibility)
Relative % reduction in R_t


Model3
Governmental intervention
bs(neg_log_transit_proportion, df = 5, degree = 35
bs(neg_log_transit_proportion, df = 5, degree = 34
bs(neg_log_transit_proportion, df = 5, degree = 33
bs(neg_log_transit_proportion, df = 5, degree = 32
bs(neg_log_transit_proportion, df = 5, degree = 31
0%
25%
50%
75%
100%
(no effect on transmissibility)
(ends transmissibility)
Relative % reduction in R_t

R0 for different countries

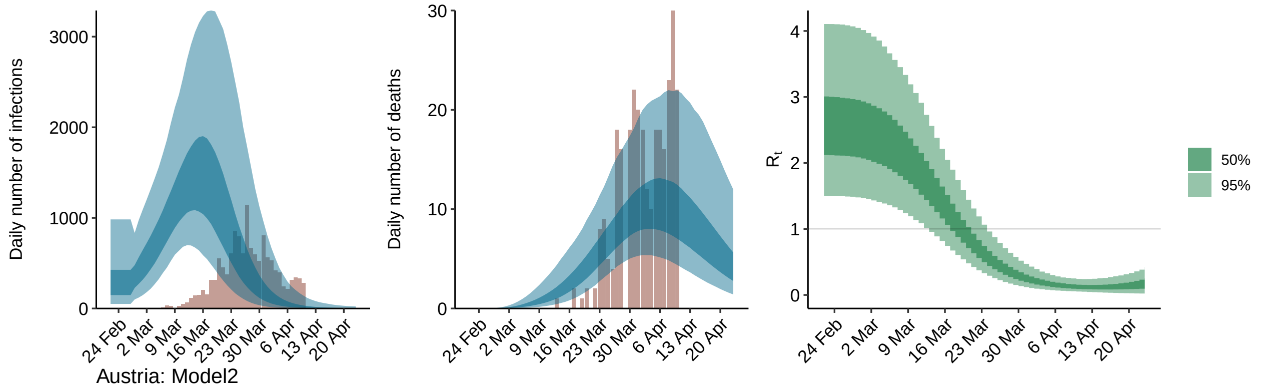




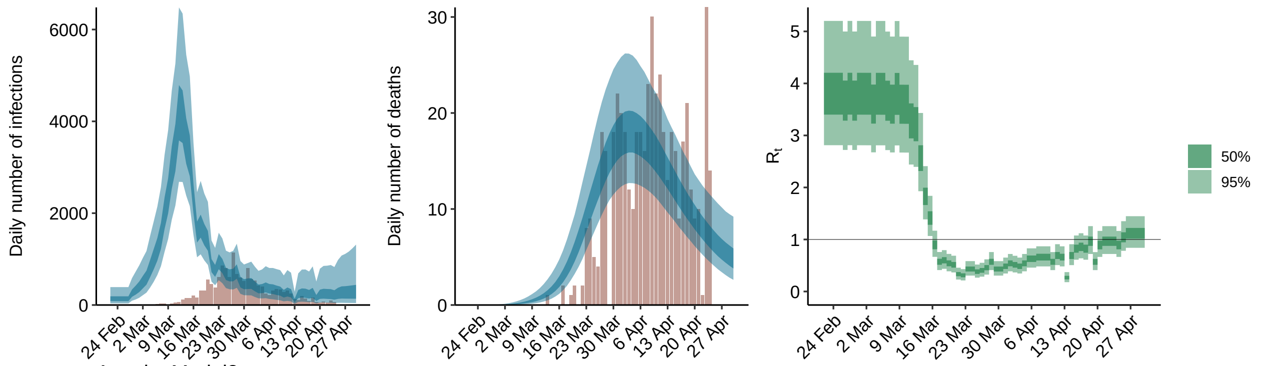


