Bayesian workflow for disease transmission modeling

Julien Riou, MD PhD

Swiss Epidemiology Winter School
2022

Preface

- Objective: learn how to fit compartmental models in `Stan`
- Based on Grinsztajn et al., 2021 (link)
- Prerequisites:
  - basic programming with `R`
  - introduction to infectious disease modelling (day 1)
  - compartmental models (day 1)
- All material is available on github (link)

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Introduction

Models of disease transmission:

- Interpretability: phenomenological, mechanistic
- Scale: population-based, individual-based
- Framework: deterministic, stochastic
- Data-generating mechanisms: incubation, contagion, immunity, vaccination, mobility...

Introduction

Models of disease transmission:

- Interpretability: phenomenological, mechanistic
- Scale: population-based, individual-based
- Framework: deterministic, stochastic
- Data-generating mechanisms: incubation, contagion, immunity, vaccination, mobility...

Mechanistic + population-based + deterministic

→ compartmental, ODE-based models

Introduction

ODE-based compartmental model:

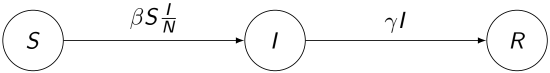
- Divide the population into homogeneous groups (compartments)
- Define the flows between compartments with ODEs
- Define initial conditions (at t_0)
- Solve for the time-dependent volume in each compartment

Introduction

ODE-based compartmental model:

- Divide the population into homogeneous groups (compartments)
- Define the flows between compartments with ODEs
- Define initial conditions (at t_0)
- Solve for the time-dependent volume in each compartment

The susceptible-infectious-recovered (SIR) model:

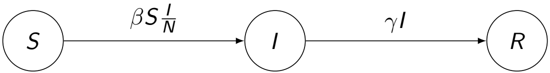


$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$

$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

The SIR model



$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$

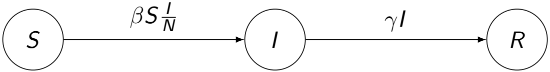
$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Where:

- $S(t)$ is the number of people susceptible to infection
- $I(t)$ is the number of people infected (i.e. the prevalence)
- $R(t)$ is the number of people recovered (lifelong immunity)
- N is the population size ($S(t) + I(t) + R(t) = N$ for any t)
- β is the infectious contact rate (per day per person)
- γ is the recovery rate (1/infectious period)

The SIR model

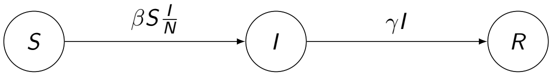


$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$
$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

Intuition behind the SIR model:

- $I(t)/N$ is the proportion of infected (and infectious)
- $\beta I(t)/N$ is the daily number of contacts with infectious people
- hence each day, $\beta SI(t)/N$ people become infected (the force of infection)

The SIR model



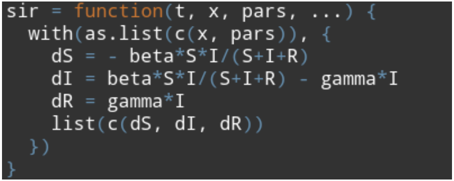
$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$
$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

Assumptions behind the SIR model:

- homogeneous mixing
- β and γ constant over time
- all infections are observed
- no incubation, exponentially-distributed recovery
- lifelong immunity
- stable population

Simulate a SIR in R

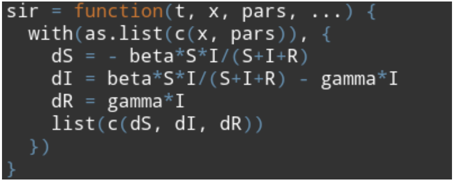
- load library `deSolve` (and `tidyverse`)
- set compartments and differential equations



```
sir = function(t, x, pars, ...) {
  with(as.list(c(x, pars)), {
    dS = - beta*S*I/(S+I+R)
    dI = beta*S*I/(S+I+R) - gamma*I
    dR = gamma*I
    list(c(dS, dI, dR))
  })
}
```

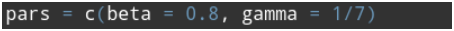
Simulate a SIR in R

- load library deSolve (and tidyverse)
- set compartments and differential equations



```
sir = function(t, x, pars, ...) {
  with(as.list(c(x, pars)), {
    dS = - beta*S*I/(S+I+R)
    dI = beta*S*I/(S+I+R) - gamma*I
    dR = gamma*I
    list(c(dS, dI, dR))
  })
}
```

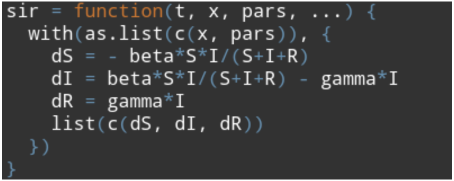
- set (fixed) values for parameters: $\beta = 0.8$; $\rho = 1/7$



```
pars = c(beta = 0.8, gamma = 1/7)
```

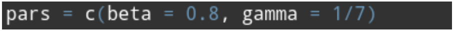
Simulate a SIR in R

- load library deSolve (and tidyverse)
- set compartments and differential equations



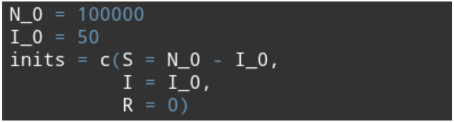
```
sir = function(t, x, pars, ...) {
  with(as.list(c(x, pars)), {
    dS = - beta*S*I/(S+I+R)
    dI = beta*S*I/(S+I+R) - gamma*I
    dR = gamma*I
    list(c(dS, dI, dR))
  })
}
```

- set (fixed) values for parameters: $\beta = 0.8$; $\rho = 1/7$



```
pars = c(beta = 0.8, gamma = 1/7)
```

- set (fixed) values for initial values

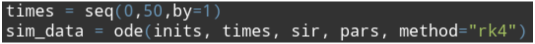


```
N_0 = 100000
I_0 = 50
inits = c(S = N_0 - I_0,
          I = I_0,
          R = 0)
```

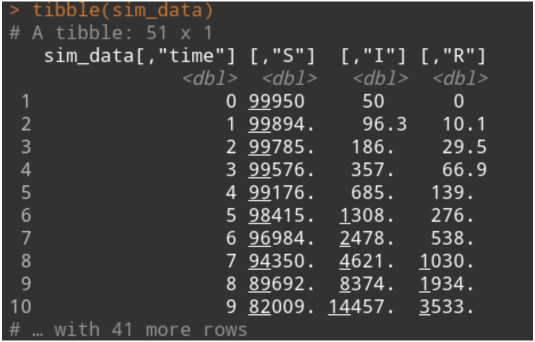
Simulate a SIR in R

- solve the ODE system numerically (Runge-Kutta 4th order) to obtain unique solutions for $S(t)$, $I(t)$ and $R(t)$

$$f(\beta, \gamma, S_0, I_0, R_0) = \{S(t), I(t), R(t)\}$$



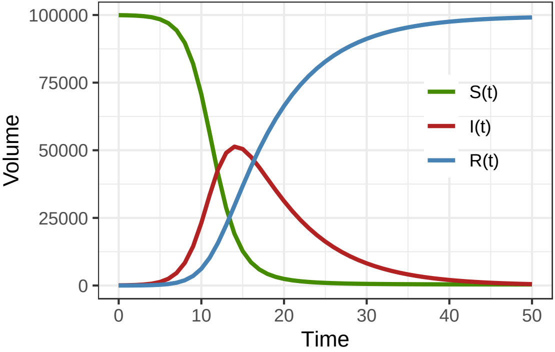
```r
times = seq(0,50,by=1)
sim_data = ode(inits, times, sir, pars, method="rk4")
```



```
> tibble(sim_data)
# A tibble: 51 x 1
   sim_data[,"time"] [,"S"]  [,"I"] [,"R"]
               <dbl>  <dbl>   <dbl>  <dbl>
 1                 0 99950      50      0
 2                 1 99894.     96.3   10.1
 3                 2 99785.    186.    29.5
 4                 3 99576.    357.    66.9
 5                 4 99176.    685.   139.
 6                 5 98415.   1308.   276.
 7                 6 96984.   2478.   538.
 8                 7 94350.   4621.  1030.
 9                 8 89692.   8374.  1934.
10                 9 82009.  14457.  3533.
# … with 41 more rows
```

