Comparison of Methods : Compartmental Models and Renewal Models

The spread of infectious diseases in a given setup, provided the starting point, the source, the vulnerable groups, the rate of transmissions and recovery and many other catalysts is an interesting phenomenon to model and make inferences upon.
Two mainstream ways to go about modelling the same is by relying on Mathematical models like Compartmental Models and Statistical Models like the Renewal Models.
In this write up, the focus would be on drawing a parallel between the two methods, to see them in their light of merits and demerits and give a suggestion as to which would be an ideal choice given the scenario and availability of the data.

The Process

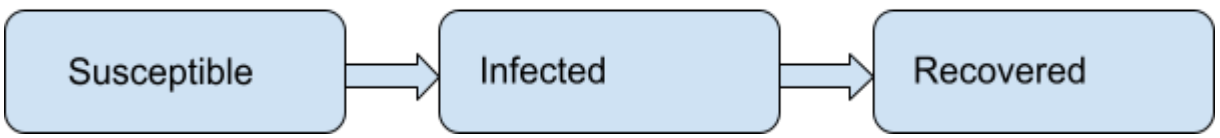
For modelling the spread of an infectious disease, the lifecycle of the problem statement typically follows the following steps:

1) A research question needs to be defined.
2) The population and the time period of the study needs to be stated.
3) To think of a methodology that would produce results that align with real data.
4) Design models either by using compartmental models or renewal models.

Compartmental Models

Throughout history of epidemics and spread of infections, compartmental models have come to our rescue by providing a deterministic way of mimicking the transmission of an infection, in order to make predictions for the upcoming trends and make informed decisions accordingly.
Compartmental model, as the name suggests, divides the population into different compartments exclusively, depending on which state of interaction with infection they are at. These models are governed by a set of differential equations that formulate the movement of between these compartments by employing various parameters in them.
The simplest model, that we would be discussing here , is called
Susceptible-Infected-Recovered, abbreviated as SIR Model.
Depending on the infection, its multiplicity, fatality, incubation period and various other characteristics, the simple SIR model can have various other compartments and parameters added to it to model the infection accordingly, such as SEIR, SEIRD etc.

THE SIR MODEL



The SIR model as the name suggests has three states, which are dependent on time:

Susceptible ($S(t)$) : Denotes the count from the population that is not yet infected but is susceptible to the disease.
Infected ($I(t)$) : Denotes the count from population who are currently infected and can spread disease to others
Recovered ($R(t)$): Denotes the count from the population who have recovered from the disease and are assumed to be immune and can no longer spread infection.
The total population is represented by N which is nothing but the sum of all the states at a point of time, i.e. $N = S(t) + I(t) + R(t)$

The states at $t = 0$ are called the initial states and are denoted by S_0, I_0, R_0 respectively.

There are two parameters involved in this model:

β : Transmission Rate : Represents the average number of contacts per person per time resulting in transmission of the disease. Therefore, it is responsible for the rate at which individuals move from state S to state I.

Since transmission depends on number of infectious people, we account for the same by a quantity called as Force of Infection (FOI) which is the average rate that susceptible contact the infection : $\lambda = \beta\frac{1}{N}$

γ : Recovery Rate : Represents the fraction of infected individuals recovering per unit of time. Therefore, it governs the rate at which individuals move from state I to state R.

This also implies that the average duration of infection is $\frac{1}{\gamma}$.

The model is handicapped without making certain underlying assumptions:

1) There is homogenous mixing, which implies the likelihood of any two people coming in contact is the same. It also ensures uniform mixing.
2) Constant population implies that the totality of all the compartments remains the same. Birth and death counts are not considered here.It also assumes no one is dying due to the infection or natural causes.
3) No external influences such as vaccination, or public health interventions.
4) Recovered Individuals gain immunity and do not spread infection ahead in time.
5) No latent period, which means not accounting for the time gap between getting infected but not being infectious yet.
   Due to these assumptions, the SIR model becomes ideal for short term predictions and infectious diseases where people do not lose immunity.

Given the assumptions, the SIR model can simulate how the states will behave over the course of the infection at different time points.

The differential equations that capture the movement across compartments are:

$$\frac{\partial S}{\partial t} = -\ \beta SI$$
$$\frac{\partial I}{\partial t} = \beta SI\ -\ \gamma I$$
$$\frac{\partial R}{\partial t} = \gamma I$$

The reproduction number, R_0 is the expected number of cases directly generated by one case in a population where all individuals are susceptible to the infection. It is calculated by multiplying the transmission rate with the average duration of an infectious disease,i.e., $R_0 = \frac{\beta}{\gamma}$

The parameters for the SIR Model can be assumed or estimated by using various techniques. The former leads to inducing subjective bias and the latter leads to variability in predictions. One way to go about this is by relying on the Bayesian approach to derive an inference and apply MCMC methods for parameter calibration. Other methods include sum of squares, non-linear least squares and maximum likelihood estimation.

Therefore, in the SIR model, a host of assumptions have to be met in order to predict the behaviour of the disease in the entire time period of the pandemic. These assumptions try to mechanistically simulate the same.
This naturally brings the drawbacks to the surface. If the assumptions are not met, the ability to capture the actual behaviour of the pathogen would be inhibited.

Bayesian approach for diseases transmission modelling using Stan:

By employing the SIR Model, the objective is to utilise the Bayesian tools available in Stan in order to develop an efficient workflow for epidemiological models.

The first step is to simulate the data by using the 'deSolve' package. Here the starting point for each of the states were given (Susceptible, Infected and Recovered), along with transmission rate (beta) and recovery rate (gamma).

The hypothetical data is created to model Measles outbreak.

Measles:

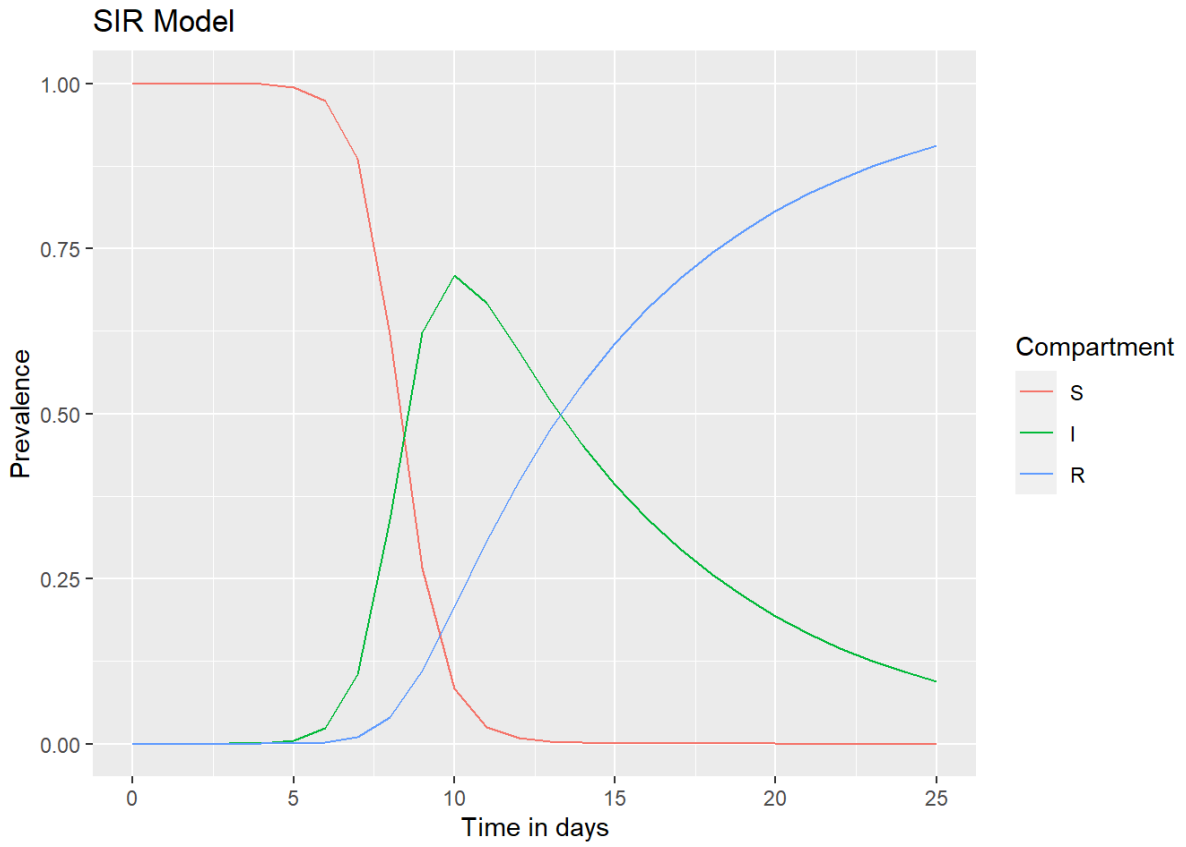
- Initial State: S ≈ 49999900 , I ≈ 100, R ≈ 0
- Population : N = 50000000
- R0: 12

- Gamma: Inverse of the average duration of infectiousness (e.g., 1/7 days if the infectious period is around 7 days).
- Beta: Relatively high, taken to be 1.71 per day (back calculated).