Infections, deaths and Rt by country and model

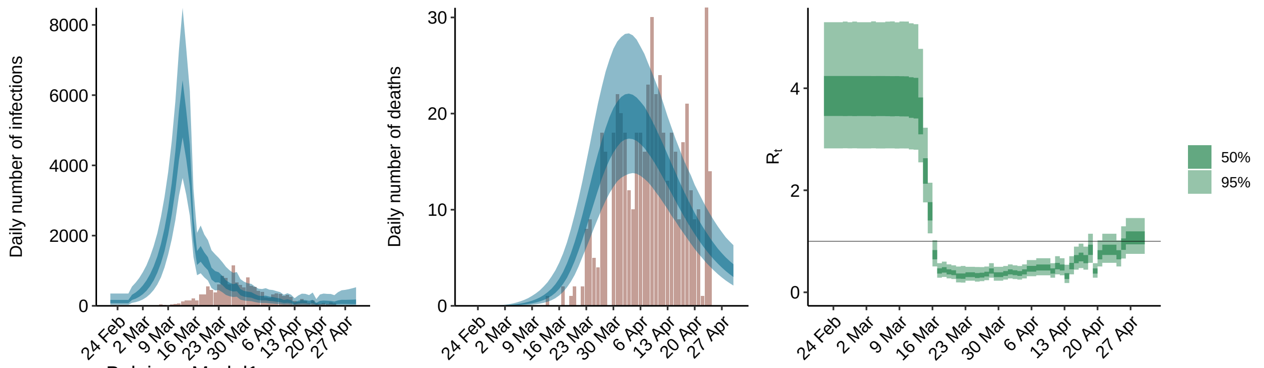
Austria: Model1



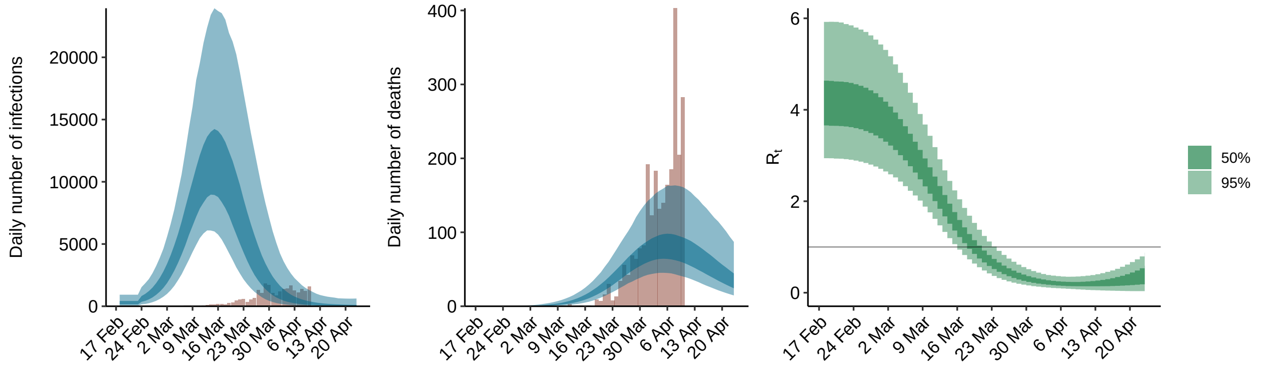
Austria: Model2

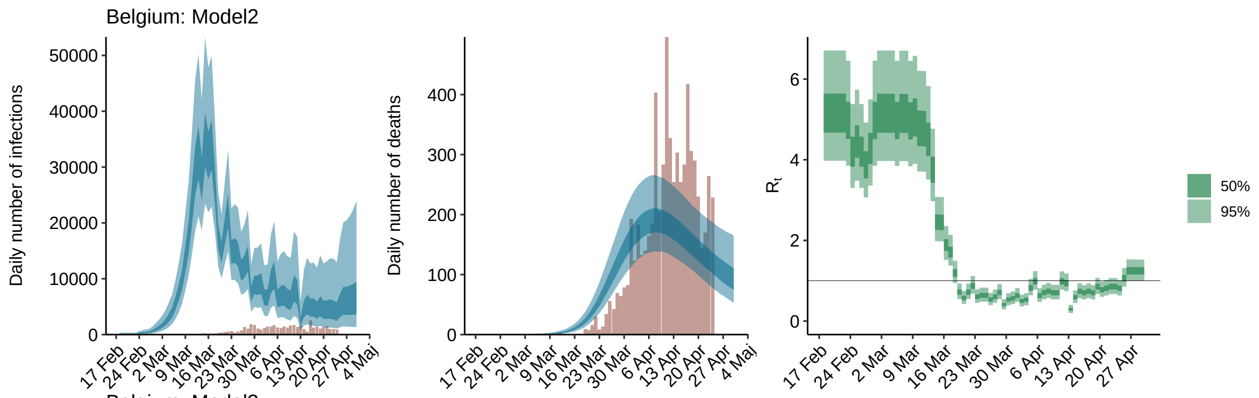


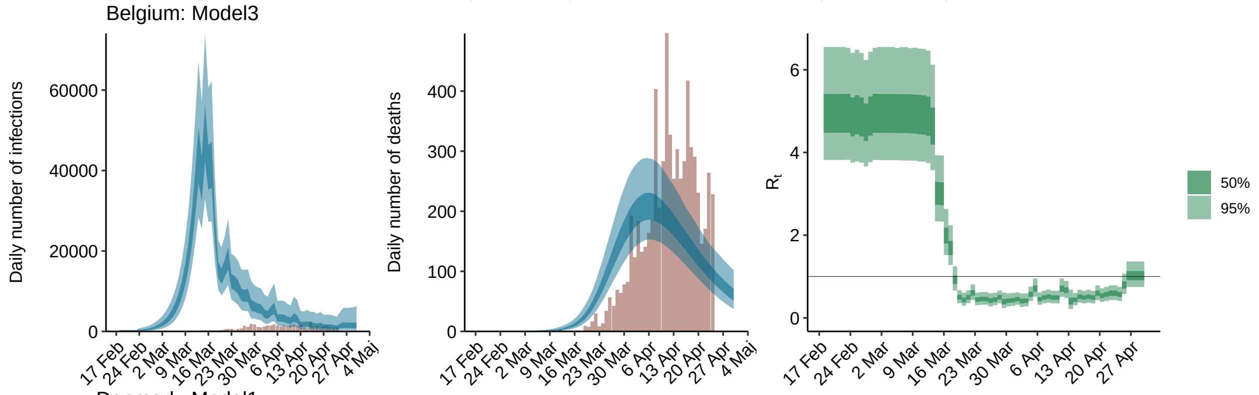
Austria: Model3

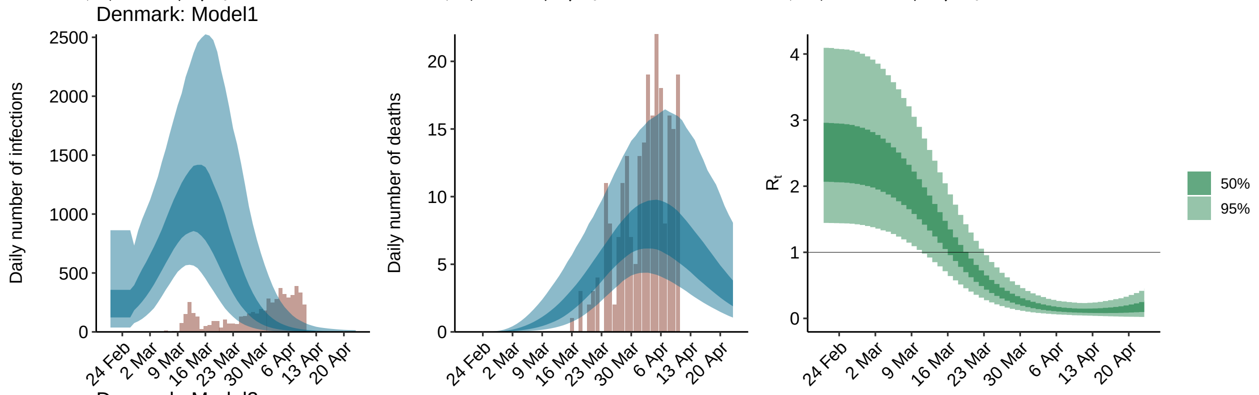


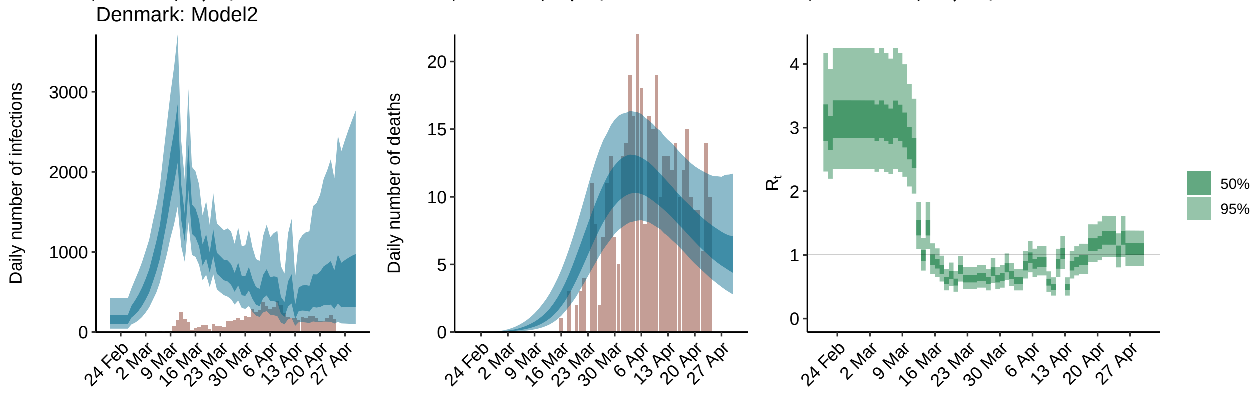
Belgium: Model1

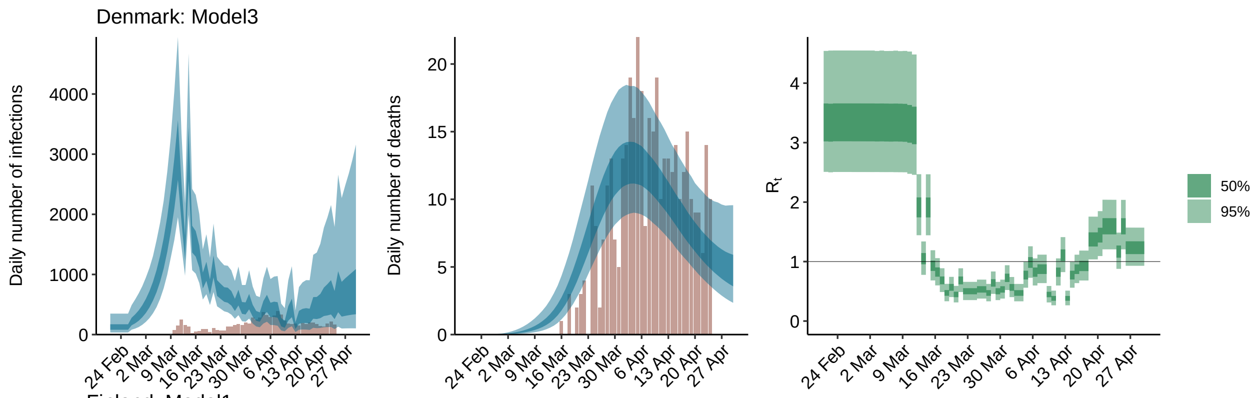
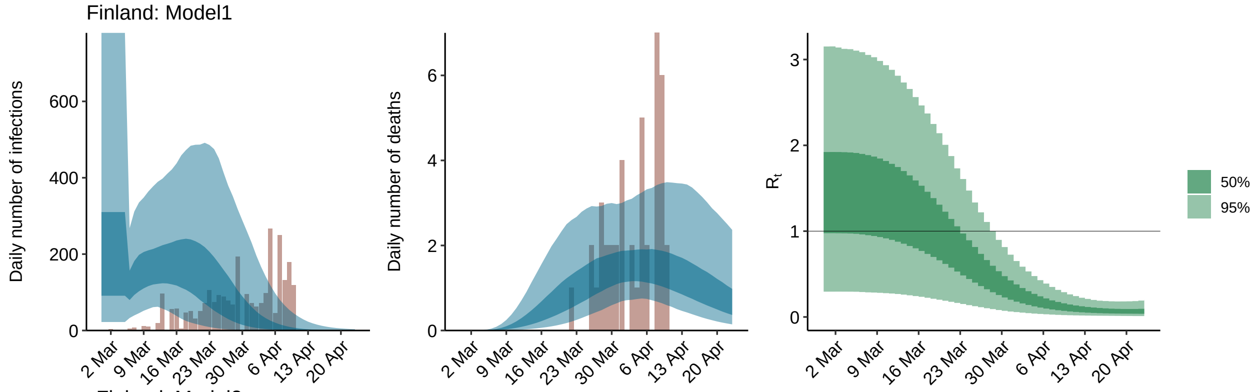
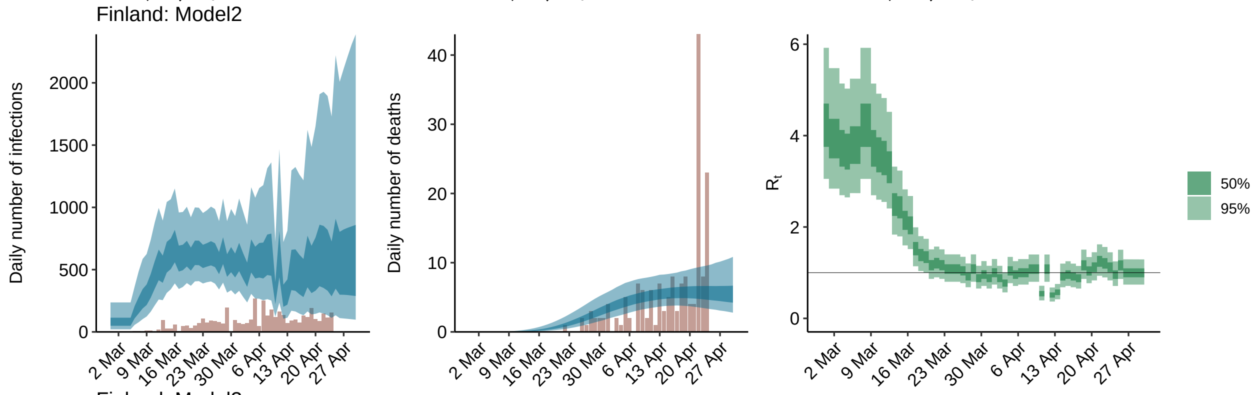
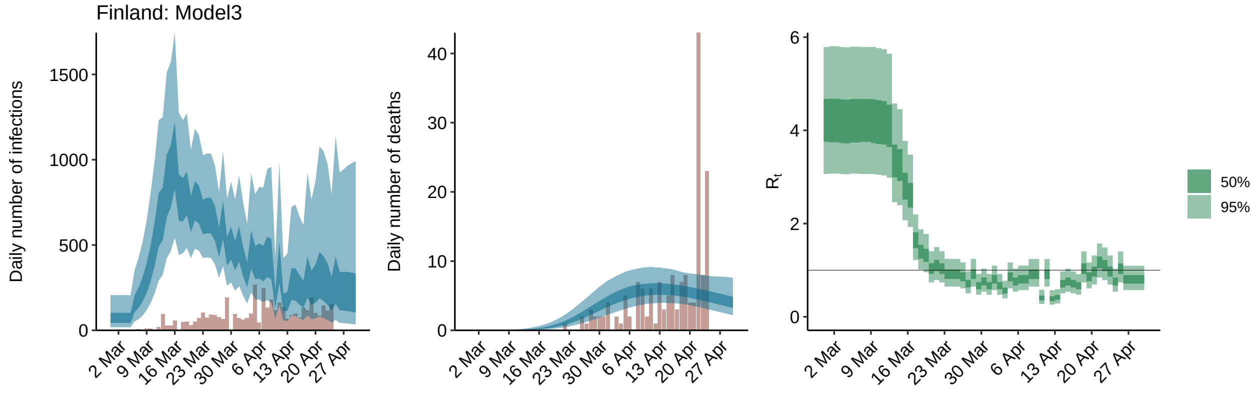