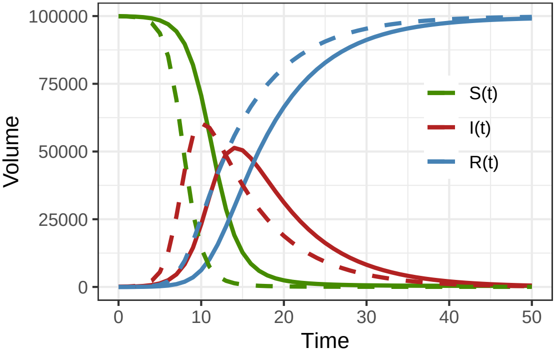
Simulate a SIR in R

with $\beta = 0.8$; $\rho = 1/7$; $S_0 = 100000 - 50$; $I_0 = 50$ and $R_0 = 0$



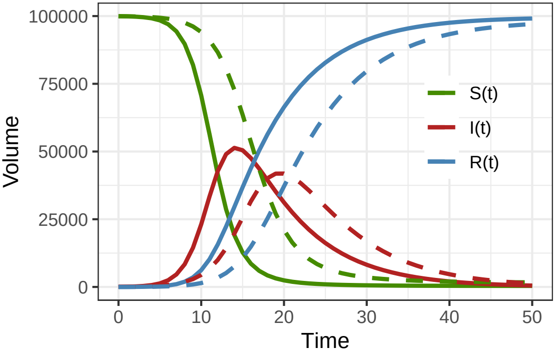
Simulate a SIR in R

with $\beta = 1.1$ instead of 0.8, we get



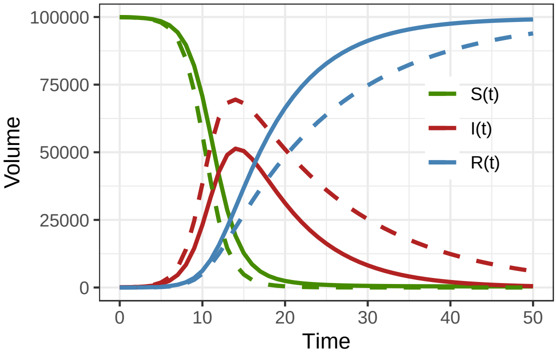
Simulate a SIR in R

with $\beta = 0.6$ instead of 0.8, we get



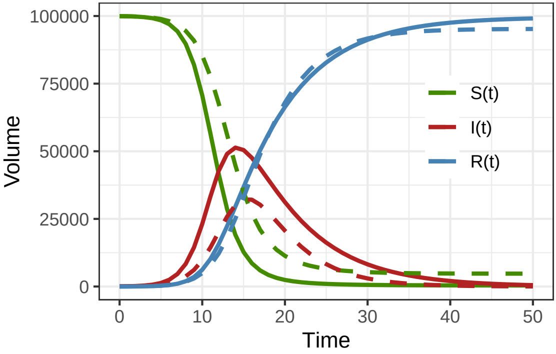
Simulate a SIR in R

with $\gamma = 1/14$ instead of $1/7$, we get



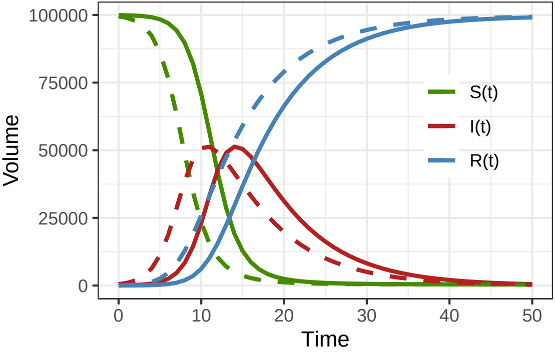
Simulate a SIR in R

with $\gamma = 1/4$ instead of $1/7$, we get



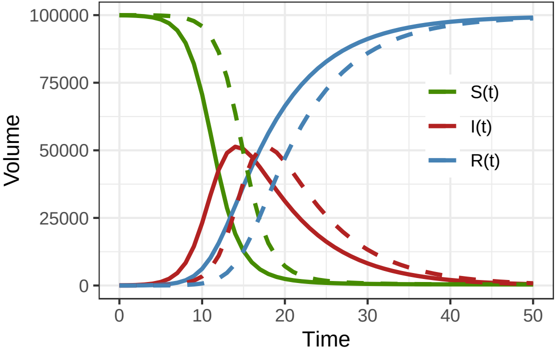
Simulate a SIR in R

with $I(0) = 500$ instead of 50, we get



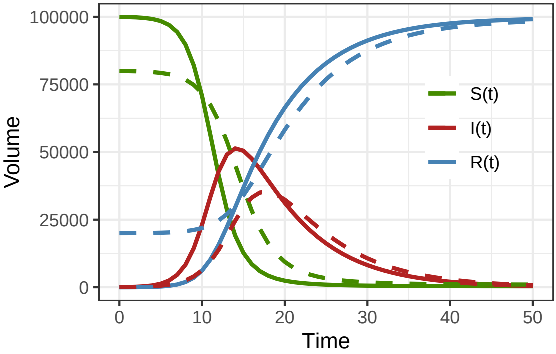
Simulate a SIR in R

with $I(0) = 5$ instead of 50, we get



Simulate a SIR in R

with $R(0) = 20,000$ instead of 0, we get



Applications

Compartmental models have many uses:

- get mechanistic insight about an epidemic (transmissibility levels, drivers of transmission)

$$\mathcal{R}_0 = \frac{\beta}{\gamma}$$

- formalize and put numerical values on general concepts (herd immunity threshold...)

$$V_c = \frac{1}{\mathcal{R}_0}$$

- produce (short-term) forecasts accounting for contagion and immunity

Applications

Compartmental models have many uses:

- get mechanistic insight about an epidemic (transmissibility levels, drivers of transmission)

$$\mathcal{R}_0 = \frac{\beta}{\gamma}$$

- formalize and put numerical values on general concepts (herd immunity threshold...)

$$V_c = \frac{1}{\mathcal{R}_0}$$

- produce (short-term) forecasts accounting for contagion and immunity

→ all these uses are based on numerical values for the parameters!

Applications

Enters Bayesian inference:

- infer parameter values by integrating data and domain knowledge
- more efficient for complex models (high dimensionality)
- rigorously quantify and propagate uncertainty in parameter estimates and forecast

Applications

Enters Bayesian inference:

- infer parameter values by integrating data and domain knowledge
- more efficient for complex models (high dimensionality)
- rigorously quantify and propagate uncertainty in parameter estimates and forecast

→ Markov Chain Monte Carlo (MCMC) methods and Stan

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Bayesian inference in theory

General principle of Bayesian inference[1,2]:

(1) specify a complete Bayesian model

- consider N data points $y = \{y_1, ..., y_N\}$ that are noisy measurements based on an unknown quantity θ (a parameter)

[1] Gelman et al. *Bayesian Data Analysis*

[2] Betancourt, *An introduction to Stan* (2020)

Bayesian inference in theory

General principle of Bayesian inference[1,2]:

(1) specify a complete Bayesian model

- consider N data points $y = \{y_1, ..., y_N\}$ that are noisy measurements based on an unknown quantity θ (a parameter)
- encode the data-generating processes in a likelihood-based observation model ("how do I get from θ to y?")

$$\Pr(y|\theta) = \prod_{n=1}^{N} \text{normal}(y_n|\theta, 1)$$

[1] Gelman et al. *Bayesian Data Analysis*

[2] Betancourt, *An introduction to Stan* (2020)

Bayesian inference in theory

General principle of Bayesian inference[1,2]:

(1) specify a complete Bayesian model

- consider N data points $y = \{y_1, ..., y_N\}$ that are noisy measurements based on an unknown quantity θ (a parameter)
- encode the data-generating processes in a likelihood-based observation model ("how do I get from θ to y?")

$$\mathrm{Pr}(y|\theta) = \prod_{n=1}^{N} \mathrm{normal}(y_n|\theta, 1)$$

- encode domain knowledge about θ in a prior distribution ("what do I know about θ?")

$$\mathrm{Pr}(\theta) = \mathrm{normal}(0, 1)$$

[1] Gelman et al. *Bayesian Data Analysis*

[2] Betancourt, *An introduction to Stan* (2020)

Bayesian inference in theory

(2) reverse the question:

"*given a model and data, what are plausible parameter values?*"

- the answer comes from Bayes' rule:

$$\Pr(\theta \mid y) \propto \Pr(\theta) \Pr(y \mid \theta)$$

- where $p(\theta \mid y)$ is the posterior distribution, i.e. the set of plausible parameter values given data and model

Bayesian inference in practice

In practice, we rarely have an analytic expression for $p(\theta \mid y)$ and rely on sampling algorithms to learn about the posterior distribution

- Monte Carlo methods

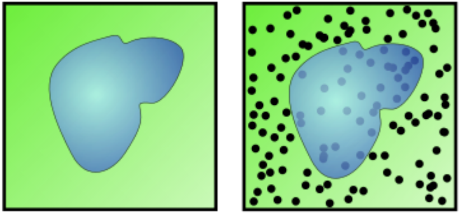


Figure: Monte Carlo approach to determine the area of a lake using random artillery fire.

Bayesian inference in practice

- Markov chain Monte Carlo (MCMC) samplers apply this principle to probability distributions

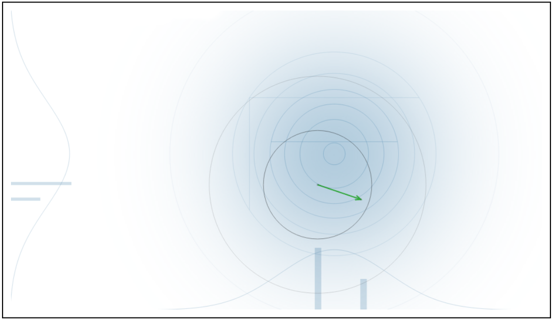


Figure: A Metropolis-Hastings sampling algorithm exploring a bivariate normal target distribution (`https://chi-feng.github.io/mcmc-demo/app.html`).

Sampling

Samples obtained through MCMC give an approximation of the posterior distribution of the parameters

- multiple, sufficiently long chains (asymptotic property)
- converging to a similar answer from different starting points

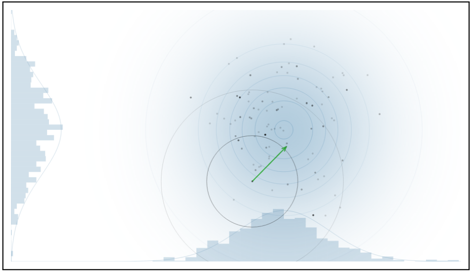


Figure: A small set of samples approximating a bivariate normal target distribution.

Sampling

Samples obtained through MCMC give an approximation of the posterior distribution of the parameters

- multiple, sufficiently long chains (asymptotic property)
- converging to the same answer from different starting points

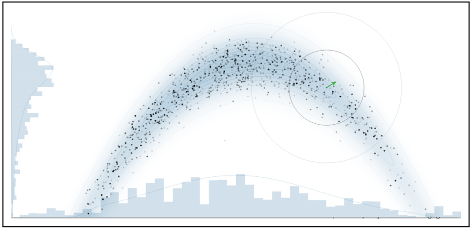


Figure: A large set of samples that approximate a "banana-shaped" target distribution.

Sampling

Samples obtained through MCMC give an approximation of the posterior distribution of the parameters

- multiple, sufficiently long chains (asymptotic property)
- converging to the same answer from different starting points

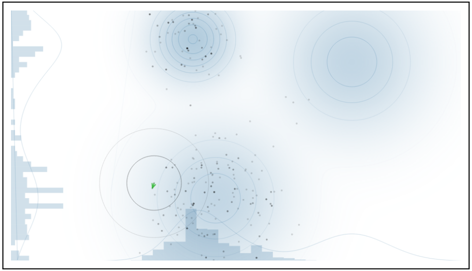


Figure: A set of samples that fail to approximate a multi-modal target distribution.

Sampling

Samples obtained through MCMC give an approximation of the posterior distribution of the parameters

- multiple, sufficiently long chains (asymptotic property)
- converging to the same answer from different starting points

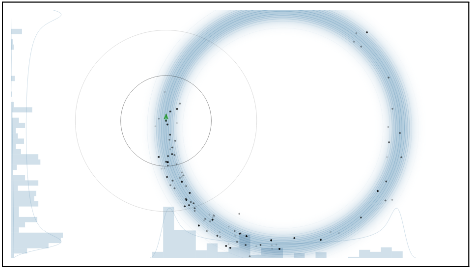


Figure: A set of samples that fail to approximate a “donut-shaped” target distribution.

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Stan

Stan is a probabilistic programming framework for Bayesian inference

- it is designed to let the user focus on modeling while inference happens under the hood
- object-oriented language (based on `C++`) that supports many operations, probability densities and ODE solvers
- efficient MCMC algorithm (Hamiltonian Monte Carlo)
- diagnostic tools to evaluate the inference
- interfaces in `R` (package `rstan`), `python`, `julia`...

Stan

Stan primarily uses NUTS (No U-Turn Sampler), a very efficient Hamiltonian Monte Carlo sampling algorithm

- based on Hamiltonian dynamics (classical mechanics)
- follows the "curvature" of the target distribution

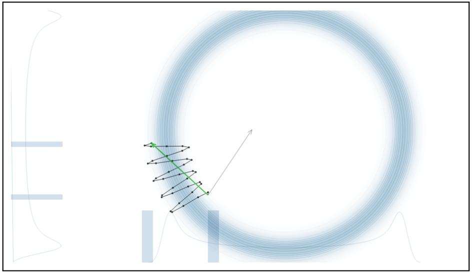


Figure: An HMC sampling algorithm exploring a "donut-shaped" target distribution (`https://chi-feng.github.io/mcmc-demo/app.html`).