Data Simulation

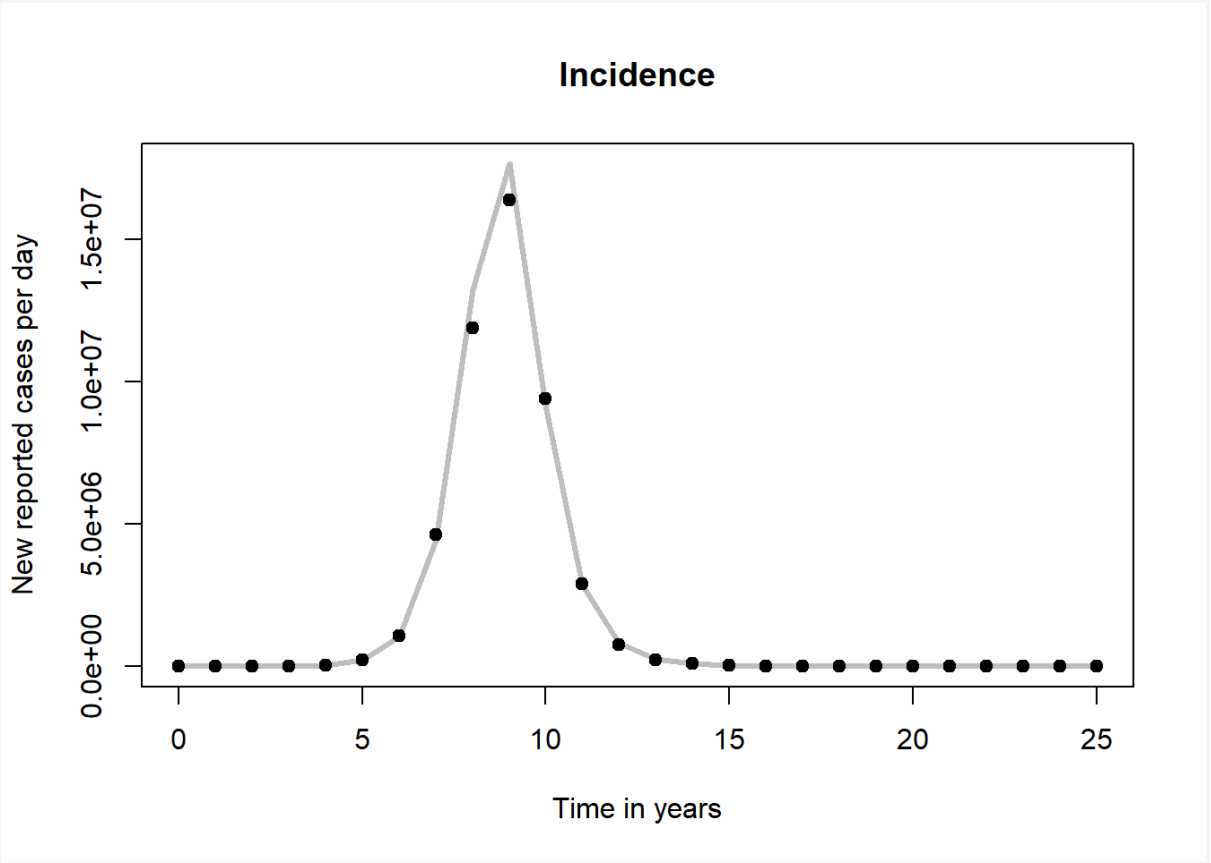
```
#library:
library(deSolve)
library(reshape2)
library(ggplot2)

#inputs
initP<-50000000# population size
initI<-100 # Infectious
initR<-0 # Immune
initS<-initP-initI-initR # Susceptible (non-immune)
R0=12 # Taken value
state <- c(S = initS, I = initI,R = initR)
parameters <- c(beta<-R0*1/7 ,
                gamma<-1/7)

#Timeframe
t_start <- 0
t_stop <- 25
times <- seq(t_start, t_stop)

#Model
sir_model<-function(time, state, parameters){
  with(as.list(c(state,parameters)),{
    N<-S+I+R
    lambda=beta*(I/N)
    dS=-lambda*S
    dI=lambda*S-gamma*I
    dR=gamma*I
    return(list(c(dS,dI,dR)))
    })
}

#Calculation of differential equations
out<-ode(y=state,times=times,func=sir_model,parms=parameters)
out_full<-melt(as.data.frame(out),id='time')
```

```
#The purpose of the melt function is to transform a wide-format
data frame into a long-format data frame.

#Plot
ggplot(out_full,aes(x=time,y=proportion, color=variable,
gropu=variable))+
  geom_line()+
  xlab('Time in days')+
  ylab("Prevalence")+
  labs(color='Compartment',title='SIR Model')
```




```
# total population
pop<-out[,"S"]+out[,"I"]+out[,"R"]
time<-out[,"time"]
# weekly incidence
inc <- 0*time
```

```
inc[2:length(time)]<-
-(out[2:length(time),"S"]-out[1:(length(time)-1),"S"])

# generate simulated data
incdat1<-rnbinom(rep(1,length(times)), size= 200, mu=inc)
#generate data from negative binomial process

# COMPARING THE MODEL out WITH DATA
par(mfrow=c(1,1))
yl <- c(0,max(incdat1,inc))
plot(time,inc,type='l',lwd=3,main = "Incidence",xlab = "Time in
years",ylab="New reported cases per day",ylim=yl,col='grey')
points(time,incdat1,pch=19,col='black')
```



The idea is to conduct an experiment with the underlying assumption that incidence can be derived from the difference between two consecutive 'Susceptible' states.

With the following inputs to the disease model, the time was bound for 25 days. The daily incidence was inferred by taking a difference between two consecutive Susceptible States. Against these incidences, simulated data was generated by using samples from negative binomial distribution. The smooth curve here depicts the incidence, and the points depict the simulated incidence.

Given that this is the data, the next step is to provide these inputs to the Stan model.
The Stan code takes gamma and initial state values as an input to be able to estimate beta value. The Stan code also infers incidence by taking a difference between consecutive Susceptible cases.

Stan Code

```
functions {
  real[] SIR(real t, //time
  real[] y, //system state
  real[] theta, //parameters
  real[] x_r, //real data
  int[] x_i) //integer data
  {
    //define real data
    real gamma = x_r[1];

    // define parameters
    real beta = theta[1];
    real i0= theta[2];

    //define integer data
    int  N = x_i[1];

    //define compartments
    real S; real I; real R;

    //define states
    real dS_dt; real dI_dt; real dR_dt;

    //define FOI
    real FOI;
```

```
    //initial conditions
    S = y[1]; I = y[2]; R = y[3];

    //calculate FOI
    FOI = beta * I / N;

    //ODE
    dS_dt = -FOI * S;
    dI_dt = FOI * S - gamma * I;
    dR_dt = gamma * I;
    return {dS_dt, dI_dt, dR_dt};
  }
}

data {
  int<lower=1> n_ts;    //no of days observed
  real<lower=0> gamma;   //recovery rate
  real ts[n_ts];        //1:n days
  int cases[n_ts];     //incidence
  int N;
  real t0;
}



transformed data {
  int x_i[1] = {N};
  real x_r[1] = {gamma};
}



parameters {
  real<lower=0,upper=50> beta;
  real<lower=0,upper=1000> i0;
}

transformed parameters {
  real y[n_ts, 3];
  real <lower=0> init[3];
  real inc[n_ts];
  {
```

```
    real theta[2];
    theta[1]=beta;
    theta[2]=i0;

    init[1]=N;
    init[2]=i0;
    init[3]=N-i0;

    y=integrate_ode_rk45(SIR, init, t0, ts, theta, x_r, x_i);
  }

  for (t in 1:n_ts-1) {
    inc[t] = -(y[t+1, 1]-y[t, 1]);
  }
  inc[n_ts]=inc[n_ts-1];
}

model {
  //priors
  beta ~ normal(10,8);
  i0~normal(100,5);

  cases~neg_binomial_2(inc, 200);

}


generated quantities {
  real pred_cases[n_ts];
  pred_cases = neg_binomial_2_rng(inc, 200);
  real R_0 = beta / gamma;
}
```