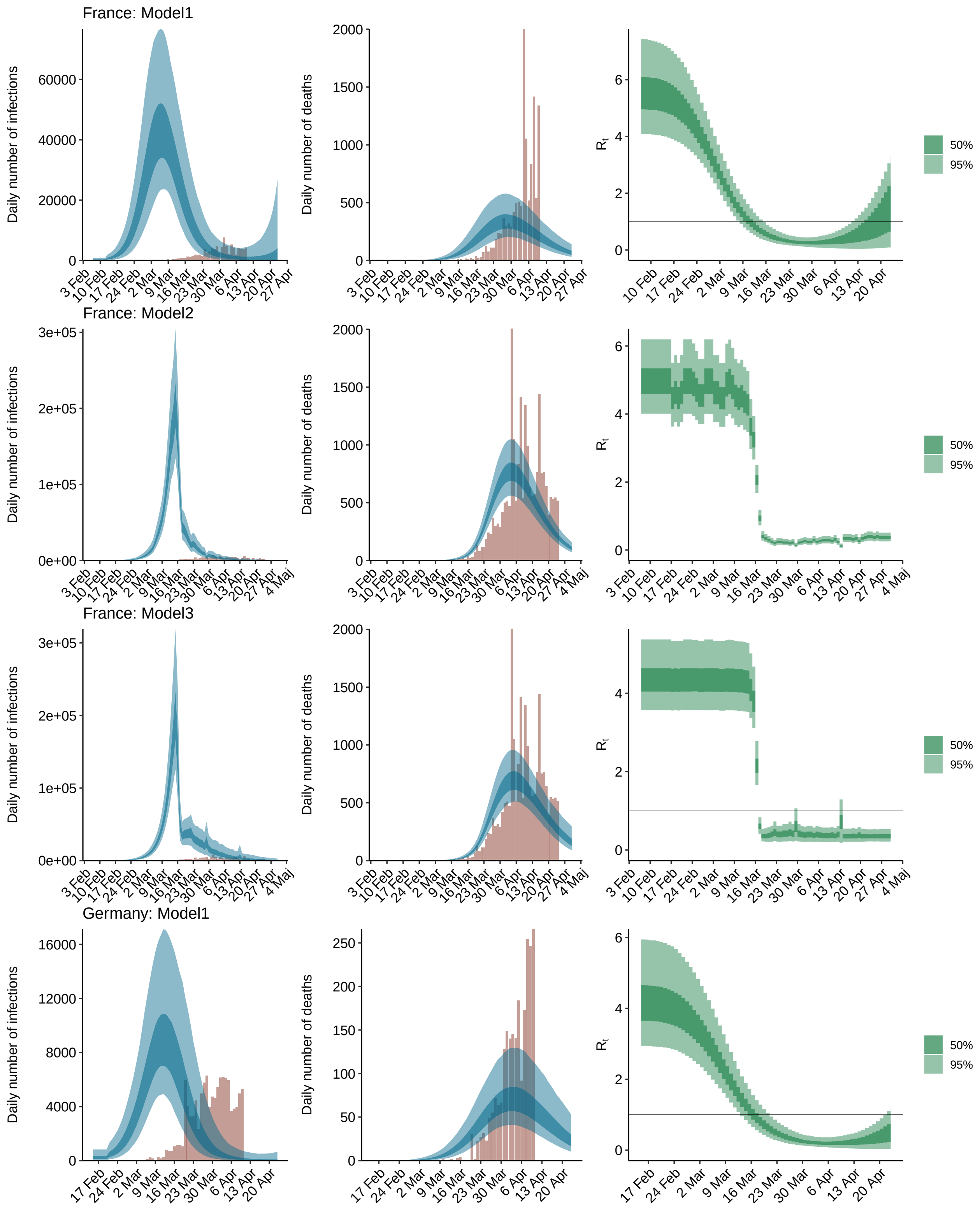

Belgium: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%

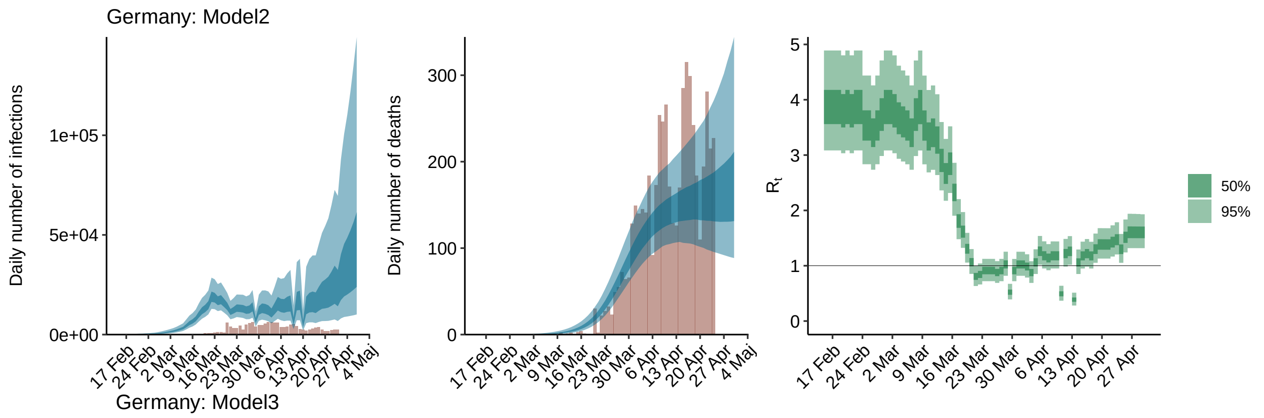

Belgium: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%

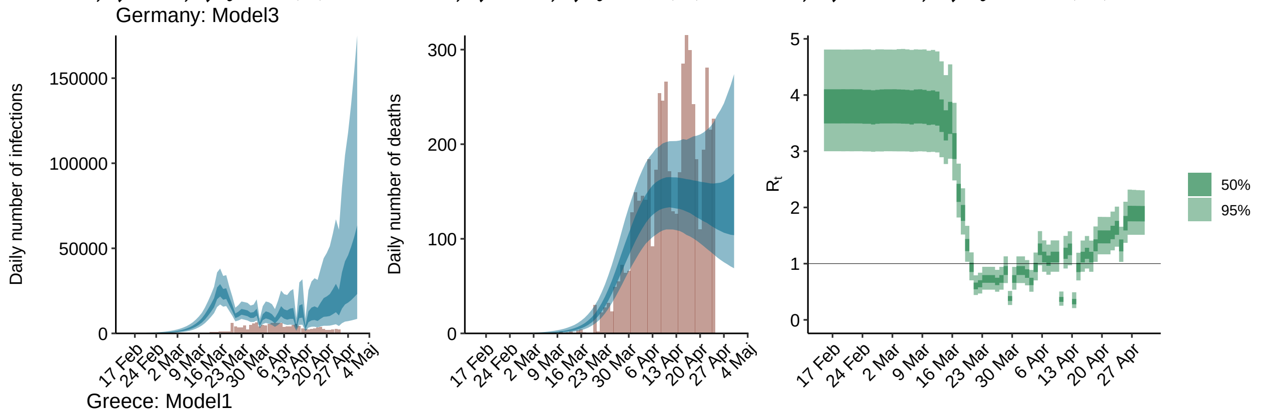

Denmark: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%

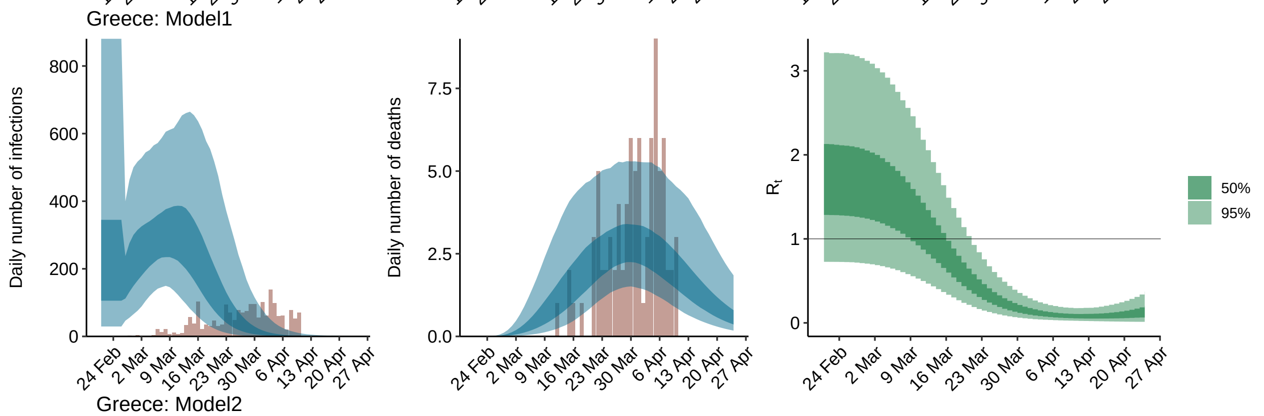

Denmark: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%

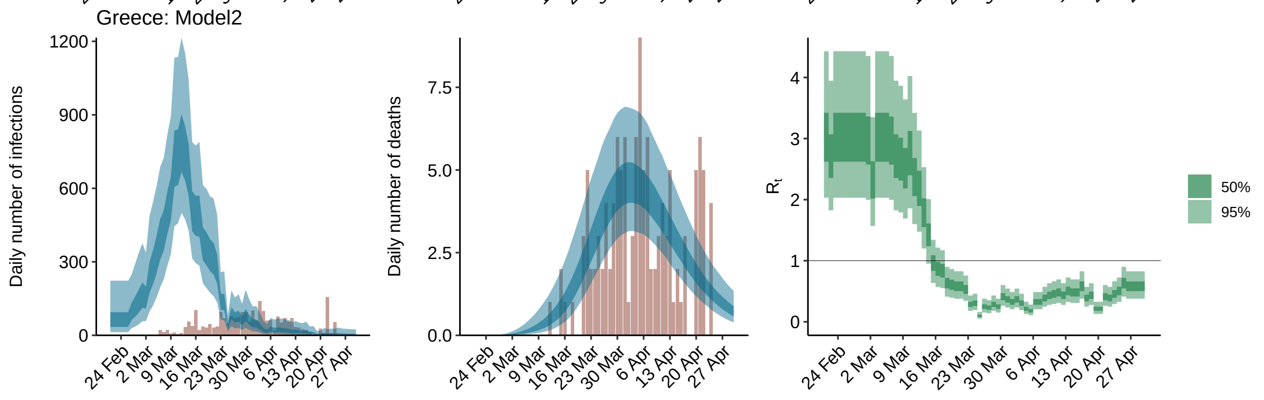

Denmark: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr

Finland: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr

Finland: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr

Finland: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr

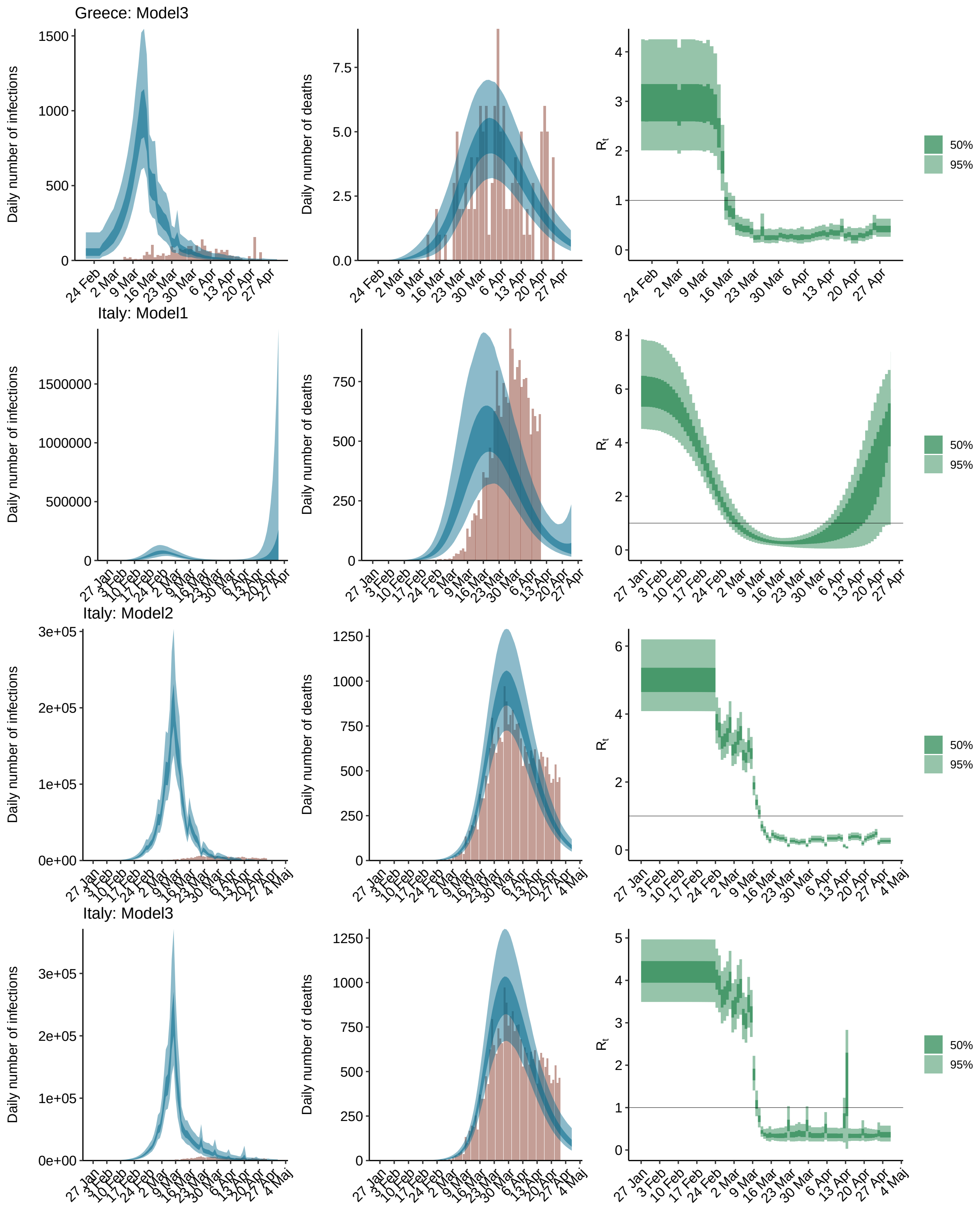

France: Model1
France: Model2
France: Model3
Germany: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%
3 Feb
10 Feb
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj
60000
40000
20000
0
2000
1500
1000
500
3e+05
2e+05
1e+05
0e+00
16000
12000
8000
4000
250
200
150
100
50
6
4
2