Stan

Stan primarily uses NUTS (No U-Turn Sampler), a very efficient Hamiltonian Monte Carlo sampling algorithm

- based on Hamiltonian dynamics (classical mechanics)
- follows the "curvature" of the target distribution

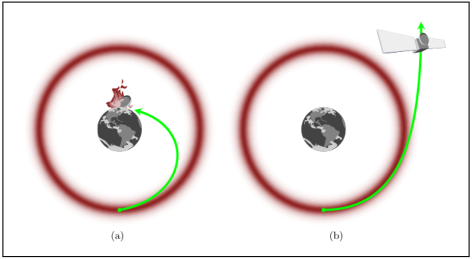


Figure: Illustration of how an HMC sampling algorithm works using a satellite orbiting earth with too few (a) or too much momentum (b) [source: Betancourt, *A Conceptual Introduction to Hamiltonian Monte Carlo*, 2017].

Stan

Stan primarily uses NUTS (No U-Turn Sampler), a very efficient Hamiltonian Monte Carlo sampling algorithm

- based on Hamiltonian dynamics (physics of movement)
- follows the gradient of the target distribution

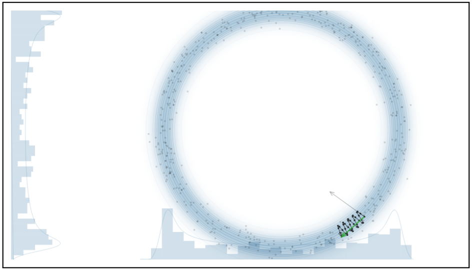
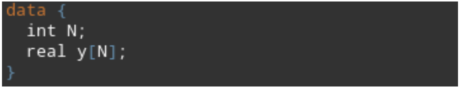


Figure: A set of samples that approximate a “donut-shaped” target distribution.

Stan example

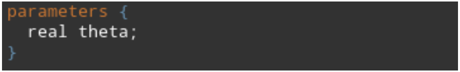
Programming in Stan is structured in blocks:

- the `data` block defines data variables



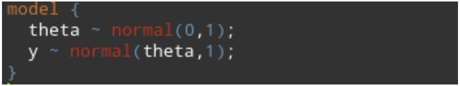
```
data {
  int N;
  real y[N];
}
```

- the `parameters` block defines parameters



```
parameters {
  real theta;
}
```

- the `model` block defines the target log probability density function



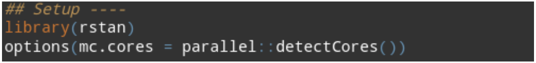
```
model {
  theta ~ normal(0,1);
  y ~ normal(theta,1);
}
```

- save in `model_linear.stan`

Stan example

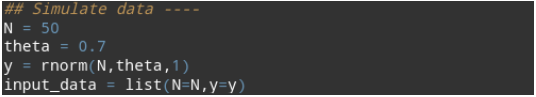
We then explore the posterior distribution of θ with Stan:

- load `rstan` package



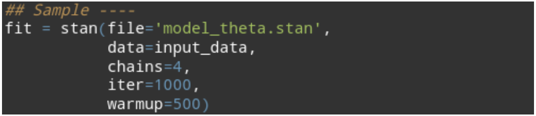
```
## Setup ----
library(rstan)
options(mc.cores = parallel::detectCores())
```

- simulate $N = 50$ data points with $\theta = 0.7$ (the “true” value)



```
## Simulate data ----
N = 50
theta = 0.7
y = rnorm(N,theta,1)
input_data = list(N=N,y=y)
```

- run MCMC sampling!



```
## Sample ----
fit = stan(file='model_theta.stan',
           data=input_data,
           chains=4,
           iter=1000,
           warmup=500)
```

Stan example

Visual assessment of convergence:

- "Trace plots" show the evolution of each chain throughout sampling [`stan_trace()`]

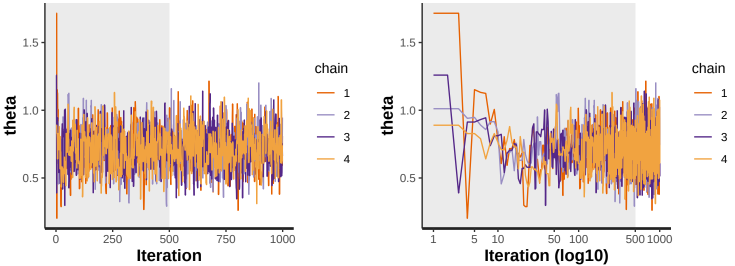


Figure: "Trace plot" of 4 chains of 1,000 iterations exploring the posterior distribution of θ. The warm-up period (in grey) is discarded. Each chain starts at a different initial value (as visible on the right due to the log-transformed x-axis) but all chains end up mixing relatively well ("hairy caterpillar").

Stan example

- Take a look at the post-warm-up samples [`stan_dens()`]

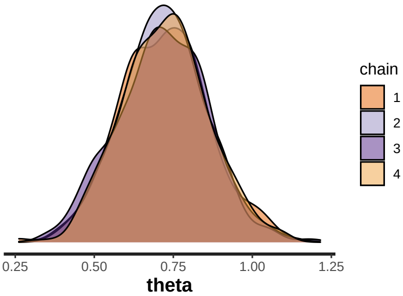


Figure: Distribution of the ? post-warm-up samples for θ for each chain. Each chain should lead to a similar distribution (mean and variance). This similarity is measured by the Gelman-Rubin $\hat{R}$ (or R-hat) convergence diagnostic.

Stan example

Visual assessment of autocorrelation [`stan_ac()`]:

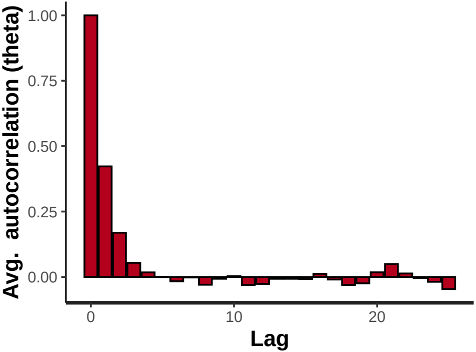
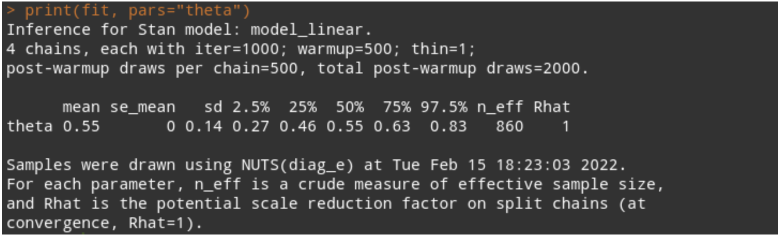


Figure: Correlation between successive samples. Starts at 1 for a lag of 0, and should drop quickly to around 0 as the lag increases. This indicates that the sampler explores the parameter space efficiently, and that each new iteration adds more information. On the contrary, if autocorrelation remains high beyond short lags, this means that the sampler is not progressing as much. This is used to compute the effective sample size.

Stan example

Numerical assessment of convergence and autocorrelation:



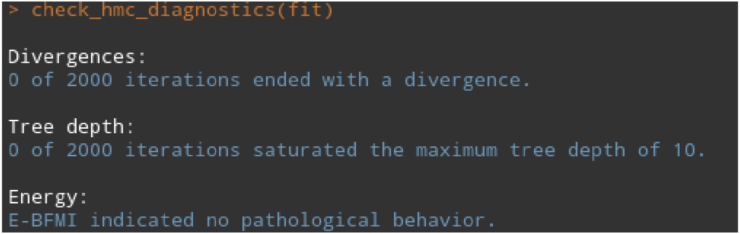
```
> print(fit, pars="theta")
Inference for Stan model: model_linear.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

      mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
theta 0.55       0 0.14 0.27 0.46 0.55 0.63  0.83   860    1

Samples were drawn using NUTS(diag_e) at Tue Feb 15 18:23:03 2022.
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).
```

- The Gelman-Rubin $\hat{R}$ (or R-hat) convergence diagnostic is a measure of convergence, it should be close to 1 (< 1.05)

- The effective sample size is the number of samples corrected for autocorrelation, it determines the precision of the posterior estimates (at least in the 100s, more if you are interested in the tails of the distribution)

Stan example

Advanced diagnostics of the sampling process:

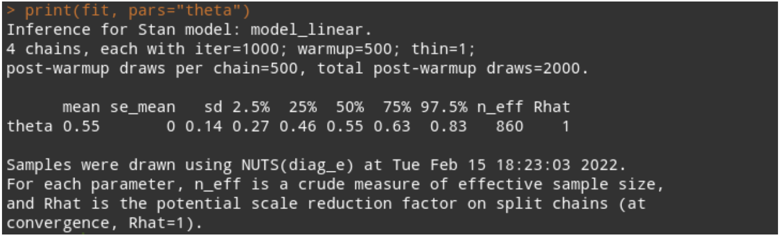


```
> check_hmc_diagnostics(fit)

Divergences:
0 of 2000 iterations ended with a divergence.

Tree depth:
0 of 2000 iterations saturated the maximum tree depth of 10.

Energy:
E-BFMI indicated no pathological behavior.
```

- Divergences may lead to biases in the inference and should never be ignored
- Tree depth and energy issues are more of an efficiency concern

Stan example

If we find no reason to doubt the inference, we can interpret the samples as an approximation of the posterior distribution of θ:



```
> print(fit, pars="theta")
Inference for Stan model: model_linear.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

      mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
theta 0.55       0 0.14 0.27 0.46 0.55 0.63  0.83   860    1

Samples were drawn using NUTS(diag_e) at Tue Feb 15 18:23:03 2022.
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).
```

- point estimate of θ from the median or mean (0.55)
- 95% credible interval from the 2.5% and 97.5% percentile (0.27-0.83)
- other intervals, transformations, aggregations, predictions...