R Code

```
library(rstan)
## Warning: package 'rstan' was built under R version 4.2.3
## Loading required package: StanHeaders
```

```
## Warning: package 'StanHeaders' was built under R version 4.2.3
##
## rstan version 2.32.3 (Stan version 2.26.1)
## For execution on a local, multicore CPU with excess RAM we
recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend
calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()`
Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a
Makevars file
library(readr)

# Compile the Stan model
stan_model <- compiled_model



inc_data <- read_csv("new_data.csv")
cases<-round(inc_data[[2]])
cases<-cases[-1]
N<-50000000
n_ts<-length(cases)
t<-seq(0,25)
t0=0
gamma = 1/7
t<-t[-1]

data_list <- list(n_ts=n_ts,
                  gamma=gamma,
                  ts=t,
                  cases=cases,
                  N = N,
                  t0=t0)



# Set the number of chains and iterations
chains <- 4
```

```
niter <- 2000

# Run MCMC
stan_output <- sampling(stan_model, data = data_list, chains =
chains, iter = niter)
##
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 20.138 seconds (Warm-up)
## Chain 1:                15.128 seconds (Sampling)
## Chain 1:                35.266 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 2:
## Chain 2:  Elapsed Time: 13.626 seconds (Warm-up)
## Chain 2:                10.631 seconds (Sampling)
## Chain 2:                24.257 seconds (Total)
## Chain 2:
##
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 10.683 seconds (Warm-up)
## Chain 3:                9.184 seconds (Sampling)
## Chain 3:                19.867 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 4:
## Chain 4:  Elapsed Time: 8.616 seconds (Warm-up)
## Chain 4:                7.781 seconds (Sampling)
## Chain 4:                16.397 seconds (Total)
## Chain 4:
# Print summary statistics and diagnostics
print(stan_output)
```

The summary table here displays the posterior mean, standard error, quantiles, and some useful diagnostics of the parameters of interest.

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                      mean  se_mean         sd        2.5%
25%
## beta                 1.70     0.00       0.00        1.70
1.70
## i0                  24.24     0.04       1.03       22.33
23.53
## y[1,1]         49999900.55     0.13       3.84 49999892.69
49999898.15
## y[1,2]              115.34     0.16       4.53      106.87
112.26
## y[1,3]         49999984.10     0.02       0.69 49999982.69
49999983.67
## y[2,1]         49999427.34     0.72      20.65 49999384.98
49999414.37
## y[2,2]              548.86     0.69      19.86      511.51
535.45
## y[2,3]         50000023.80     0.03       0.80 50000022.29
50000023.28
## y[3,1]         49997175.56     3.22      94.05 49996982.97
49997116.16
## y[3,2]             2611.71     2.97      86.78     2447.12
2554.30
## y[3,3]         50000212.72     0.25       7.28 50000198.92
50000207.84
```

```
## y[4,1]         49986462.84    13.85     414.30 49985607.08
49986193.86
## y[4,2]            12425.56    12.65     378.78    11706.28
12174.29
## y[4,3]         50001111.60     1.21      35.56 50001044.52
50001087.90
## y[5,1]         49935548.58    58.38    1812.09 49931826.20
49934362.48
## y[5,2]            59065.15    53.15    1653.50    55895.72
57956.02
## y[5,3]         50005386.26     5.24     158.90 50005084.34
50005280.57
## y[6,1]         49694732.27   240.52    7895.79 49678618.09
49689635.35
## y[6,2]           279603.87   218.56    7201.43   265757.52
274873.70
## y[6,3]         50025663.86    21.98     696.38 50024332.45
50025200.22
## y[7,1]         48581210.55   940.38   33517.83 48514366.25
48559352.86
## y[7,2]          1298067.21   851.64   30524.18  1238309.14
1277789.56
## y[7,3]         50120722.24    88.88    3005.62 50114920.75
50118757.76
## y[8,1]         43925024.87  3074.55  123046.27 43681561.04
43843214.46
## y[8,2]          5531628.87  2749.65  111032.58  5313954.26
5457746.92
## y[8,3]         50543346.27   325.99   12081.04 50519784.65
50535351.34
## y[9,1]         30405655.02  5610.62  263470.21 29898087.14
30227021.11
## y[9,2]         17507897.64  4727.93  226942.55 17058969.75
17356127.61
## y[9,3]         52086447.33   890.39   36885.56 52013772.60
52061750.99
## y[10,1]        12984031.37  3548.80  203072.89 12593695.14
12842247.35
## y[10,2]        31360068.87  2261.51  139750.45 31074432.51
31265860.28
```

```
## y[10,3]        55655899.77  1414.38   65178.76 55526483.04
55612309.32
## y[11,1]         4052388.94  1132.06   74608.93  3911205.93
3999864.66
## y[11,2]        35407438.23   784.67   23361.89 35361647.75
35391802.32
## y[11,3]        60540172.83  1486.98   73430.25 60393781.41
60491007.50
## y[12,1]         1247355.73   348.82   23333.05  1202655.54
1230999.61
## y[12,2]        33270200.64  1196.64   49317.94 33173535.36
33236951.74
## y[12,3]        65482443.63  1343.93   69028.89 65343972.54
65436729.45
## y[13,1]          427098.88   122.60    7953.44   411853.57
421561.74
## y[13,2]        29594986.12  1162.49   55169.89 29488751.22
29557533.38
## y[13,3]        69977915.00  1173.16   61439.23 69854471.13
69937041.23
## y[14,1]          166130.59    55.73    3113.84   160179.83
163979.97
## y[14,2]        25895360.48  1027.27   51581.06 25796107.35
25859783.73
## y[14,3]        73938508.93  1016.68   53779.64 73829967.62
73902742.74
## y[15,1]           72906.37    29.24    1393.36    70261.25
71960.01
## y[15,2]        22534006.09   889.81   45947.44 22445180.97
22502093.31
## y[15,3]        77393087.53   880.03   46810.00 77298802.67
77361995.00
## y[16,1]           35634.80    16.29     700.22    34308.12
35160.81
## y[16,2]        19568613.37   768.79   40280.11 19490349.23
19540686.79
## y[16,3]        80395751.83   761.81   40656.18 80313844.08
80368772.50
## y[17,1]           19143.70     9.74     388.21    18407.28
18883.18
```

```
## y[17,2]        16978827.43   664.42   35098.90 16910671.41
16954420.10
## y[17,3]        83002028.87   659.63   35279.19 82930936.72
82978595.48
## y[18,1]           11167.15     6.20     233.81    10726.48
11009.49
## y[18,2]        14725943.41   574.62   30506.27 14666751.72
14704703.07
## y[18,3]        85262889.43   571.30   30600.29 85201216.44
85242581.15
## y[19,1]            6997.48     4.18     151.05     6715.60
6893.86
## y[19,2]        12769452.39   497.24   26483.37 12718101.65
12750989.38
## y[19,3]        87223550.13   494.87   26536.17 87170063.12
87205952.11
## y[20,1]            4665.76     2.95     103.60     4472.79
4594.95
## y[20,2]        11071714.79   430.47   22977.44 11027168.29
11055682.51
## y[20,3]        88923619.45   428.73   23009.10 88877239.29
88908359.50
## y[21,1]            3283.30     2.19      74.79     3144.47
3231.76
## y[21,2]         9599105.55   372.78   19929.35  9560478.19
9585164.04
## y[21,3]        90397611.15   371.45   19949.46 90357396.71
90384383.84
## y[22,1]            2421.03     1.68      56.42     2316.36
2382.25
## y[22,2]         8322051.72   322.89   17282.50  8288560.89
8309962.57
## y[22,3]        91675527.24   321.85   17295.94 91640660.56
91664070.97
## y[23,1]            1859.07     1.34      44.20     1777.42
1828.75
## y[23,2]         7214723.84   279.72   14985.58  7185688.42
7204241.03
## y[23,3]        92783417.09   278.88   14994.98 92753187.98
92773487.91
```