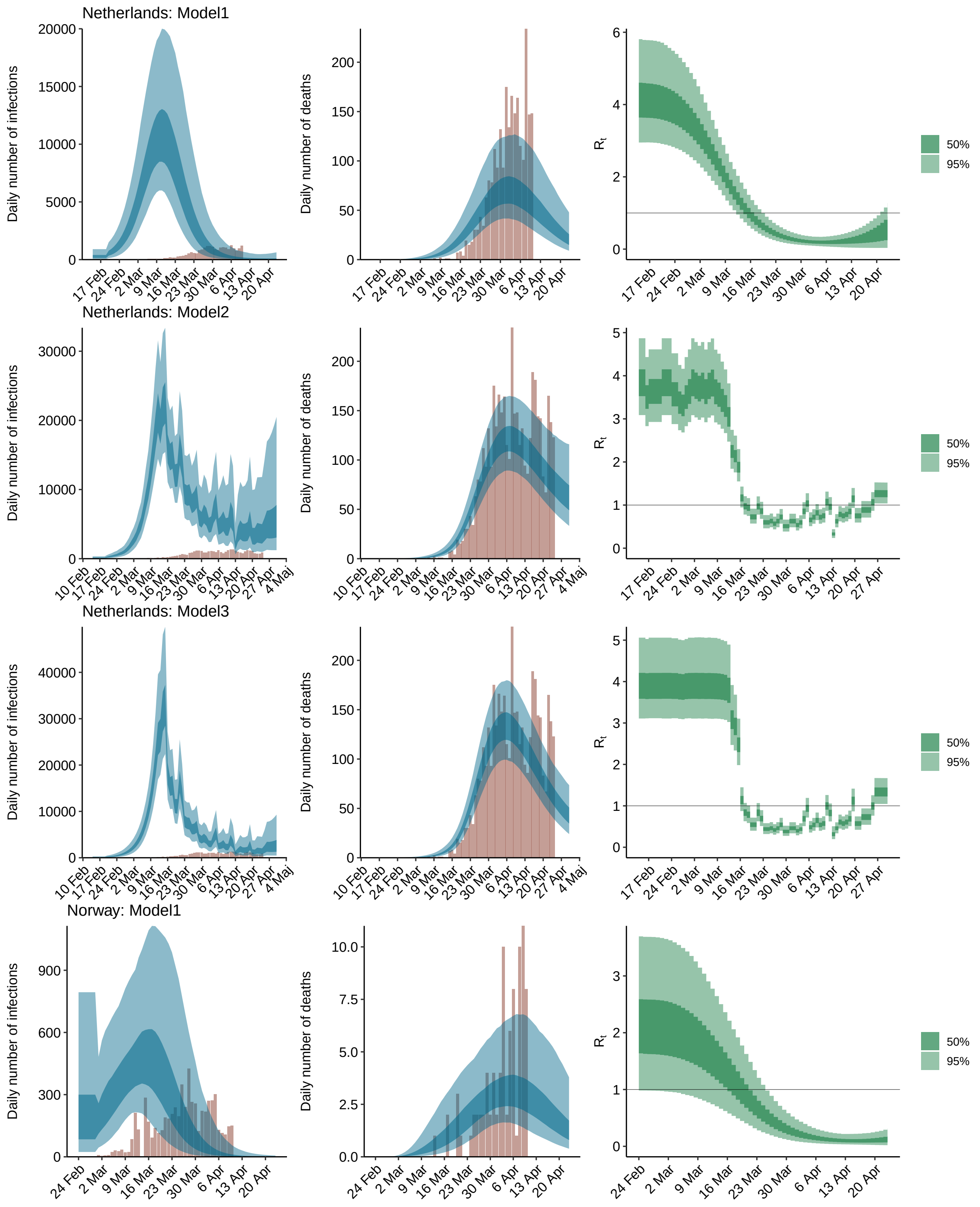

Germany: Model2
Daily number of infections
0e+00
5e+04
1e+05
Daily number of deaths
0
100
200
300
R_t
0
1
2
3
4
5
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj
50%
95%


Germany: Model3
Daily number of infections
0
50000
100000
150000
Daily number of deaths
0
100
200
300
R_t
0
1
2
3
4
5
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj
50%
95%

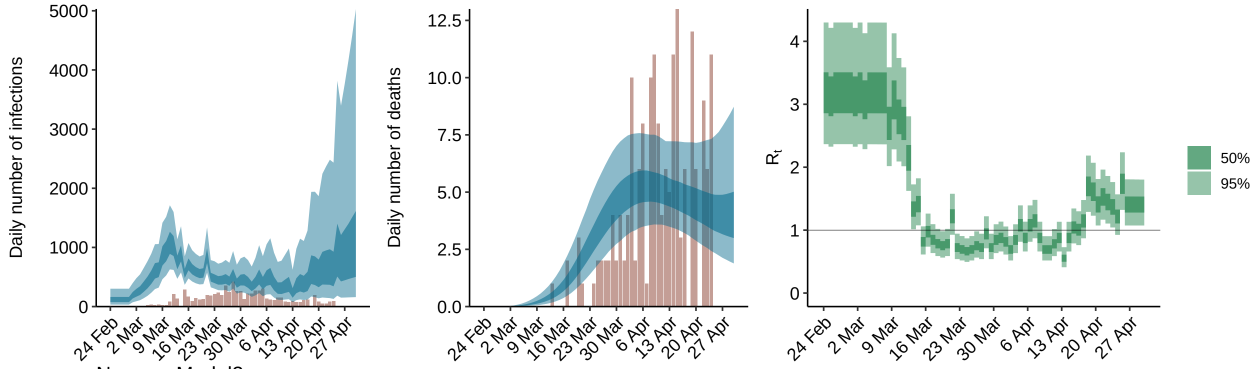

Greece: Model1
Daily number of infections
0
200
400
600
800
Daily number of deaths
0.0
2.5
5.0
7.5
R_t
0
1
2
3
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
50%
95%


Greece: Model2
Daily number of infections
0
300
600
900
1200
Daily number of deaths
0.0
2.5
5.0
7.5
R_t
0
1
2
3
4
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
50%
95%

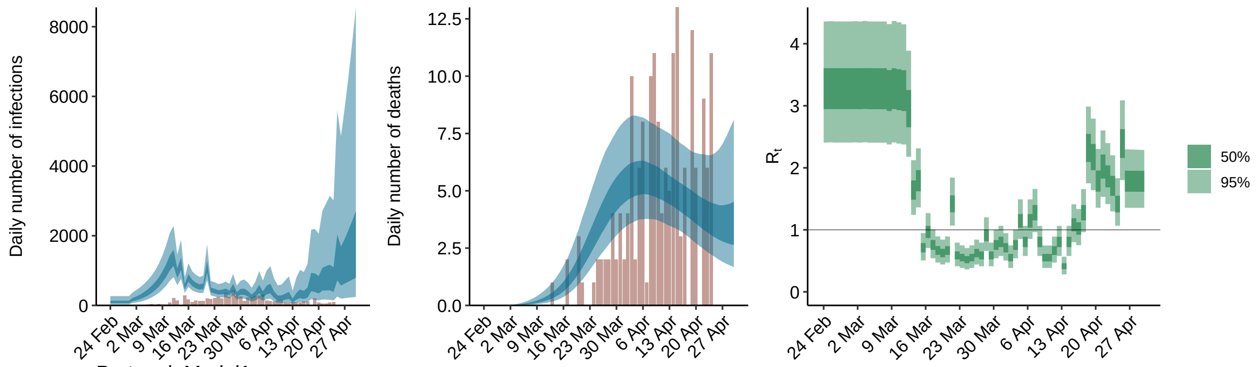

Greece: Model3
Italy: Model1
Italy: Model2
Italy: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%