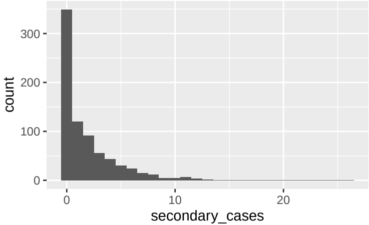
Exercises

Exercise 1

- Go through the code for the previous example (link)

Exercise 2 (optional)

- Estimate $\mathcal{R}_0$ and superspreading for SARS-CoV 2 from a distribution of secondary cases (link)
- Load data `secondary_cases.rds`



- Write, fit and diagnose a custom `Stan` model (hint: `neg_binomial_2()`)

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

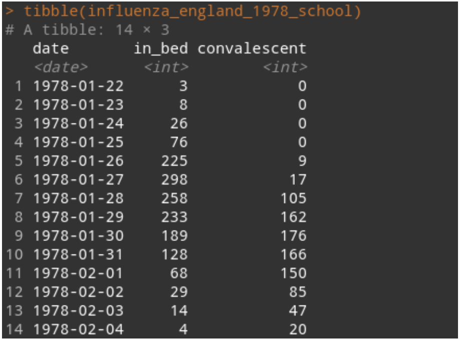
Extending from the simple SIR model (short version)

Conclusions

Example dataset

Outbreak of influenza A (H1N1) at a British boarding school in 1978 (available in R package outbreaks)

- 763 students, 512 had symptoms
- daily number of students in bed over 14 days (prevalence)

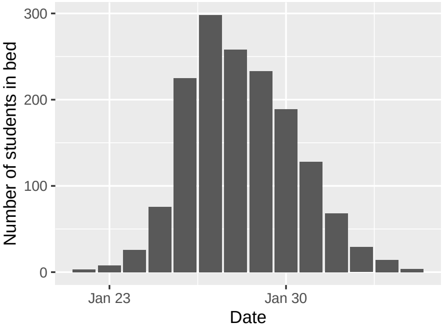


```
> tibble(influenza_england_1978_school)
# A tibble: 14 × 3
   date       in_bed convalescent
   <date>      <int>        <int>
 1 1978-01-22      3            0
 2 1978-01-23      8            0
 3 1978-01-24     26            0
 4 1978-01-25     76            0
 5 1978-01-26    225            9
 6 1978-01-27    298           17
 7 1978-01-28    258          105
 8 1978-01-29    233          162
 9 1978-01-30    189          176
10 1978-01-31    128          166
11 1978-02-01     68          150
12 1978-02-02     29           85
13 1978-02-03     14           47
14 1978-02-04      4           20
```

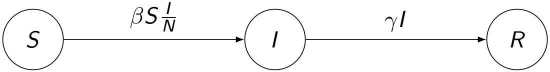
Example dataset

Outbreak of influenza A (H1N1) at a British boarding school in 1978 (available in `R` package `outbreaks`)

- 763 students, 512 had symptoms
- daily number of students in bed over 14 days (prevalence)



The SIR model



$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$

$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Connect the model to the situation using domain knowledge and assumptions:

- $I(t)$ is the number of people infected at time t → in bed
- N is the population size → fixed to 763
- $I(0)$ is the initial number of infected → fixed to 1
- $R(0)$ is the initial number of recovered/resistant → fixed to 0
- β is the infectious contact rate → parameter
- γ is the recovery rate → parameter

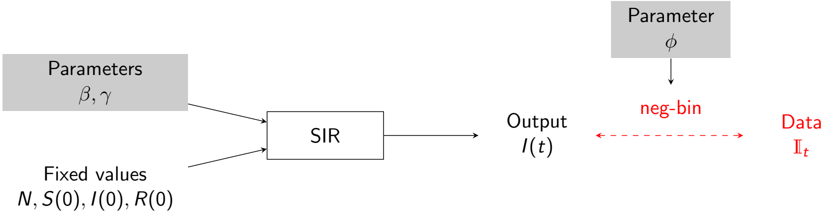
Specify a complete Bayesian model

(1) Encode the data-generating processes in an observation model:

- upon solving the ODE system with given parameter values (and fixed values), the SIR model will produce output $I(t)$

- the number of students in bed $\mathbb{I}_t$ is a noisy measurement of the model output $I(t)$, likely with overdispersion

$$\mathbb{I} \sim \text{negative-binomial}\left(I(t), \phi\right)$$



Specify a complete Bayesian model

(2) Encode domain knowledge in prior distributions:

- The infectious contact rate is strictly positive, likely <4

$$\beta \sim \text{exponential}(1)$$

- The recovery time is about 2 days (more convenient than the recovery rate)

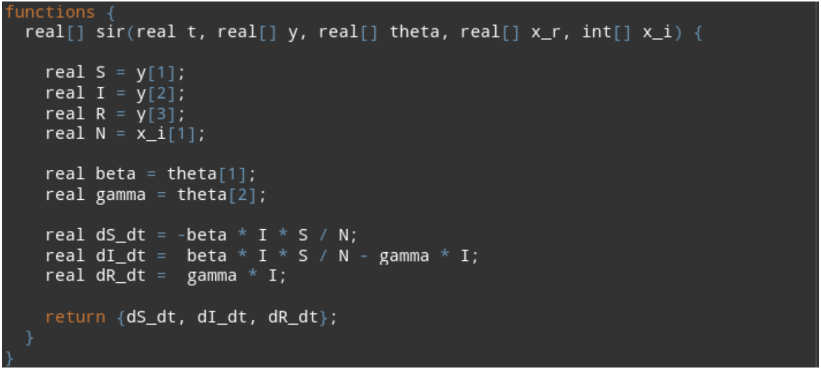
$$\gamma^{-1} \sim \text{normal}(2, 0.5)$$

- Overdispersion should be low by default

$$\phi^{-1} \sim \text{exponential}(5)$$

Build the model

We define the ODE system in the `function` block



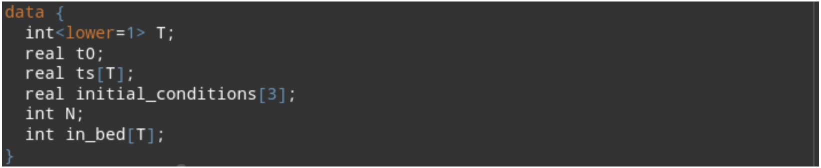
```
functions {
  real[] sir(real t, real[] y, real[] theta, real[] x_r, int[] x_i) {

    real S = y[1];
    real I = y[2];
    real R = y[3];
    real N = x_i[1];

    real beta = theta[1];
    real gamma = theta[2];

    real dS_dt = -beta * I * S / N;
    real dI_dt =  beta * I * S / N - gamma * I;
    real dR_dt =  gamma * I;

    return {dS_dt, dI_dt, dR_dt};
  }
}
```

Be careful of the signature and formats (some gymnastic needed!):

- all parameters go in `theta`
- all fixed values go in `x_r` (reals) or `x_i` (integers)

Build the model

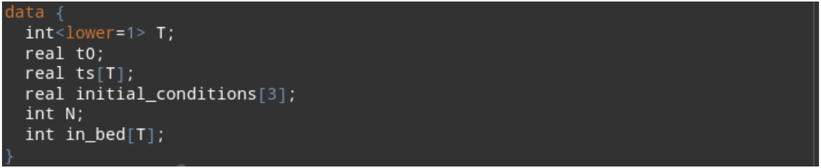
We declare the data in the `data` block



```
data {
  int<lower=1> T;
  real t0;
  real ts[T];
  real initial_conditions[3];
  int N;
  int in_bed[T];
}
```

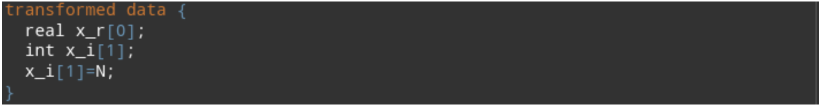
Build the model

We declare the data in the `data` block



```
data {
  int<lower=1> T;
  real t0;
  real ts[T];
  real initial_conditions[3];
  int N;
  int in_bed[T];
}
```

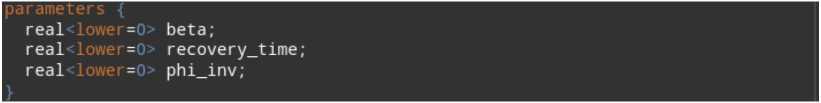
and define additional data variables (in particular `x_r` and `x_i`) in the `transformed data` block



```
transformed data {
  real x_r[0];
  int x_i[1];
  x_i[1]=N;
}
```

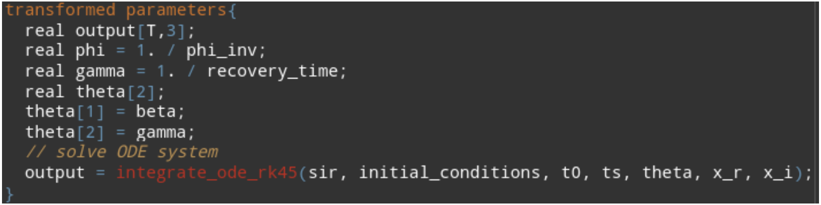
Build the model

Parameters are declared in the `parameters` block



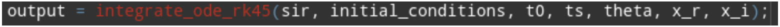
```
parameters {
  real<lower=0> beta;
  real<lower=0> recovery_time;
  real<lower=0> phi_inv;
}
```

and we define additional parameters and solve the ODE system in `transformed parameters`



```
transformed parameters{
  real output[T,3];
  real phi = 1. / phi_inv;
  real gamma = 1. / recovery_time;
  real theta[2];
  theta[1] = beta;
  theta[2] = gamma;
  // solve ODE system
  output = integrate_ode_rk45(sir, initial_conditions, t0, ts, theta, x_r, x_i);
}
```

Build the model



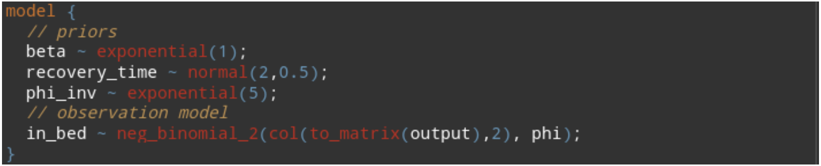
```
output = integrate_ode_rk45(sir, initial_conditions, t0, ts, theta, x_r, x_i);
```

Be careful about the signatures and formats:

- output is an array of size T×3 (number of evaluation points and number of compartments)
- sir is the name of the function defined in the function block
- initial conditions is an array of size 3
- ts is an array of size T (evaluation points)
- theta stores the parameters
- x_r and x_i store the fixed values

Build the model

In the `model` block, we put the priors and the observation model



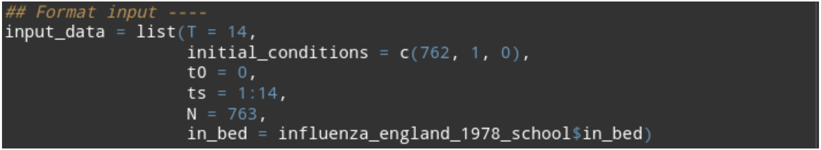
```
model {
  // priors
  beta ~ exponential(1);
  recovery_time ~ normal(2,0.5);
  phi_inv ~ exponential(5);
  // observation model
  in_bed ~ neg_binomial_2(col(to_matrix(output),2), phi);
}
```

⚠ It's important that the chosen distributions correspond with the boundaries set in the `parameters` block (`<lower=0>`)

⚠ `col(to_to_matrix(y))` extracts the 2nd column of `y`

Sampling

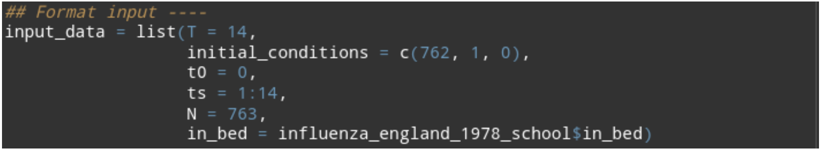
As before, we interact with Stan from R with the package rstan:



```
## Format input ----
input_data = list(T = 14,
                  initial_conditions = c(762, 1, 0),
                  t0 = 0,
                  ts = 1:14,
                  N = 763,
                  in_bed = influenza_england_1978_school$in_bed)
```

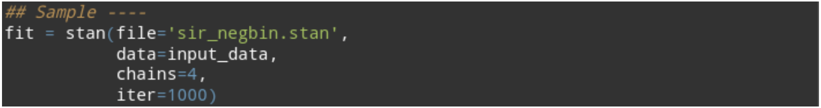
Sampling

As before, we interact with Stan from R with the package rstan:



```
## Format input ----
input_data = list(T = 14,
                  initial_conditions = c(762, 1, 0),
                  t0 = 0,
                  ts = 1:14,
                  N = 763,
                  in_bed = influenza_england_1978_school$in_bed)
```