```
## y[24,1]           1478.61     1.09      35.78     1412.73
1454.06
## y[24,2]        6254637.57   242.35   12993.06  6229465.78
6245546.17
## y[24,3]       93743883.83   241.65   12999.89 93717675.99
93735273.14
## y[25,1]           1212.39     0.92      29.79     1157.75
1191.98
## y[25,2]        5422254.23   209.99   11264.95  5400432.65
5414372.84
## y[25,3]       94576533.37   209.40   11270.10 94553812.13
94569065.68
## init[1]       50000000.00      NaN       0.00 50000000.00
50000000.00
## init[2]             24.24     0.04       1.03       22.33
23.53
## init[3]       49999975.76     0.04       1.03 49999973.66
49999975.12
## inc[1]              473.22     0.58      16.82      441.64
461.84
## inc[2]             2251.77     2.50      73.46     2112.77
2203.07
## inc[3]            10712.72    10.64     320.68    10102.14
10499.52
## inc[4]            50914.26    44.55    1400.56    48232.67
49980.16
## inc[5]           240816.31   182.30    6100.67   229089.65
236808.24
## inc[6]          1113521.73   701.01   25722.01  1063418.64
1096390.48
## inc[7]          4656185.68  2143.32   90093.06  4478331.58
4596440.95
## inc[8]         13519369.84  2588.53  143734.64 13230304.50
13424135.97
## inc[9]         17421623.66  2286.80   73240.74 17271634.44
17373679.44
## inc[10]         8931642.42  2552.60  130435.33  8680473.64
8841256.24
## inc[11]         2805033.21   831.68   51833.89  2706191.13
2768758.38
```

```
## inc[12]          820256.85   232.75   15554.70   790703.38
809236.94
## inc[13]          260968.29    73.84    4898.40   251573.20
257610.25
## inc[14]           93224.22    27.82    1741.56    89889.46
92015.01
## inc[15]           37271.58    12.91     701.24    35938.10
36788.47
## inc[16]           16491.10     6.61     315.38    15892.39
16277.08
## inc[17]            7976.55     3.56     155.90     7681.49
7870.86
## inc[18]            4169.67     2.04      83.48     4011.42
4114.26
## inc[19]            2331.72     1.23      47.83     2241.63
2299.48
## inc[20]            1382.46     0.77      29.02     1327.76
1362.83
## inc[21]             862.27     0.51      18.49      827.78
849.69
## inc[22]             561.97     0.34      12.29      539.06
553.54
## inc[23]             380.46     0.24       8.47      364.66
374.68
## inc[24]             266.21     0.17       6.02      255.02
262.07
## inc[25]             266.21     0.17       6.02      255.02
262.07
## pred_cases[1]       473.14     0.81      43.37      392.00
443.00
## pred_cases[2]      2253.34     3.40     180.82     1912.00
2129.00
## pred_cases[3]     10737.54    16.41     839.49     9128.98
10162.75
## pred_cases[4]     50967.93    70.36    3919.85    43651.68
48275.75
## pred_cases[5]    240571.14   322.91   18282.85   206717.22
227799.50
## pred_cases[6]   1114399.40  1373.07   82026.38   962971.95
1057802.50
```

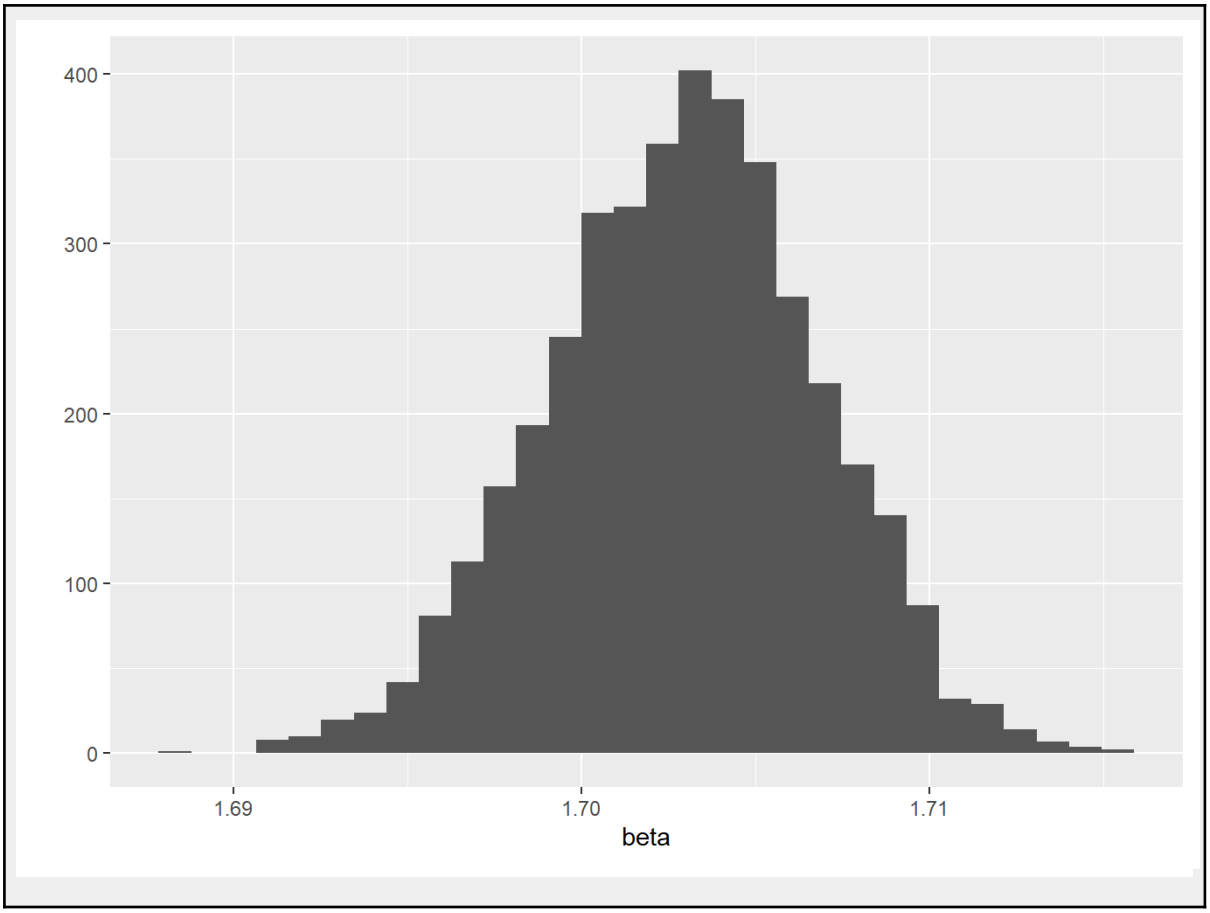
```
## pred_cases[7]   4658184.70  5492.14  338927.09  4031299.48
4426008.50
## pred_cases[8]  13518857.13 15343.53  956350.89 11715978.65
12866916.75
## pred_cases[9]  17438790.38 19678.99 1257632.36 15110524.45
16586181.50
## pred_cases[10]  8924330.66 10032.06  634989.81  7772115.12
8479330.00
## pred_cases[11]  2800895.94  3260.37  203739.59  2411209.08
2665257.00
## pred_cases[12]   820709.25   952.40   59621.79   707306.20
779405.00
## pred_cases[13]   261259.04   307.22   19576.29   224510.72
247759.50
## pred_cases[14]    93328.10   108.93    6823.37    80092.90
88771.00
## pred_cases[15]    37296.26    42.74    2767.70    32185.75
35382.00
## pred_cases[16]    16488.65    20.12    1209.99    14190.85
15647.75
## pred_cases[17]     7996.14     9.75     583.96     6832.92
7598.00
## pred_cases[18]     4168.64     5.08     312.81     3575.95
3949.00
## pred_cases[19]     2335.29     3.04     179.98     2004.95
2210.00
## pred_cases[20]     1382.05     1.85     106.91     1182.00
1310.00
## pred_cases[21]      861.56     1.16      69.98      727.00
815.75
## pred_cases[22]      563.00     0.78      48.03      471.00
530.00
## pred_cases[23]      380.90     0.61      35.03      313.00
357.00
## pred_cases[24]      265.60     0.41      25.96      215.00
248.00
## pred_cases[25]      266.52     0.41      25.83      218.00
249.00
## R_0                 11.92     0.00       0.03       11.87
11.90
```