Netherlands: Model1
Netherlands: Model2
Netherlands: Model3
Norway: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%
10 Feb
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj

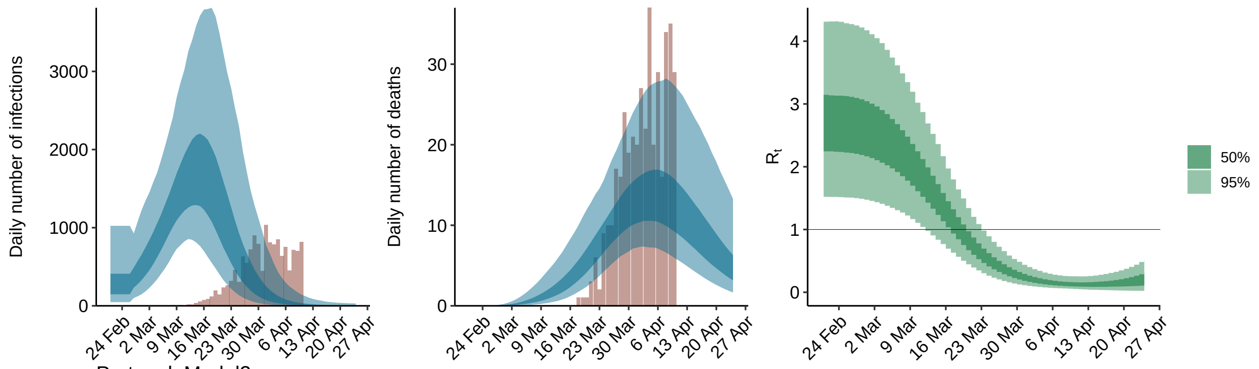
Norway: Model2



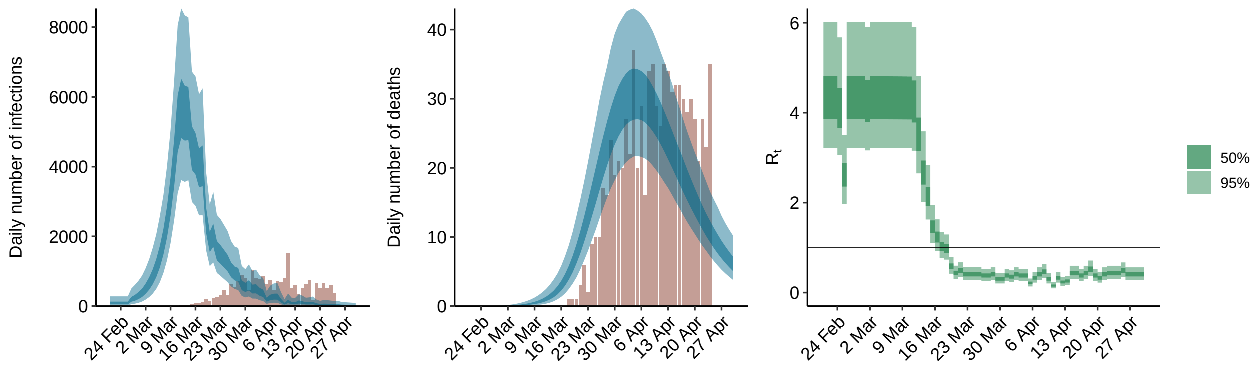
Norway: Model3

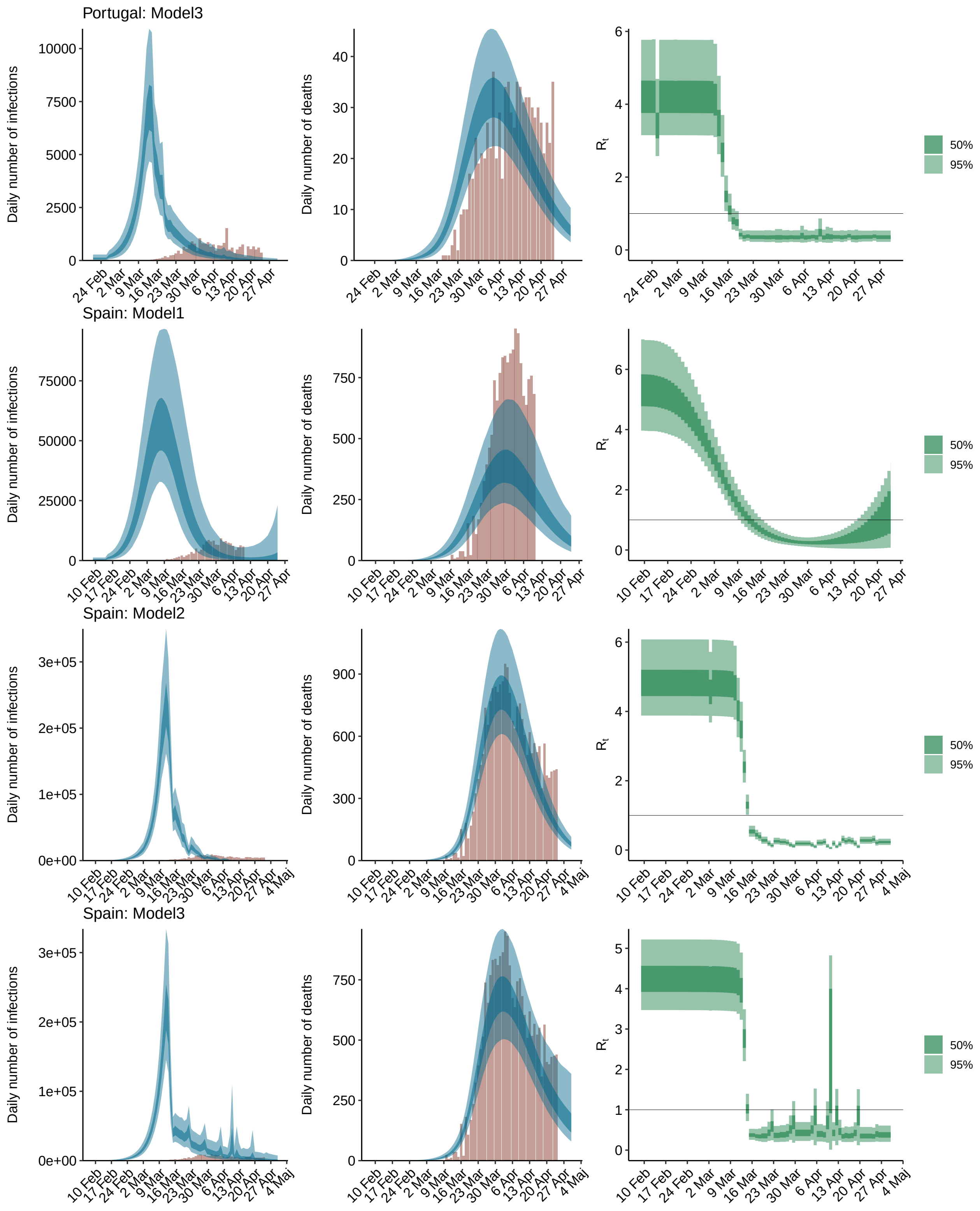


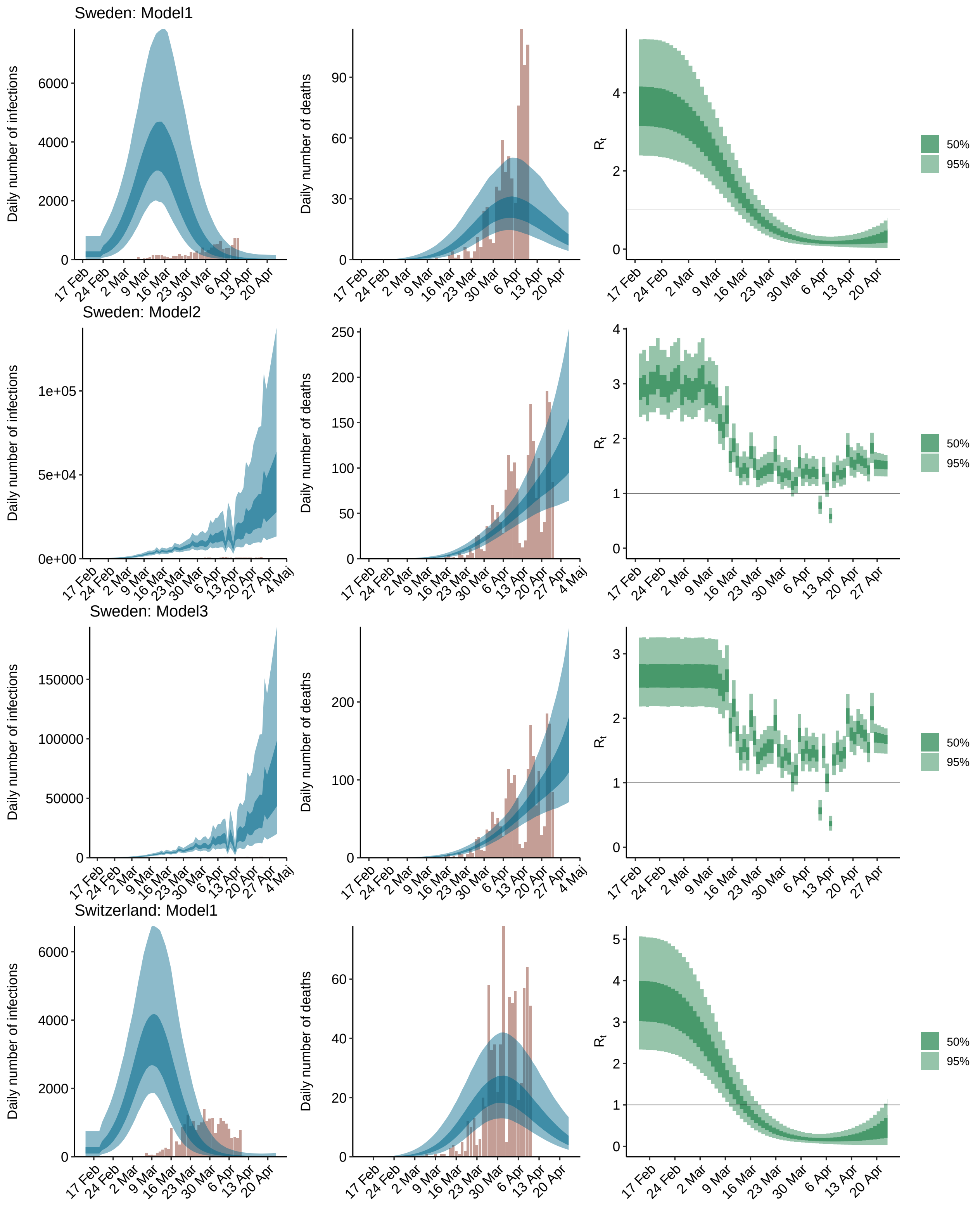
Portugal: Model1

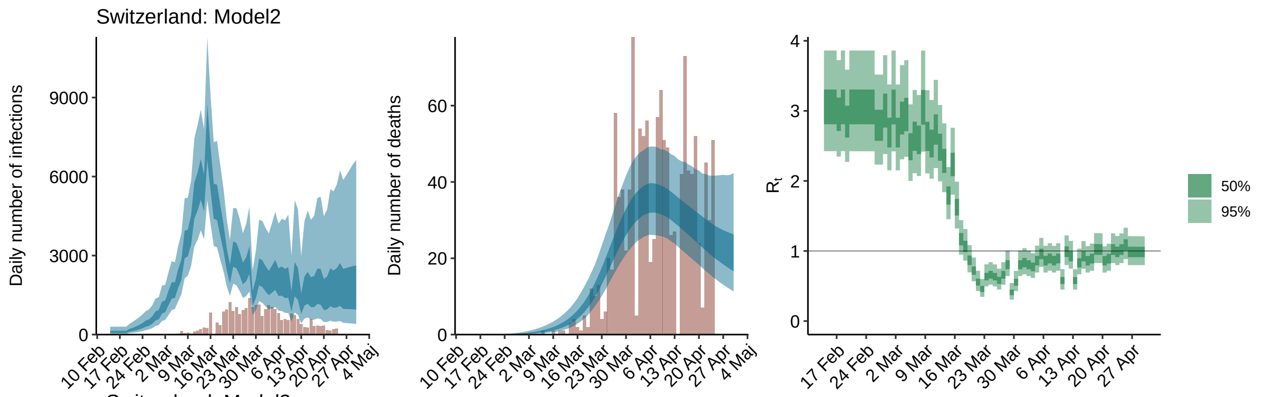