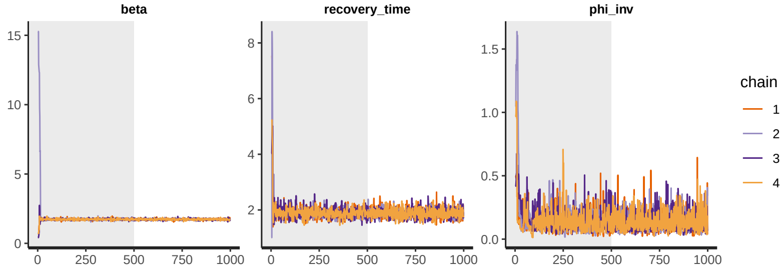
and we run MCMC sampling!



```
## Sample ----
fit = stan(file='sir_negbin.stan',
           data=input_data,
           chains=4,
           iter=1000)
```

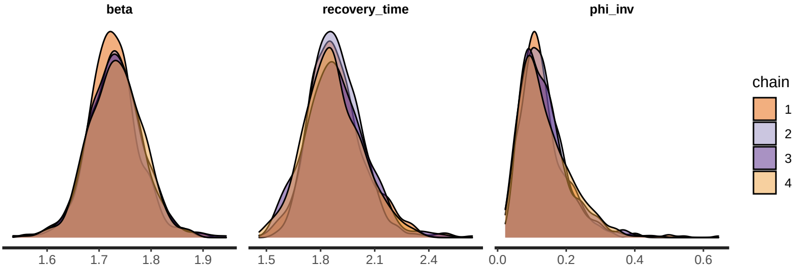
Checking the inference

Visual assessment of convergence:



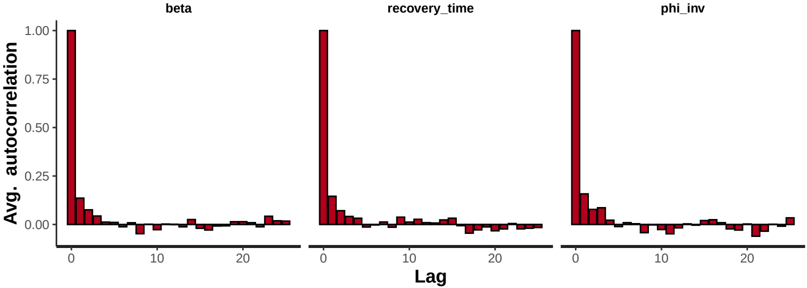
Checking the inference

Visual assessment of convergence:



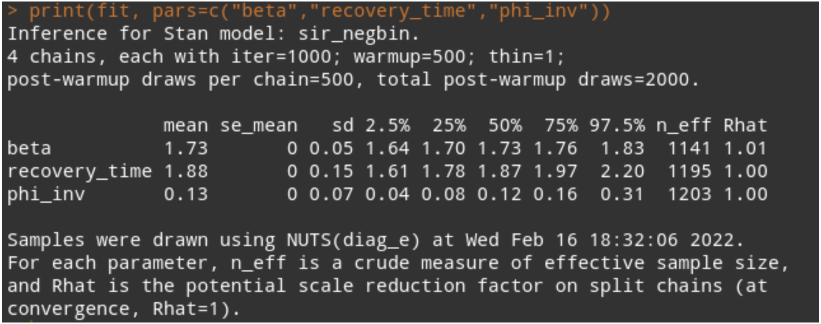
Checking the inference

Visual assessment of autocorrelation:



Checking the inference

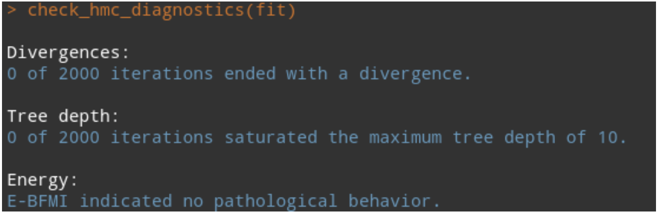
Numerical assessment of convergence and autocorrelation:



```
> print(fit, pars=c("beta","recovery_time","phi_inv"))
Inference for Stan model: sir_negbin.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

              mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
beta          1.73       0 0.05 1.64 1.70 1.73 1.76  1.83  1141 1.01
recovery_time 1.88       0 0.15 1.61 1.78 1.87 1.97  2.20  1195 1.00
phi_inv       0.13       0 0.07 0.04 0.08 0.12 0.16  0.31  1203 1.00

Samples were drawn using NUTS(diag_e) at Wed Feb 16 18:32:06 2022.
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).
```

Checking the inference

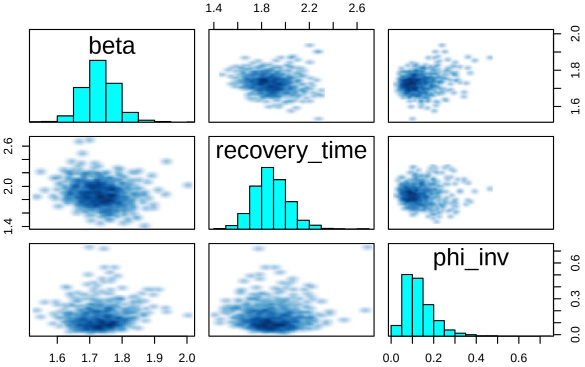
Check advanced diagnostics:



```
> check_hmc_diagnostics(fit)

Divergences:
0 of 2000 iterations ended with a divergence.

Tree depth:
0 of 2000 iterations saturated the maximum tree depth of 10.

Energy:
E-BFMI indicated no pathological behavior.
```

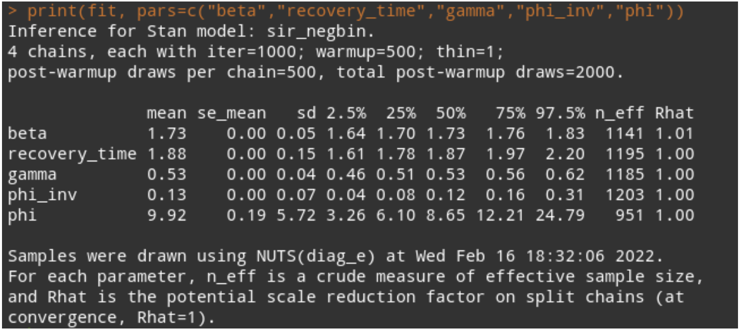
Checking the inference

We can also visualize the samples with a pairs plot:



Results

The inference seems good, we can interpret the samples as posterior distributions or estimates of the parameters:



```
> print(fit, pars=c("beta","recovery_time","gamma","phi_inv","phi"))
Inference for Stan model: sir_negbin.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

              mean se_mean   sd 2.5%  25%  50%   75% 97.5% n_eff Rhat
beta          1.73    0.00 0.05 1.64 1.70 1.73  1.76  1.83  1141 1.01
recovery_time 1.88    0.00 0.15 1.61 1.78 1.87  1.97  2.20  1195 1.00
gamma         0.53    0.00 0.04 0.46 0.51 0.53  0.56  0.62  1185 1.00
phi_inv       0.13    0.00 0.07 0.04 0.08 0.12  0.16  0.31  1203 1.00
phi           9.92    0.19 5.72 3.26 6.10 8.65 12.21 24.79   951 1.00

Samples were drawn using NUTS(diag_e) at Wed Feb 16 18:32:06 2022.
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).
```

- for this flu epidemic (and with our assumptions), we estimate the infectious contact rate to 1.73 (95% CrI: 1.64 to 1.83)
- the recovery time appears to be on the short side of what we expected, 1.88 days (95% CrI: 1.61 to 2.20)

Limits

This is not entirely satisfying:

1. parameter estimates are not everything that we want to know (we want to know the $\mathcal{R}_0$!)
2. we don't know how well the model fits to the data
3. the prior distributions we chose seem somewhat arbitrary
4. we don't really know if the model itself is adequate

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Beyond inference: generated quantities

Getting more from the model:

- transformed outputs
- posterior predictive checks
- prior predictive checks

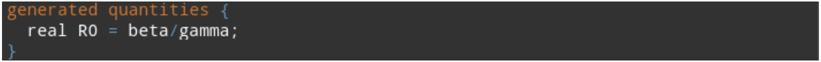
Transformed outputs

All model outputs can be transformed in any way necessary:

- apply all kinds of transformations and post-processing
- combine multiple parameters
- aggregate in various ways
- and all of that while preserving the propagation of uncertainty without worrying about variances and covariances!

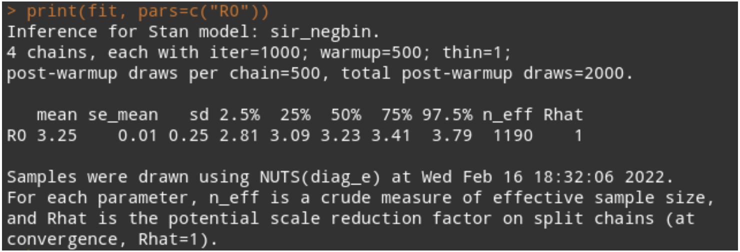
Transformed outputs

For that we use the `generated quantities` block:



```
generated quantities {
  real R0 = beta/gamma;
}
```

and the result is directly available using `print()`:

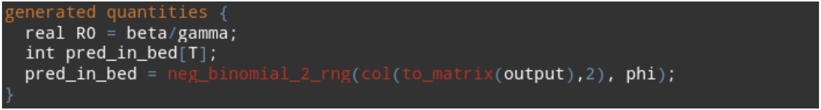


```
> print(fit, pars=c("R0"))
Inference for Stan model: sir_negbin.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

   mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
R0 3.25    0.01 0.25 2.81 3.09 3.23 3.41  3.79  1190    1

Samples were drawn using NUTS(diag_e) at Wed Feb 16 18:32:06 2022.
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).
```

- for this flu epidemic (and with our assumptions), we estimate $\mathcal{R}_0$ to 3.25 (95% CrI: 2.81 to 3.79)

Posterior predictive check

We can use the same block to produce model predictions that will mirror the data-generating mechanisms:



```
generated quantities {
  real R0 = beta/gamma;
  int pred_in_bed[T];
  pred_in_bed = neg_binomial_2_rng(col(to_matrix(output),2), phi);
}
```

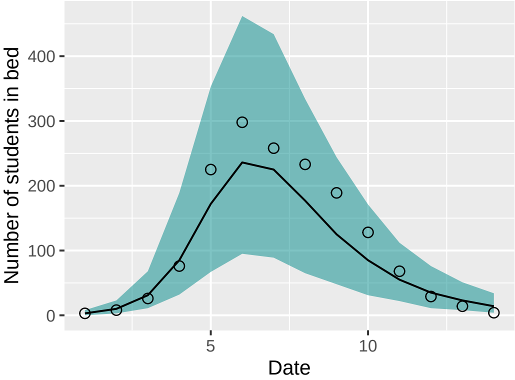
These predictions will incorporate all the sources of uncertainty within the model[3]:

- parameter uncertainty on β, γ and ϕ
- stochastic uncertainty from the negative binomial noise

[3] Zelner et al., *Accounting for uncertainty during a pandemic* (2021)