```
## lp__              -30773.82     0.03       0.98   -30776.46
-30774.23
##                        50%         75%       97.5% n_eff Rhat
## beta                  1.70        1.71        1.71   957 1.01
## i0                   24.20       24.88       26.34   800 1.01
## y[1,1]         49999900.73 49999903.17 49999907.76   813 1.01
## y[1,2]              115.15      118.19      124.61   810 1.01
## y[1,3]         49999984.14 49999984.58 49999985.39   798 1.01
## y[2,1]         49999428.14 49999441.28 49999466.21   827 1.01
## y[2,2]              548.07      561.31      589.61   826 1.01
## y[2,3]         50000023.78 50000024.31 50000025.43   851 1.01
## y[3,1]         49997178.30 49997237.80 49997354.01   852 1.01
## y[3,2]             2609.25     2666.59     2789.33   853 1.01
## y[3,3]         50000212.52 50000217.26 50000227.70   843 1.01
## y[4,1]         49986471.17 49986737.42 49987249.05   894 1.01
## y[4,2]            12417.73    12671.01    13207.15   897 1.01
## y[4,3]         50001110.75 50001134.63 50001184.35   870 1.01
## y[5,1]         49935567.73 49936760.73 49939023.53   963 1.01
## y[5,2]            59051.21    60148.24    62465.81   968 1.01
## y[5,3]         50005383.83 50005489.04 50005714.10   920 1.01
## y[6,1]         49694749.33 49699902.53 49709890.55  1078 1.00
## y[6,2]           279589.18   284235.87   294275.16  1086 1.00
## y[6,3]         50025658.54 50026122.03 50027084.31  1004 1.00
## y[7,1]         48581223.73 48603288.62 48646839.62  1270 1.00
## y[7,2]          1298036.39  1317936.67  1358814.51  1285 1.00
## y[7,3]         50120714.25 50122670.35 50126844.07  1144 1.00
## y[8,1]         43924847.10 44006958.44 44165182.97  1602 1.00
## y[8,2]          5532062.49  5605565.69  5751106.88  1631 1.00
## y[8,3]         50543367.30 50551274.24 50567232.16  1373 1.00
## y[9,1]         30405557.22 30581353.60 30928351.63  2205 1.00
## y[9,2]         17507102.60 17661969.77 17944233.10  2304 1.00
## y[9,3]         52086719.34 52111291.29 52159207.49  1716 1.00
## y[10,1]        12983424.90 13118477.10 13397122.08  3274 1.00
## y[10,2]        31361752.49 31457915.59 31624468.45  3819 1.00
## y[10,3]        55655919.50 55700009.70 55782016.76  2124 1.00
## y[11,1]         4051929.79  4102713.06  4205944.07  4343 1.00
## y[11,2]        35408411.13 35423284.71 35450280.89   886 1.01
## y[11,3]        60539805.37 60590446.94 60681580.65  2439 1.00
## y[12,1]         1246952.86  1262972.14  1294780.90  4474 1.00
## y[12,2]        33269628.87 33302801.71 33367954.80  1699 1.00
## y[12,3]        65482582.71 65530274.70 65615982.98  2638 1.00
```

```
## y[13,1]          427069.41   432257.58   443332.29  4208 1.00
## y[13,2]        29594905.17 29631766.20 29704212.84  2252 1.00
## y[13,3]        69978323.86 70020492.60 70097268.42  2743 1.00
## y[14,1]          166125.15   168139.56   172599.59  3122 1.00
## y[14,2]        25895413.43 25929918.22 25998085.44  2521 1.00
## y[14,3]        73938886.29 73975939.08 74042921.84  2798 1.00
## y[15,1]           72897.53    73813.05    75720.87  2270 1.00
## y[15,2]        22533856.42 22564484.29 22626117.86  2666 1.00
## y[15,3]        77393357.85 77425707.24 77483886.14  2829 1.00
## y[16,1]           35630.00    36098.54    37055.06  1847 1.00
## y[16,2]        19568342.10 19595398.45 19649574.92  2745 1.00
## y[16,3]        80395981.54 80424114.75 80474574.82  2848 1.00
## y[17,1]           19138.83    19399.49    19940.33  1589 1.01
## y[17,2]        16978613.83 17002198.31 17049644.26  2791 1.00
## y[17,3]        83002232.51 83026710.67 83070403.20  2860 1.00
## y[18,1]           11165.18    11321.69    11645.69  1422 1.01
## y[18,2]        14725772.17 14746239.68 14787453.97  2819 1.00
## y[18,3]        85263024.47 85284291.88 85322181.18  2869 1.00
## y[19,1]            6994.38     7098.05     7306.50  1309 1.01
## y[19,2]        12769300.23 12787018.05 12822787.60  2837 1.00
## y[19,3]        87223664.45 87242118.27 87274957.74  2875 1.00
## y[20,1]            4663.51     4735.38     4878.63  1229 1.01
## y[20,2]        11071585.13 11086964.12 11118007.56  2849 1.00
## y[20,3]        88923724.12 88939714.52 88968179.96  2880 1.00
## y[21,1]            3281.94     3333.70     3436.73  1171 1.01
## y[21,2]         9598994.59  9612346.65  9639264.26  2858 1.00
## y[21,3]        90397705.15 90411561.88 90436221.61  2884 1.00
## y[22,1]            2420.12     2458.87     2536.09  1128 1.01
## y[22,2]         8321967.02  8333525.21  8356881.34  2865 1.00
## y[22,3]        91675610.26 91687621.05 91708983.78  2888 1.00
## y[23,1]            1858.58     1888.73     1948.96  1095 1.01
## y[23,2]         7214658.28  7224667.98  7244926.95  2870 1.00
## y[23,3]        92783491.79 92793898.73 92812408.45  2891 1.00
## y[24,1]            1478.18     1502.48     1551.50  1070 1.01
## y[24,2]         6254582.28  6263255.28  6280826.42  2874 1.00
## y[24,3]        93743950.48 93752967.84 93769006.82  2894 1.00
## y[25,1]            1211.85     1232.24     1272.86  1050 1.01
## y[25,2]         5422204.40  5429723.53  5444960.79  2878 1.00
## y[25,3]        94576592.68 94584407.85 94598314.46  2897 1.00
## init[1]        50000000.00 50000000.00 50000000.00   NaN  NaN
## init[2]              24.20       24.88       26.34   800 1.01
```