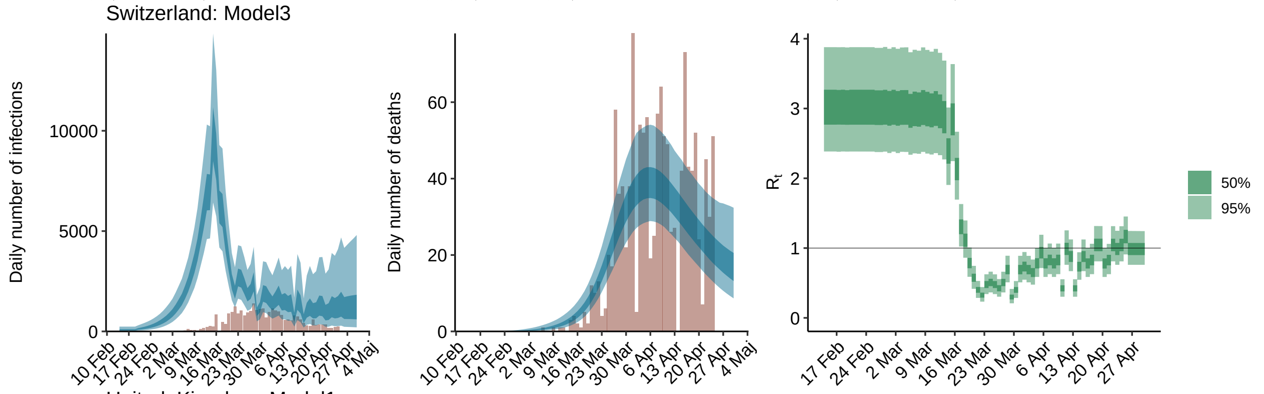
Portugal: Model2

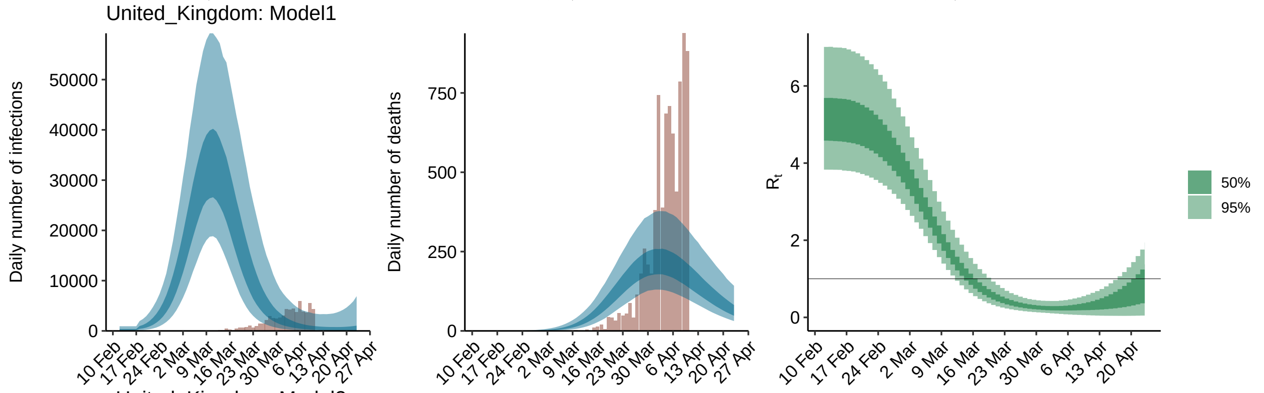


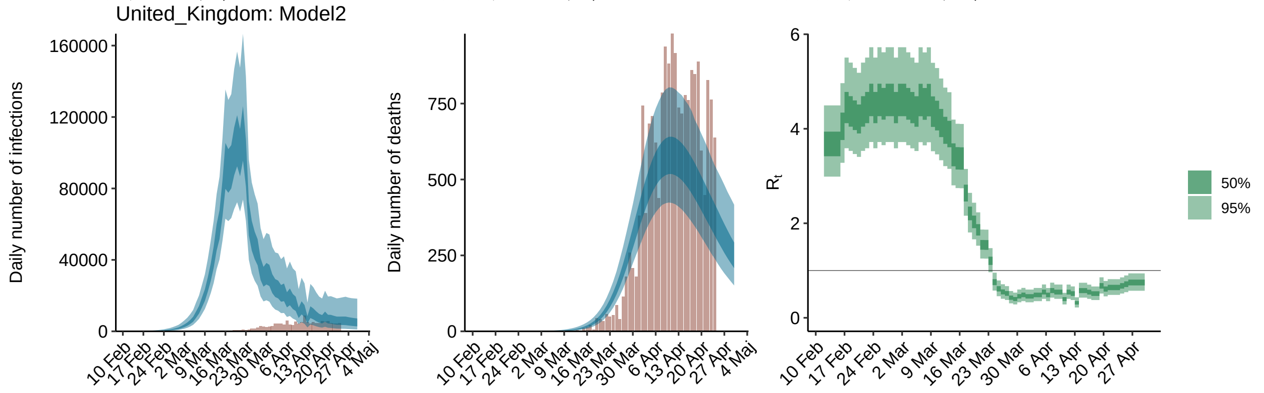

Portugal: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%
Spain: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%
Spain: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%
Spain: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%
10 Feb
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj

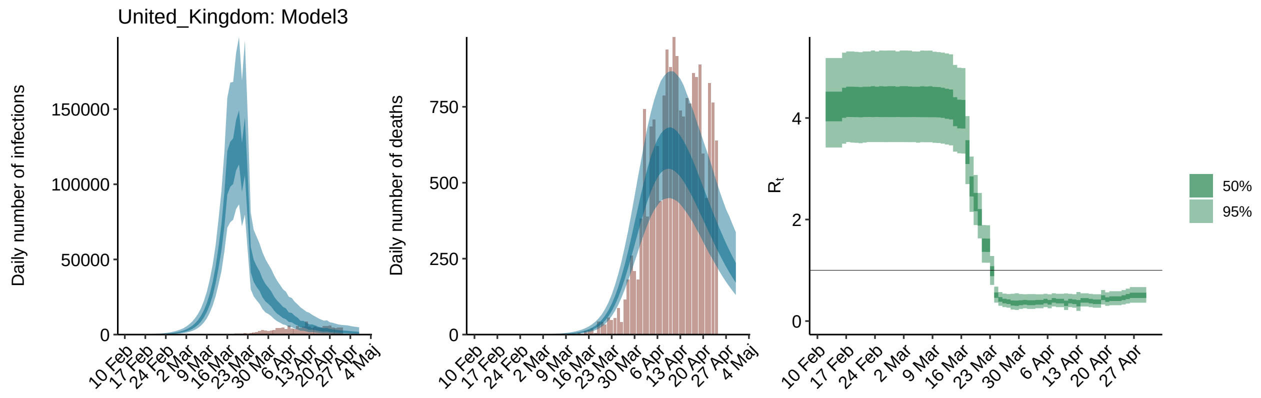

Sweden: Model1
Sweden: Model2
Sweden: Model3
Switzerland: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%
17 Feb
24 Feb
2 Mar
9 Mar
16 Mar
23 Mar
30 Mar
6 Apr
13 Apr
20 Apr
27 Apr
4 Maj


Switzerland: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%


Switzerland: Model3
Daily number of infections
Daily number of deaths
R_t
50%
95%


United_Kingdom: Model1
Daily number of infections
Daily number of deaths
R_t
50%
95%


United_Kingdom: Model2
Daily number of infections
Daily number of deaths
R_t
50%
95%



Stan code

```
## S4 class stanmodel 'base_general_speed' coded as follows:
## data {
##   int <lower=1> M; // number of countries
##   int <lower=1> P; // number of covariates
##   int <lower=1> N0; // number of days for which to impute infections
##   int<lower=1> N[M]; // days of observed data for country m. each entry must be <= N2
##   int<lower=1> N2; // days of observed data + # of days to forecast
##   int cases[N2,M]; // reported cases
##   int deaths[N2, M]; // reported deaths -- the rows with i > N contain -1 and should be ignored
##   matrix[N2, M] f; // h * s
##   matrix[N2, P] X[M];
##   int EpidemicStart[M];
##   real pop[M];
##   real SI[N2]; // fixed pre-calculated SI using emprical data from Neil
## }
##
## transformed data {
##   vector[N2] SI_rev; // SI in reverse order
##   vector[N2] f_rev[M]; // f in reversed order
##
##   for(i in 1:N2)
##     SI_rev[i] = SI[N2-i+1];
##
##   for(m in 1:M){
##     for(i in 1:N2) {
##      f_rev[m, i] = f[N2-i+1,m];
##     }
##   }
## }
##
##
## parameters {
##   real<lower=0> mu[M]; // intercept for Rt
##   real<lower=0> alpha_hier[P]; // sudo parameter for the hier term for alpha
##   real<lower=0> kappa;
##   real<lower=0> y[M];
##   real<lower=0> phi;
##   real<lower=0> tau;
##   real <lower=0> ifr_noise[M];
## }
##
## transformed parameters {
##     vector[P] alpha;
##     matrix[N2, M] prediction = rep_matrix(0,N2,M);
##     matrix[N2, M] E_deaths  = rep_matrix(0,N2,M);
##     matrix[N2, M] Rt = rep_matrix(0,N2,M);
##     matrix[N2, M] Rt_adj = Rt;
##
##     {
##       matrix[N2,M] cumm_sum = rep_matrix(0,N2,M);
##       for(i in 1:P){
##         alpha[i] = alpha_hier[i] - ( log(1.05) / 6.0 );
##       }
##       for (m in 1:M){
##         /*
##         for (i in 2:N0){
##           cumm_sum[i,m] = cumm_sum[i-1,m] + y[m];
##         }
##         */
##         prediction[1:N0,m] = rep_vector(y[m],N0); // learn the number of cases in the first N0 days
##         cumm_sum[2:N0,m] = cumulative_sum(prediction[2:N0,m]);
##
##         Rt[,m] = mu[m] * exp(-X[m] * alpha);
##           Rt_adj[1:N0,m] = Rt[1:N0,m];
##         for (i in (N0+1):N2) {
##           /*
##           real convolution=0;
##           for(j in 1:(i-1)) {
##             convolution += prediction[j, m] * SI[i-j];
##           }
##           */
##           real convolution = dot_product(sub_col(prediction, 1, m, i-1), tail(SI_rev, i-1));
##
##           cumm_sum[i,m] = cumm_sum[i-1,m] + prediction[i-1,m];
##           Rt_adj[i,m] = ((pop[m]-cumm_sum[i,m]) / pop[m]) * Rt[i,m];
##           prediction[i, m] = Rt_adj[i,m] * convolution;
##         }
##
##         E_deaths[1, m]= 1e-15 * prediction[1,m];
##         for (i in 2:N2){
##           // for(j in 1:(i-1)){
##           //   E_deaths[i,m] += prediction[j,m] * f[i-j,m] * ifr_noise[m];
##           // }
##           E_deaths[i,m] = ifr_noise[m] * dot_product(sub_col(prediction, 1, m, i-1), tail(f_rev[m], 
##         }
##       }
##     }
## }
## model {
##   tau ~ exponential(0.03);
##   for (m in 1:M){
##       y[m] ~ exponential(1/tau);
##   }
##   phi ~ normal(0,5);
##   kappa ~ normal(0,0.5);
##   mu ~ normal(3.28, kappa); // citation: https://academic.oup.com/jtm/article/27/2/taaa021/5735319
##   alpha_hier ~ gamma(.1667,1);
##   ifr_noise ~ normal(1,0.1);
##   for(m in 1:M){
##     deaths[EpidemicStart[m]:N[m], m] ~ neg_binomial_2(E_deaths[EpidemicStart[m]:N[m], m], phi);
##    }
## }
##
## generated quantities {
##     matrix[N2, M] prediction0 = rep_matrix(0,N2,M);
##     matrix[N2, M] E_deaths0  = rep_matrix(0,N2,M);
##
##     {
##       matrix[N2,M] cumm_sum0 = rep_matrix(0,N2,M);
##       for (m in 1:M){
##          for (i in 2:N0){
##           cumm_sum0[i,m] = cumm_sum0[i-1,m] + y[m];
##         }
##         prediction0[1:N0,m] = rep_vector(y[m],N0);
##         for (i in (N0+1):N2) {
##           real convolution0 = 0;
##           for(j in 1:(i-1)) {
##             convolution0 += prediction0[j, m] * SI[i-j];
##           }
##           cumm_sum0[i,m] = cumm_sum0[i-1,m] + prediction0[i-1,m];
##           prediction0[i, m] =  ((pop[m]-cumm_sum0[i,m]) / pop[m]) * mu[m] * convolution0;
##         }
##
##         E_deaths0[1, m] = uniform_rng(1e-16, 1e-15);
##         for (i in 2:N2){
##           for(j in 1:(i-1)){
##             E_deaths0[i,m] += prediction0[j,m] * f[i-j,m] * ifr_noise[m];
##           }
##         }
##       }
##     }
## }
##
```