Posterior predictive check

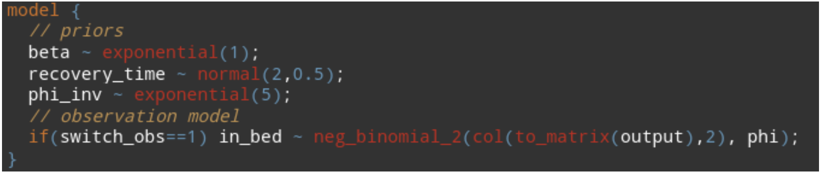
Comparing model predictions to data is called a posterior predictive check:



- the line and ribbon show the model-predicted number of students in bed (median and 95% *prediction interval*)
- gives an idea of how well the model fits

Prior predictive check

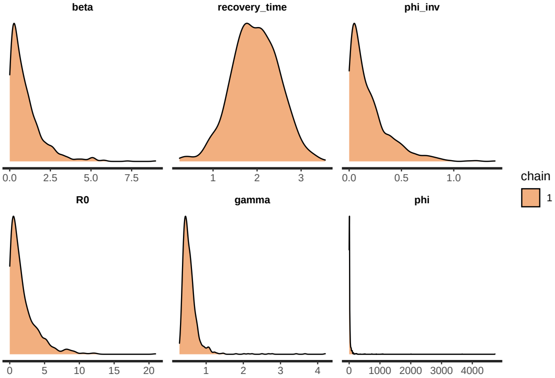
We can also compute model predictions from the unfitted model by “switching-off” the observation model:



```
model {
  // priors
  beta ~ exponential(1);
  recovery_time ~ normal(2,0.5);
  phi_inv ~ exponential(5);
  // observation model
  if(switch_obs==1) in_bed ~ neg_binomial_2(col(to_matrix(output),2), phi);
}
```

Prior predictive check

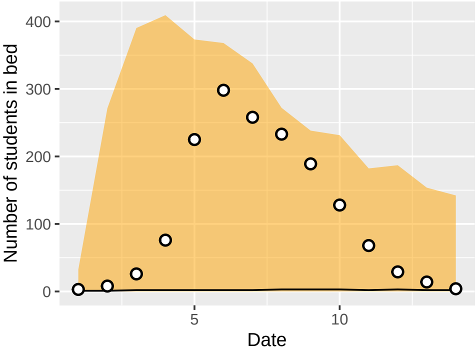
Without an observation model, parameter values will only reflect the chosen prior distributions:



NB: no need for long or multiple chains here

Prior predictive check

The model predictions now reflect the implications of the choices of prior distributions:

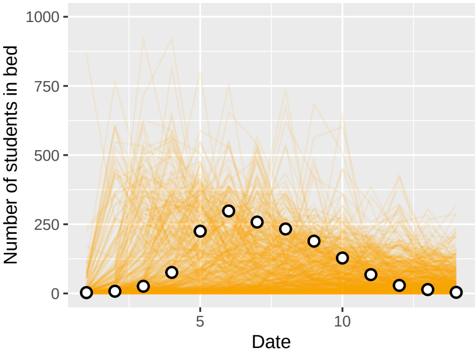


- comparing these prior-based model predictions to data is called a prior predictive check[4]

[4] Gabry et al., *Visualization in Bayesian workflow* (2019)

Prior predictive check

It can also be useful to look at epidemic trajectories:



- prior-based model predictions should have the same magnitude as the data, and allow for some level of flexibility

Prior predictive check

Prior predictive checking brings insight about non-obvious features:

- even if the priors seem weakly informative, there is actually not a lot of leeway
- highly constrained model:
  - if β is high, the epidemic will stop rapidly by lack of susceptibles
  - if β is small, the epidemic will be small
- the negative binomial might lead to problems in extreme situations, e.g. more cases (¿1000) than the overall number of students

Prior predictive check

Prior predictive checks can be used for:

- check for bugs and coding mistakes
- understanding model behaviour
- visualizing priors (especially after transformation)
- supporting and justifying the choice of prior distributions, as it is easier to elicit expert knowledge on measurable quantities rather than abstract parameter values

Back to the limits

We proposed solutions to some of the identified limits:

1. parameter estimates are not everything that we want to know
   → transformed outputs
2. we don't know how well the model fits to the data
   → posterior predictive check
3. the prior distributions we chose seem somewhat arbitrary
   → prior predictive check
4. we don't really know if the model itself is adequate
   → ?

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Simulation-based model validation

As we generally don't have access to the "true" parameter values, we can rely on simulated data for model validation.

Simulation-based model validation

As we generally don't have access to the "true" parameter values, we can rely on simulated data for model validation.

A simulation study consists in two steps:

- simulate data with chosen parameter values
- measure the capacity of the model to recover the values

Simulation-based model validation

As we generally don't have access to the "true" parameter values, we can rely on simulated data for model validation.

A simulation study consists in two steps:

- simulate data with chosen parameter values
- measure the capacity of the model to recover the values

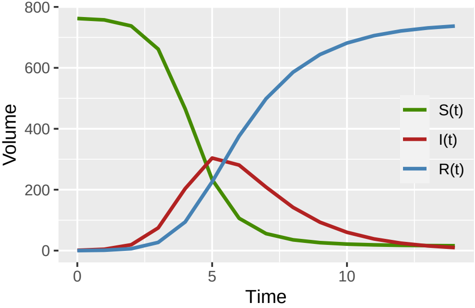
Many advantages for model validation:

- check for bugs and coding mistakes
- check for identifiability issues
- compare different versions of a model
- understand in what situations a model works or not

Simulation-based model validation

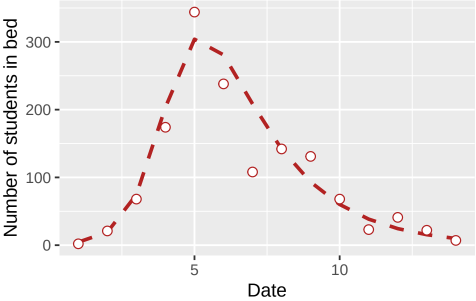
Let's go back to the simple SIR example from the beginning:

- select $\beta = 2$ and $\gamma = 0.5$ (so that $\mathcal{R}_0 = 4$)
- simulate an epidemic in an entirely susceptible population of size $N = 763$ with $I_0 = 1$



Simulation-based model validation

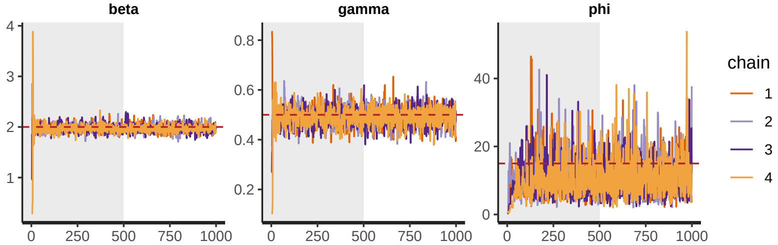
Add noise using a negative binomial distribution with $\phi = 15$



- we only use the daily prevalence $I(t)$ (the “epidemic curve”)

Simulation-based model validation

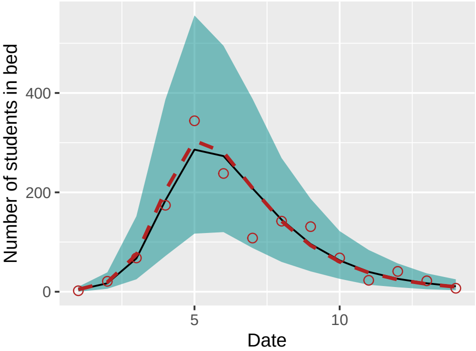
Fit the same model as for the boarding school example



- the chains quickly converge to the selected "true" values

Simulation-based model validation

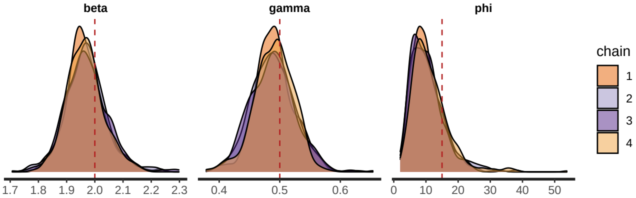
Posterior predictive check:



- the model predictions are very close to the underlying epidemic curve (before adding noise)

Simulation-based model validation

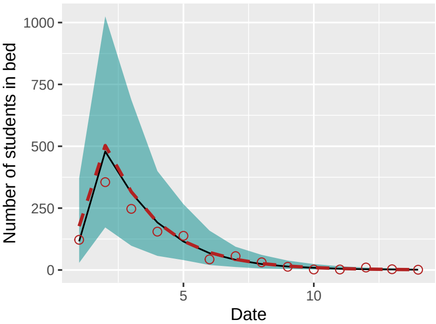
Compare the obtained posterior distributions of the parameters with the selected “true” values:



- no identifiability issue
- $\beta = 2$ and $\gamma = 0.5$ are well-estimated
- $\phi = 15$ is not well-estimated, but the model is still able to distinguish signal and noise (cf. posterior predictive check)

Simulation-based model validation

Next step is to test other parameter values to check if there are areas of the parameter space where the model fails:



- to push the idea to the maximum, test all values from the prior distribution[5]

[5] Talts et al., *Validating Bayesian inference algorithms with simulation-based calibration* (2018)

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Model development

In practice, the situation is often less clear than in the boarding school example:

- sparse or low-quality data
- insufficient domain knowledge
- complex models with many parameters
- many options regarding features and implementation
- uncertainty about the implications of each model feature
- many opportunities for a coding error
- concerns about computational efficiency

→ Bayesian workflow for iterative model development[6]

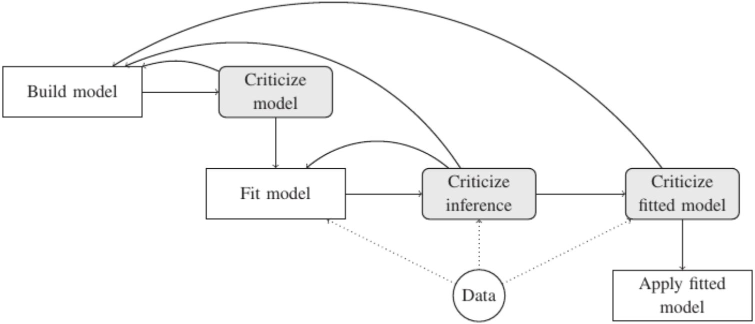
[6] Gelman et al., *Bayesian workflow* (2020)

Model development

Main principles:

- iterative expansion from a simple version
- experiment with simulated data before applying to actual data
- validation and comparison of successive versions
- modular construction
- scaling and transforming parameters
- prior and posterior predictive checking
- checking the inference then the model
- concepts from software development (version control, testing, readability, maintainability, open code)

Box's loop


Build model
Criticize model
Fit model
Criticize inference
Criticize fitted model
Data
Apply fitted model

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

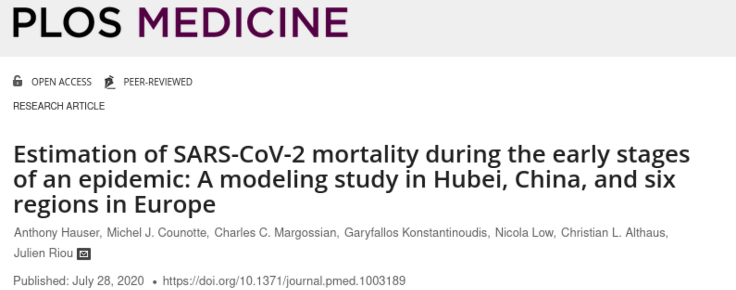
Limitations to the SIR model

In practice, the boarding school example quickly reaches its limits:

- epidemics are not often observed in such a controlled environment (under-ascertainment)
- epidemics are not always left uncontrolled
- data generally consist of daily counts of new cases (incidence) rather than counts of currently sick people (prevalence)
- most infectious diseases have an incubation period (SEIR instead of SIR)
- transmission is generally not homogeneous in the full population (stratification by age, sex...)
- ...