```
## init[3]        49999975.80 49999976.47 49999977.67   800 1.01
## inc[1]              472.61      483.71      507.70   831 1.01
## inc[2]             2249.74     2298.53     2402.16   861 1.01
## inc[3]            10707.00    10919.44    11374.14   909 1.01
## inc[4]            50901.54    51840.98    53802.34   988 1.00
## inc[5]           240796.56   244746.01   253303.10  1120 1.00
## inc[6]          1113502.14  1130361.48  1164329.61  1346 1.00
## inc[7]          4657022.56  4716642.05  4834188.11  1767 1.00
## inc[8]         13520100.62 13619542.60 13796336.23  3083 1.00
## inc[9]         17422297.93 17470167.11 17562170.03  1026 1.00
## inc[10]         8931013.12  9018026.53  9193657.49  2611 1.00
## inc[11]         2804520.47  2840006.10  2911049.94  3884 1.00
## inc[12]          820068.43   830764.23   852184.44  4466 1.00
## inc[13]          260927.75   264192.47   270923.76  4401 1.00
## inc[14]           93208.49    94353.69    96783.47  3918 1.00
## inc[15]           37272.83    37719.18    38717.17  2949 1.00
## inc[16]           16489.30    16696.29    17127.66  2274 1.00
## inc[17]            7975.16     8079.04     8291.71  1913 1.00
## inc[18]            4169.02     4225.51     4339.92  1682 1.00
## inc[19]            2331.09     2363.12     2429.85  1524 1.01
## inc[20]            1382.14     1401.58     1441.81  1412 1.01
## inc[21]             861.92      874.53      900.11  1331 1.01
## inc[22]             561.67      570.15      587.23  1270 1.01
## inc[23]             380.29      386.14      397.87  1224 1.01
## inc[24]             266.08      270.27      278.58  1188 1.01
## inc[25]             266.08      270.27      278.58  1188 1.01
## pred_cases[1]       472.00      501.00      563.00  2901 1.00
## pred_cases[2]      2247.50     2374.00     2617.02  2836 1.00
## pred_cases[3]     10727.00    11282.50    12456.53  2617 1.00
## pred_cases[4]     50757.00    53551.00    59129.50  3103 1.00
## pred_cases[5]    240075.50   252892.25   276875.00  3206 1.00
## pred_cases[6]   1109257.50  1167280.50  1287327.65  3569 1.00
## pred_cases[7]   4658535.50  4884010.75  5345265.90  3808 1.00
## pred_cases[8]  13506267.50 14136785.25 15457410.85  3885 1.00
## pred_cases[9]  17391463.50 18256891.00 19974186.65  4084 1.00
## pred_cases[10]  8919338.00  9344814.50 10223202.38  4006 1.00
## pred_cases[11]  2799225.50  2931302.75  3222769.55  3905 1.00
## pred_cases[12]   817804.50   860157.75   944246.45  3919 1.00
## pred_cases[13]   260633.00   274373.50   301324.12  4060 1.00
## pred_cases[14]    93169.00    97827.25   107125.90  3924 1.00
## pred_cases[15]    37192.00    39104.25    43136.28  4193 1.00
```

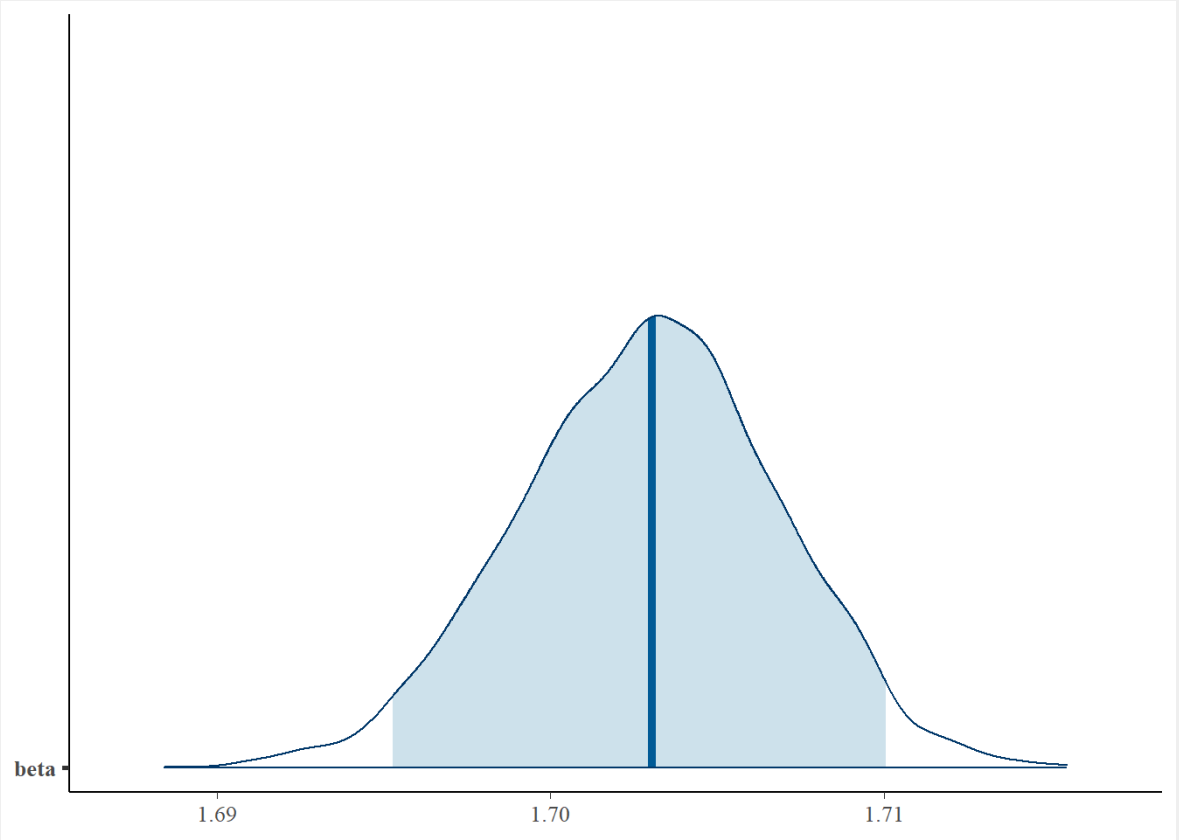
```
## pred_cases[16]    16451.00    17295.00    18915.18  3618 1.00
## pred_cases[17]     7992.00     8371.25     9186.15  3590 1.00
## pred_cases[18]     4163.00     4374.00     4798.05  3798 1.00
## pred_cases[19]     2327.00     2453.00     2708.05  3512 1.00
## pred_cases[20]     1379.00     1453.00     1599.00  3345 1.00
## pred_cases[21]      860.00      907.00     1004.00  3644 1.00
## pred_cases[22]      562.00      594.00      661.00  3769 1.00
## pred_cases[23]      380.00      404.00      452.03  3271 1.00
## pred_cases[24]      264.00      283.00      318.03  4063 1.00
## pred_cases[25]      265.50      283.00      318.03  3946 1.00
## R_0                  11.92       11.94       11.97   957 1.01
## lp__             -30773.52   -30773.11   -30772.86  1022 1.00
##
## Samples were drawn using NUTS(diag_e) at Sun Mar 24 13:55:01
2024.
## For each parameter, n_eff is a crude measure of effective
sample size,
## and Rhat is the potential scale reduction factor on split
chains (at
## convergence, Rhat=1).
# Extract and plot posterior distributions
posterior <- extract(stan_output)
# Adjust plot margins
par(mar = c(2, 2, 2, 2))

# Extract and summarize specific parameters
summary(posterior$beta)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.688   1.700   1.703   1.703   1.706   1.715
summary(posterior$R_0)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.82   11.90   11.92   11.92   11.94   12.01



#Plots
library(ggplot2)
beta <- extract(stan_output,'beta')[[1]]
qplot(beta)
```



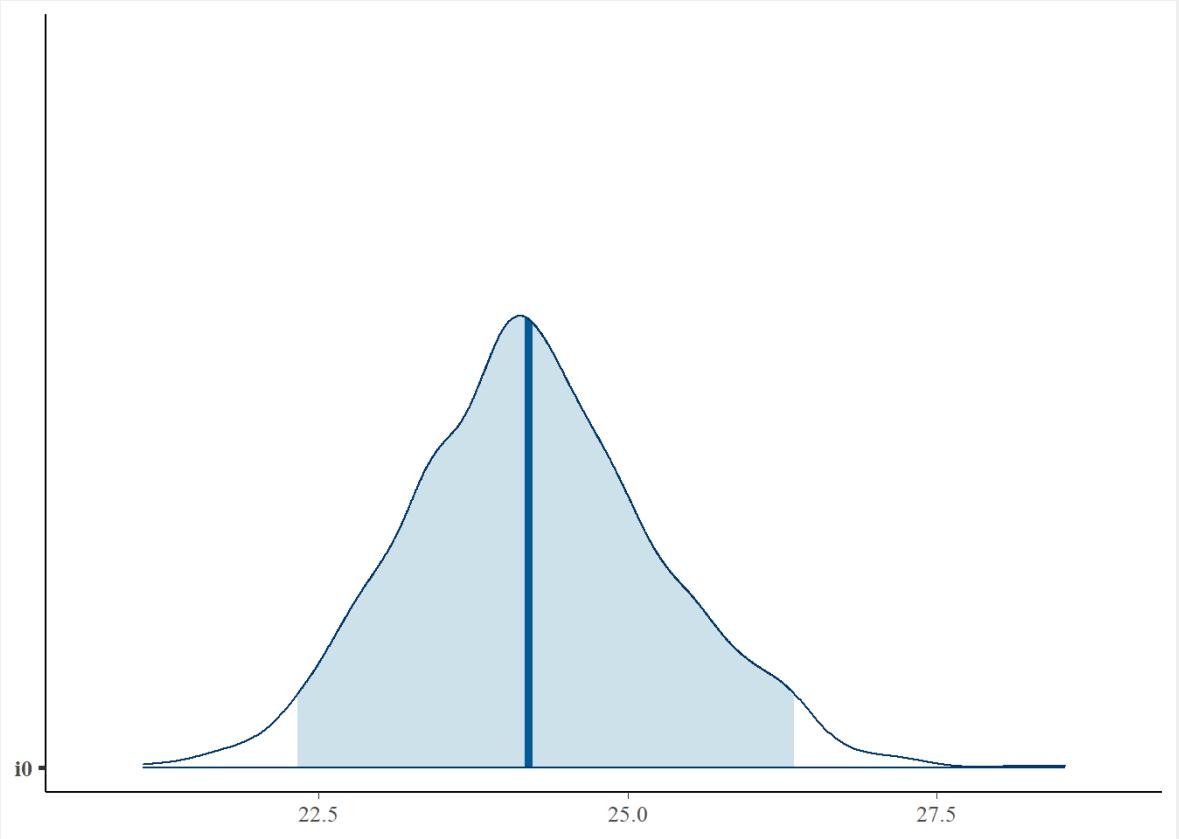
This code extracts the posterior samples of the parameter 'beta' from the Stan output.

Then qplot(beta) is used to create a plot of the posterior distribution of the parameter 'beta', which shows most values are centred around 1.74.