Example

Example with SARS-CoV-2 from Hauser et al. (2020):



PLOS MEDICINE

OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser, Michel J. Counotte, Charles C. Margossian, Garyfallos Konstantinoudis, Nicola Low, Christian L. Althaus, Julien Riou

Published: July 28, 2020 • https://doi.org/10.1371/journal.pmed.1003189

Background

The case fatality ratio (CFR) is computed as the number of deaths divided by the number of reported cases at time t.

Estimated in real time, the CFR is a misleading indicator of mortality due to SARS-CoV-2 because of two opposing biases:

- preferential ascertainment of severe cases: severe cases are both more likely to die and more likely to be diagnosed and reported

→ overestimates mortality

- right-censoring of deaths: there is a long delay between infection and deaths, so that part of the cases at time t will die in the future

→ underestimates mortality

Background

This limits the interpretability of the CFR:

- varies in time (ascending or descending phase)
- varies across countries (depending on surveillance system)

→ April 2020: from 2.4% in Wuhan to 17.8% in Lombardy)

- the real indicator of interest is the infection fatality ratio, i.e. the total number of deaths that occur among people infected with SARS-CoV-2

Objectives

(1) Simulate the dynamics of transmission and mortality of SARS-CoV-2 using publicly available surveillance data

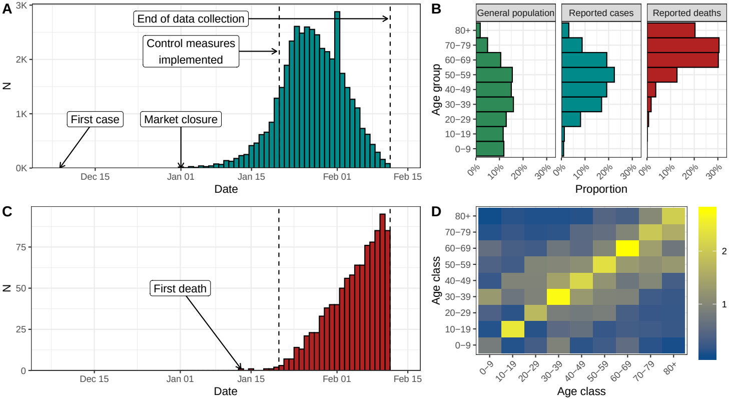
(2) Provide overall and age-stratified estimates of IFR for SARS-CoV-2 infection adjusted for right-censoring and preferential ascertainment

in seven different geographic locations with available data on:

- reported cases by date of disease onset
- deaths linked to SARS-CoV-2 infection by date of death
- age distribution of cases
- age distribution of deaths

Data

In Hubei province (China):



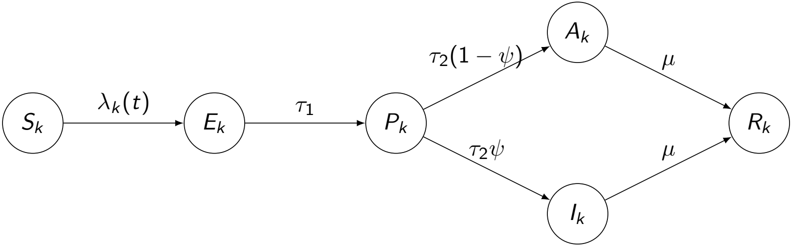
Model development

Specific features and natural history of SARS-CoV-2 infection:

- Incubation period of 5 days (SEIR)
- Pre-symptomatic transmission accounting for 44-48% (SEPIR)
- Symptoms in 81% (95%CrI 71%–89%) of cases (SEPIAR)
- Respiratory virus transmitted through contacts (age stratification)
- Effect of control measures (time-dependent force of infection)
- Mortality is delayed by 20.2 ± 11.6 days (fit to mortality data)

Model development

Final model:



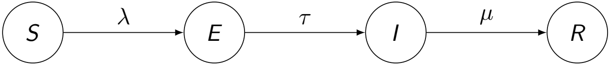
S_k	Susceptible (for age group k)
E_k	Exposed
P_k	Presymptomatic
A_k	Infected asymptomatic
I_k	Infected symptomatic
R_k	Removed

$\lambda_k(t)$	Force of infection (time-dependent)
$1/\tau_1 + 1/\tau_2$	Incubation period (split in two)
ψ	Proportion of symptomatic
$1/\tau_2$	Presymptomatic infectious period
$1/\mu$	Symptomatic infectious period

Model development

Incubation:

- SEIR: adding a compartment E for exposed, i.e. infected but not yet symptomatic



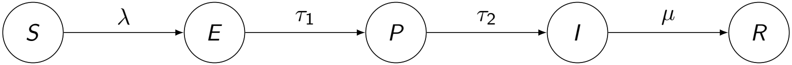
  - τ is the inverse of the incubation period
  - individuals are infectious from symptom onset (when entering I)

$$\lambda = \beta \frac{I}{N}$$

Model development

Pre-symptomatic transmission:

- SEPIR: adding a compartment P for pre-symptomatic, i.e. not yet symptomatic but already infectious



  - the incubation period is split in two phases with rates τ_1 and τ_2
  - individuals are infectious before symptom onset (entering P)

$$\lambda = \beta \frac{P + I}{N}$$

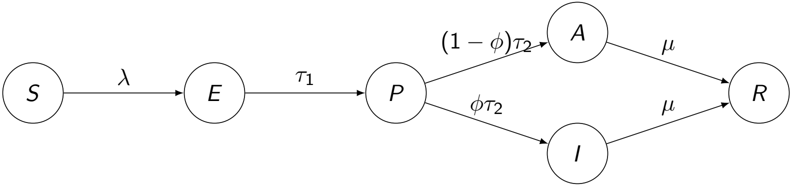
  - we can introduce reduced transmissibility before symptom onset

$$\lambda = \beta \frac{\kappa P + I}{N}$$

Model development

Asymptomatic infections:

- SEPIAR: adding a compartment A for asymptomatic



- we introduce the proportion of symptomatics ϕ
- we can introduce reduced transmissibility for asymptomatics

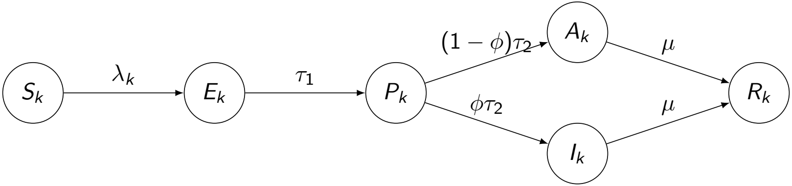
$$\lambda = \beta \frac{\kappa P + I + \kappa A}{N}$$

Model development

Age stratification:

- the transmission of respiratory viruses (influenza virus, rhinovirus...) is highly dependent on age
- the mortality of respiratory infections (even more so for SARS-CoV-2) is highly dependent on age

→ stratification in nine age groups $k \in \{1, \ldots, 9\}$ for (0-9, . . . , 80+)



Model development

Characterizing the force of infection:

- in the simple SIR, the force of infection is defined as the rate at which susceptible individuals acquire infection

$$\lambda = \beta \frac{I}{N}$$

- the transmission rate β can be split in a contact rate c times a probability of transmission upon contact β, so that

$$\lambda = \beta c \frac{I}{N}$$

⚠ the same symbol β is used for the transmission rate and the probability of transmission upon contact depending on context

Model development

- time-dependent force of infection using a forcing function

$$\lambda = f(t)\beta c\frac{I}{N}$$

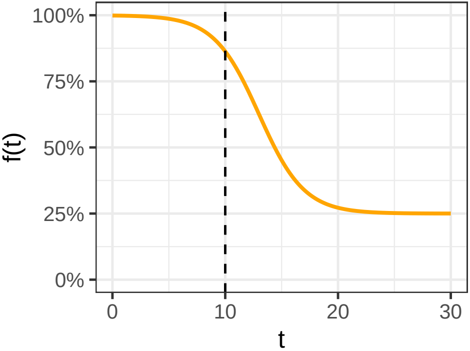
  - to model the effect of control measures, we want a downward function that maps onto the interval $[0, 1]$, e.g. a logistic function:

$$f(t) = \eta + \frac{1-\eta}{1+\exp(\xi(t-t_c-\nu))}$$

  - η is the relative reduction in transmission after control measures
  - ξ is the slope of implementation of the control measures
  - ν is the delay until the control measures are 50% effective (in days after t_c, the date of introduction of control measures).

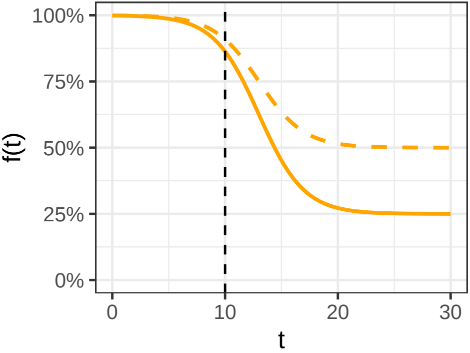
Model development

with $t_c = 10$; $\eta = 0.25$; $\nu = 3$ and $\xi = 0.5$



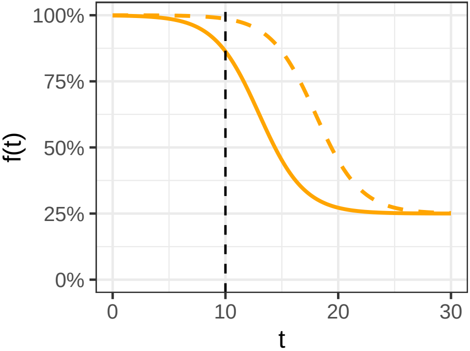
Model development

with $\eta = 0.5$ instead of 0.25, we get



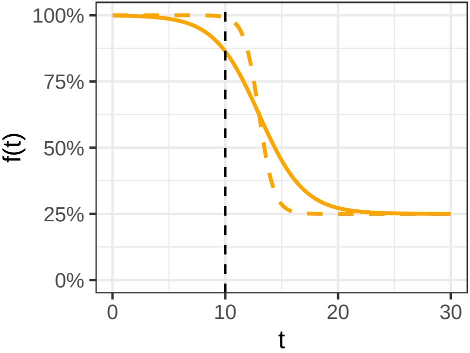
Model development

with $\nu = 8$ instead of 3, we get



Model development

with $\xi = 1.5$ instead of 0.5, we get



Model development

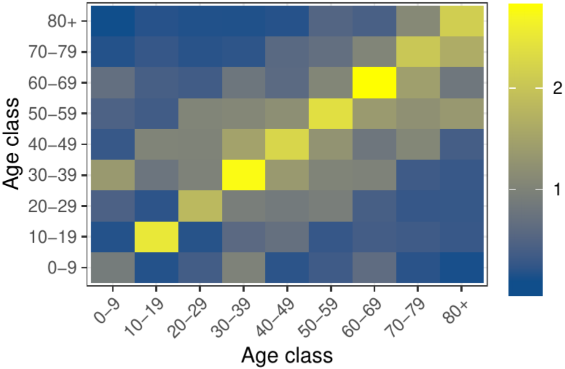
- we account for behaviour differences across age groups:

$$\lambda_k(t) = f(t)\beta \sum_{l=1}^{9} \mathbb{F}_{k,l} \frac{I_l}{N_l}$$

  - one force of infection for each age group k
  - includes a specific contact rate between age group k and each age group l (corresponding to one cell of the contact matrix $\mathbb{F}_{k,l}$)
  - includes the prevalence in age group l (I_l/N_l)