```
#Checking stan outputs
beta_pred<-rstan::extract(stan_output)$beta
i0_pred<-rstan::extract(stan_output)$i0

library(bayesplot)
mcmc_areas(stan_output,pars = "beta",prob = 0.95)    #95% credible
interval
```

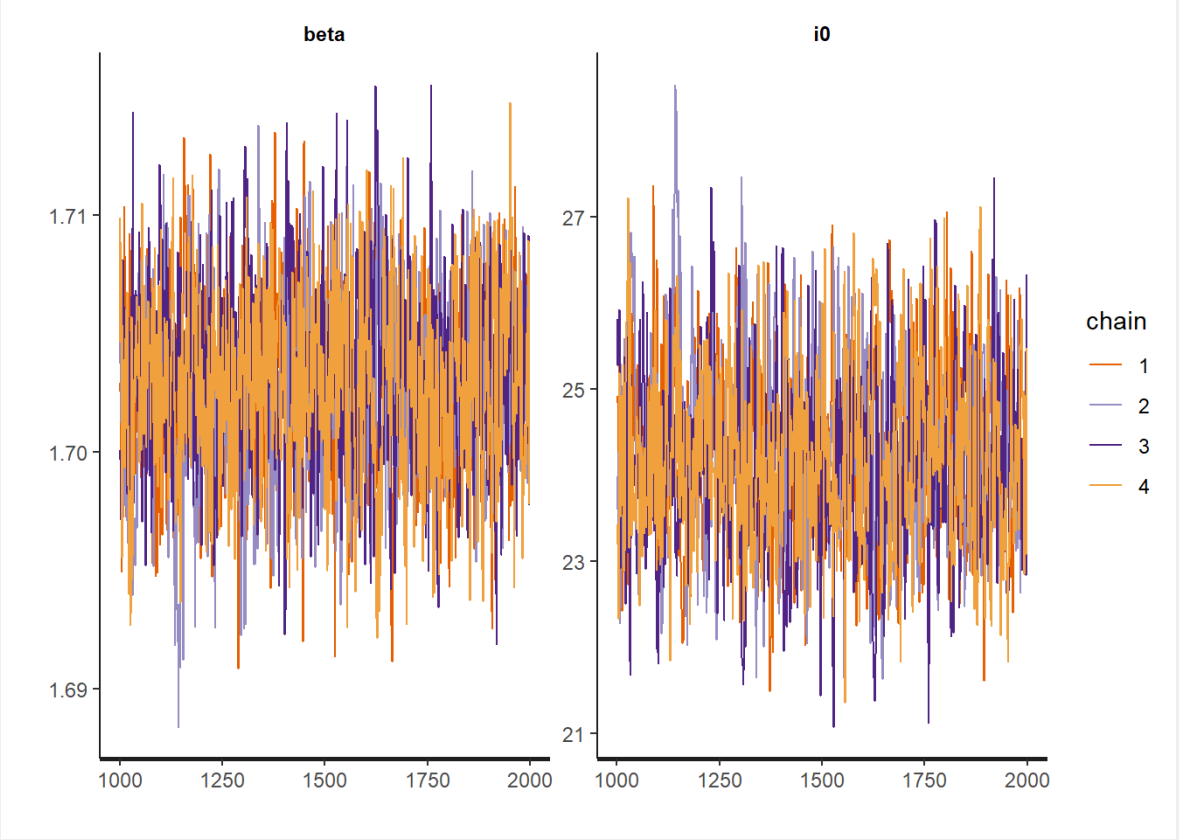



```
mcmc_areas(stan_output,pars = "i0",prob = 0.95)
```

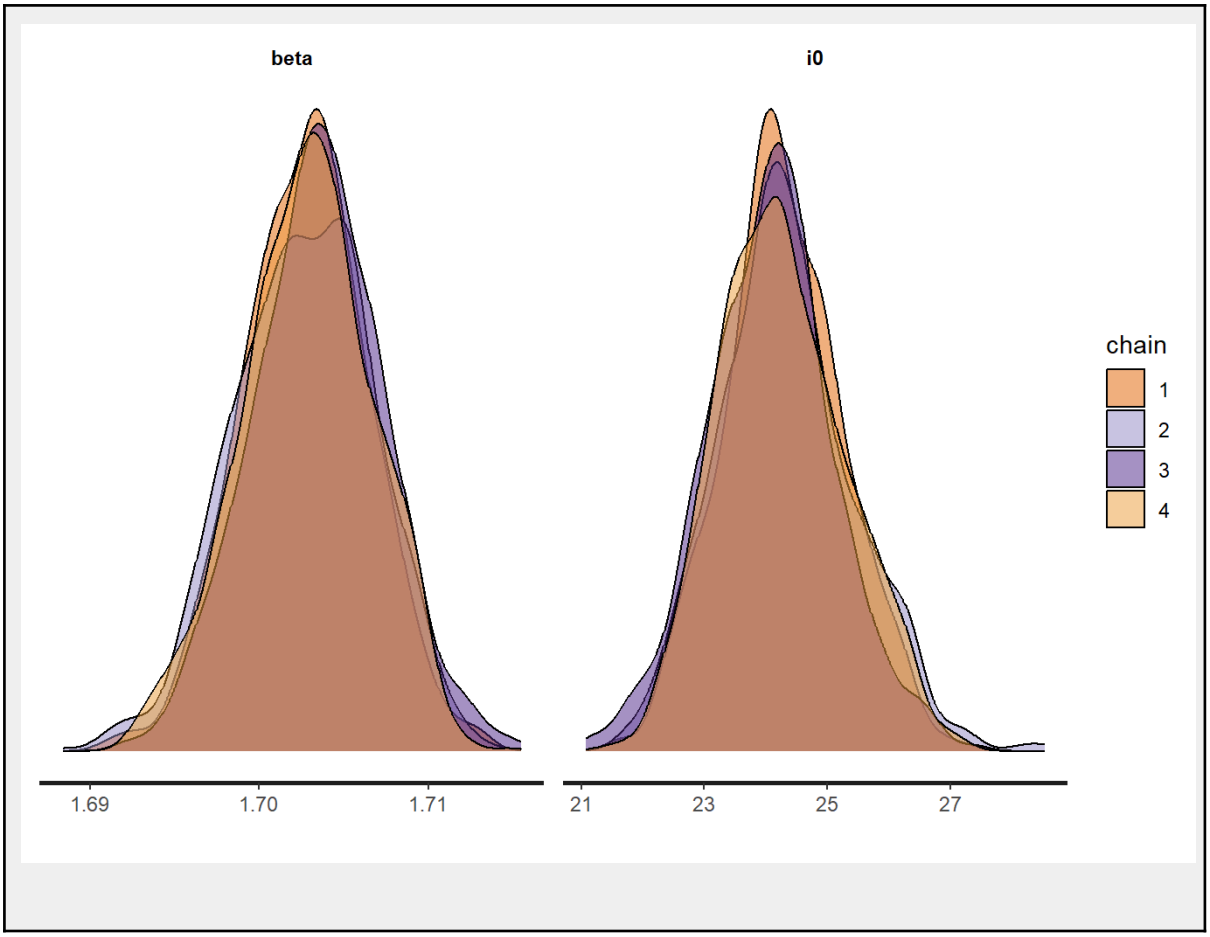



```
#diagnostic plots
```

```
traceplot(stan_output, pars = c( "beta","i0"))
```




```
stan_dens(stan_output, pars =  c("beta","i0"), separate_chains = TRUE)
```

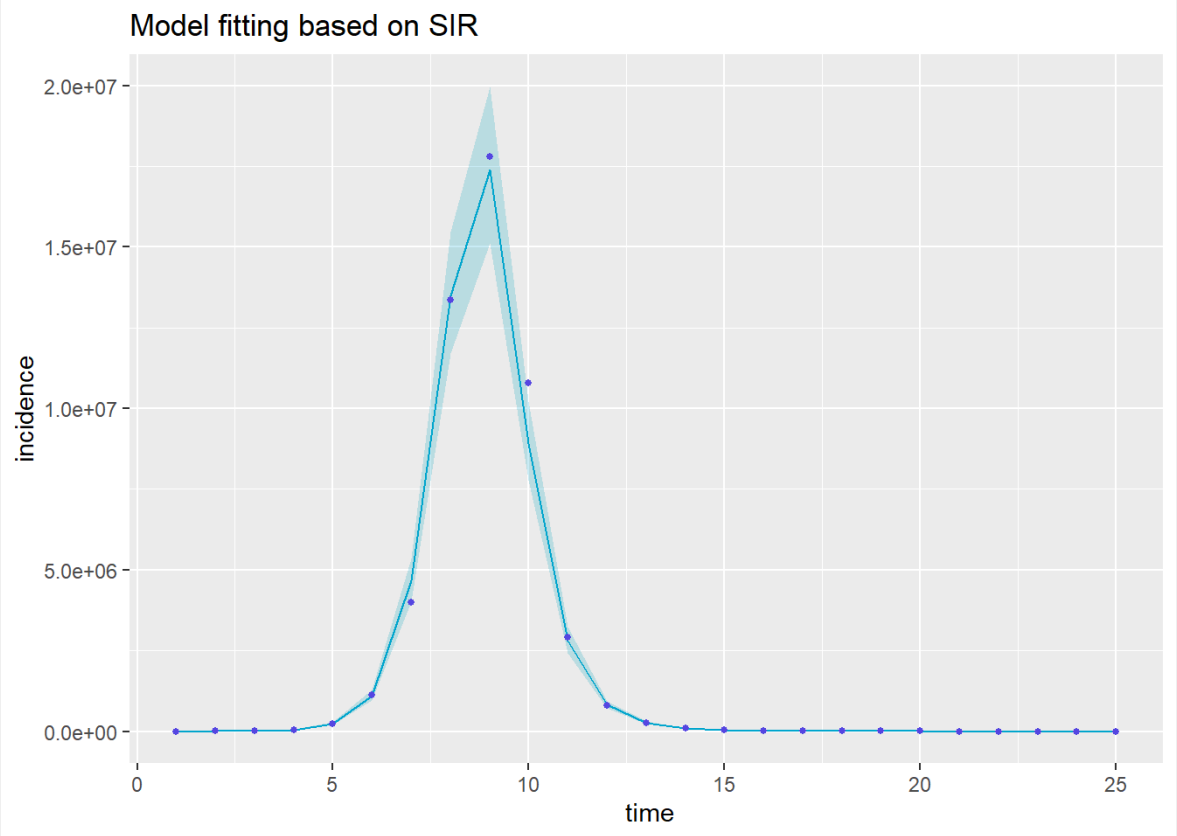


In this trace plot, each line represents an MCMC chain, with each point being a sampled value of beta and i0 respectively. Here we can see that all the chains are exploring the same space of the posterior distribution.

The marginal posterior densities plots confirm the Markov chains are in agreement with one another.

```
#Plot fitting results
pred_cases_SIR<-rstan::extract(stan_output)$pred_cases
#ggplot
t_start <- 0
t_stop <- 25
times <- seq(t_start, t_stop)[-1]

data<-data.frame(week=times,
                 pred_median=apply(pred_cases_SIR, 2, median),
```

```
                pred_low=apply(pred_cases_SIR,2, function(x)
quantile(x,probs=0.025)),
                pred_upp=apply(pred_cases_SIR,2,function(x)
quantile(x,probs=0.975)),
                obs_cases=cases)
ggplot(data, aes(x=week, y = pred_median)) +
  geom_line(size = 0.5,color= "#00A5CF") +
  geom_ribbon(aes(ymin=pred_low, ymax=pred_upp), alpha=0.2, colour
= NA,fill="#00A5CF")+
  geom_point( aes(week, obs_cases),color="#574AE2",size=1)+
  ylab("incidence") +
  xlab(" time")+
  ggtitle("Model fitting based on SIR")
```



Posterior predictive checks

The posterior predictive checks how well the model fits and generates simulations that is consistent with the observed data. It gives us a sense of how well the model fits the data and is also able to capture the uncertainty with a confidence belt around it in order to account for variation.

Refer to the table for estimations of the parameters derived:

Parameter	Estimate
beta	1.70
i0	24.20

R hat is close to 1 which implies that the 4 MC are in compliance, and the n effective value >100, indicating the parameter space has been explored.

Renewal Models

Renewal Models are another framework which requires us to make fewer assumptions than compartmental models, feels more intuitive due to its statistical backing and is easier to work with.
Renewal Models are nothing but the traditional renewal equation, where the values in the equation differ in their meaning due to its adaptation to epidemics study, leading to a change in interpretation of the variables. This adaptation was introduced by Kermack and McKendrick in 1927 in one of their works (A contribution to the mathematical theory of epidemics, Proc. Roy. Soc. London. Ser. A.)

The question that is being resolved by renewal models is that given the history of an infection, how many infections do we expect to occur today?

The attempt is to understand how infectiousness varies over time since the person becomes infected. Infectiousness is again dependent on various factors such as viral load within the individual, duration of infection, rate of shedding, contact rate between infected and susceptible individuals etc.

The equation for answering the same is :

$$E[I(t)] = R_t \sum_{\tau=1}^{\infty} W(\tau) I(t - \tau)$$

(code:https://rdrr.io/github/davidchampredon/gitest/src/R/RESuDe_FCT.R)

Here $I(t)$ represents the number of new infectious individuals, being dependent on time t. $W(\tau)$ is the probability to be infectious at time unit τ after getting the infection, which is depicted by the generation time distribution. The generation interval is the interval between the time when an individual is infected and also infects another individual.
R_t is the instantaneous reproduction number which is the average number of infections which a primary infection would generate throughout the course of their infection while the conditions in the population remain the same. Therefore, it is the transmission potential.

This equation implies that to know the average number of infections today, generation time distribution $W(\tau)$, where τ denoted the time since an individual is infected, and intensity of infection (R_t) would be necessary.

Generation Time Distribution:
It is a probabilistic distribution describing the time which elapses between the first person who is infected and the subsequent people becoming infected due to any form of contact with the first person.

Hence, we can estimate R_t by simplifying the above equation as follows:

$$R_t = \frac{E[I(t)]}{\sum_{\tau=1}^{\infty} W(\tau) I(t-\tau)}$$

We can also substitute generation time distribution with serial interval distribution which is much simpler to estimate as it is the time which elapses between individuals becoming symptomatic and the subsequent cases becoming symptomatic.

This substitution is helpful since, in the real world, the data that is available is that of symptomatic cases as against the history of infectious data.

The renewal model and the compartmental model's approach to tackle the problem in hand is quite varied. The prior focuses on cohorts of infectious individuals and how spread of infection takes place , while being dependent on time. On the other hand, the latter focuses on the count of individuals in different compartments at a given point of time.
In real practice, compartmental models outdo renewal models in terms of being a go to methods, and intuitively, one of the reasons for the same could be the necessity to predict the non-pharmaceutical or pharmaceutical interventions needed at different time points during the course of an infection. This can be forecasted better in compartmental models due to clear distinction between the populations presence in different states.
But to all the glory that compartmental models hold, the downfall is in the number of assumptions that one has to be mindful of, which is far from the reality. This complexity in the model is tackled by renewal models, which lead to a simpler to estimate effective reproduction number from incidence time series , leading to better epidemic forecasts.

Literature Review:

1) Are Epidemic Growth Rates More Informative than Reproduction Numbers?

2) Estimating Individual and Household Reproduction Numbers in an Emerging Epidemic