Model development

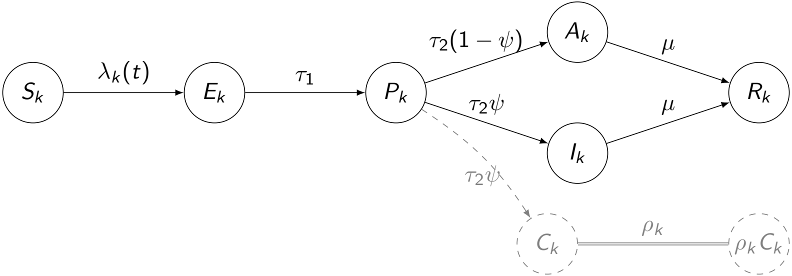
Age-specific contact matrix in China



Model development

Model fit to reported cases:

- obtaining incidence from the ODE output:



  - dummy compartment $C_k(t)$ records the cumulative incidence of symptomatic infections for each age group k

$$\frac{dC_k}{dt} = \tau_2 \psi P_k$$

Model development

- new symptomatic infections by day of symptom onset by age:

$$\Delta C_{k,t} = C_k(t) - C_k(t-1)$$

- new reported infections per day of symptom onset, introducing the age-specific ascertainment proportion ρ_k:

$$A_t = \sum_k^9 \rho_k \Delta C^I_{k,t}$$

- the age distribution of all reported cases up to $t_{\max}$:

$$B_k = \frac{\rho_k C^I_k(t_{\max})}{\sum_k^9 \rho_k C^I_k(t_{\max})}$$

Model development

- A_t can be mapped to reported incidence data $\mathbb{A}$ using a negative binomial likelihood:

$$\Pr(\theta|\mathbb{A}) = \prod_{t=t_1}^{t_{\max}} \text{Neg-Bin}(\mathbb{A}_t|A_t, \phi_1)$$

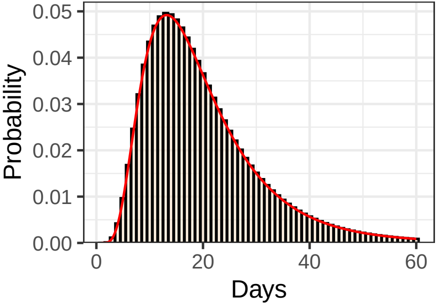
- B_k can be mapped to the age distribution of reported cases $\mathbb{B}$ using a multinomial likelihood:

$$\Pr(\theta|\mathbb{B}) = \text{Multinomial}(\mathbb{B}_1, \ldots, \mathbb{B}_9|B_1, \ldots, B_9)$$

Model development

Model fit to deaths:

- mortality is considered outside of the system of ODEs, using an age-specific mortality parameter ε_k (probability of death given *symptomatic* infection)
- we account for the delay with a discretized log-normal distribution of time from symptom onset to death $\mathbb{I}$ of length 60



Model development

- deaths in age group k at time t $(1 \leq t \leq t_{\text{max}} + 60)$ among people infected up to t_{max}:

$$M_{k,t} = \varepsilon_k \sum_{d}^{60} \Delta C_{k,t-d} \mathbb{I}_d$$

- deaths summed over age groups, assuming that all deaths are reported:

$$M_t = \sum_{k}^{9} M_{k,t}$$

- the age distribution of all deaths occurring up to t_{max}:

$$D_k = \frac{\sum_{t=1}^{t_{\text{max}}} M_{k,t}}{\sum_{t=1}^{t_{\text{max}}} M_t}$$

Model development

- M_t can be mapped to daily death data $\mathbb{C}$ using a negative binomial likelihood:

$$\Pr(\theta|\mathbb{C}) = \prod_{t=t_1}^{t_{\max}} \text{Neg-Bin}(\mathbb{C}_t|M_t, \phi_2)$$

- D_k can be mapped to the age distribution of deaths $\mathbb{D}$ using a multinomial likelihood:

$$\Pr(\theta|\mathbb{D}) = \text{Multinomial}(\mathbb{D}_1, \ldots, \mathbb{D}_9|D_1, \ldots, D_9)$$

This leads to the following joint likelihood:

$$\Pr(\theta|\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}) = \Pr(\theta|\mathbb{A}) \cdot \Pr(\theta|\mathbb{B}) \cdot \Pr(\theta|\mathbb{C}) \cdot \Pr(\theta|\mathbb{D})$$

with $\theta = \{\beta, \eta, \xi, \nu, \psi, \pi, \rho_k, \varepsilon_k, \phi_1, \phi_2\}$.

Model development

Last bits:

- there is an identifiability issue with ρ

→ fix ρ_9 (for 80+) to 100%, assuming that all symptomatic infections among very high risk persons will be reported

- some remaining unknowns (data correction in China, role of children, lower ρ_9...)

→ sensitivity analyses

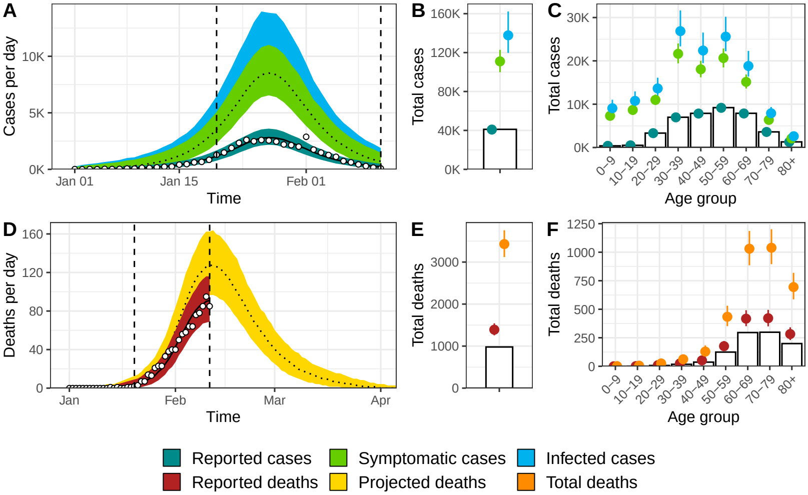
Inference

You know the drill:

- set priors
- prior predictive check
- sampling (on high performance computing cluster $\sim$ 2h)
- basic diagnostic tests
- examine trace plots and chains
- posterior predictive check

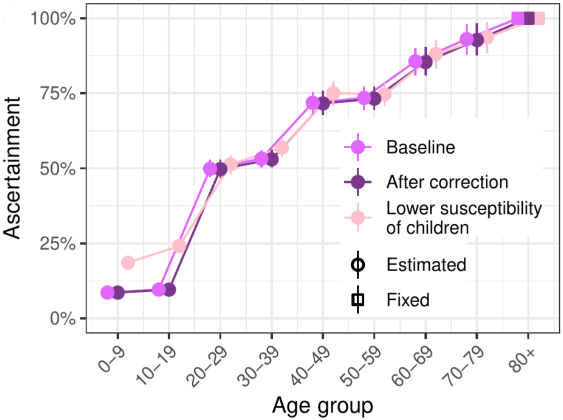
Results in Hubei

Posterior predictive check (Hubei):



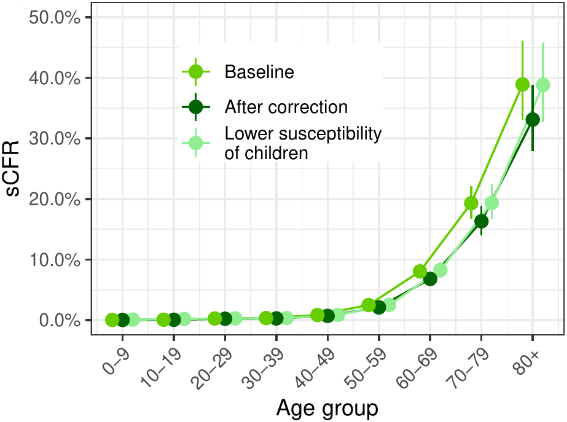
Results in Hubei

Ascertainment (posteriors of ρ_k):



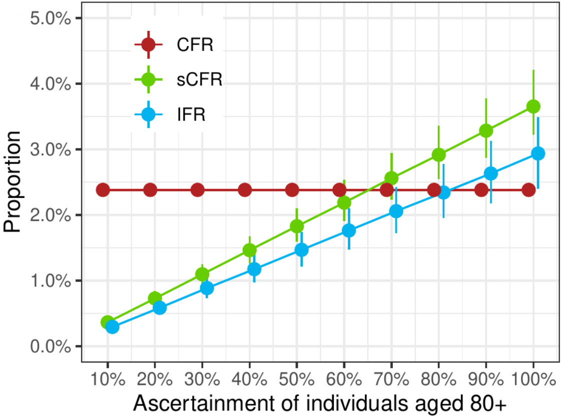
Results in Hubei

Mortality among symptomatics or sCFR (posteriors of ε_k):



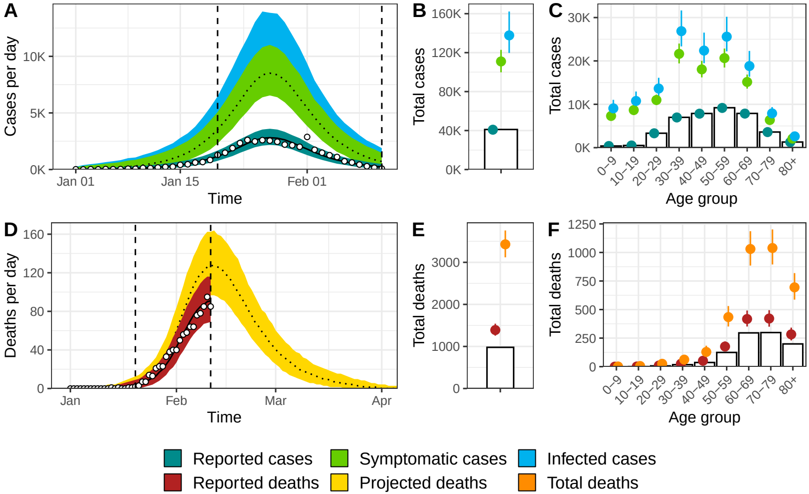
Results in Hubei

Effect on the assumption on ρ_9 on IFR estimate:



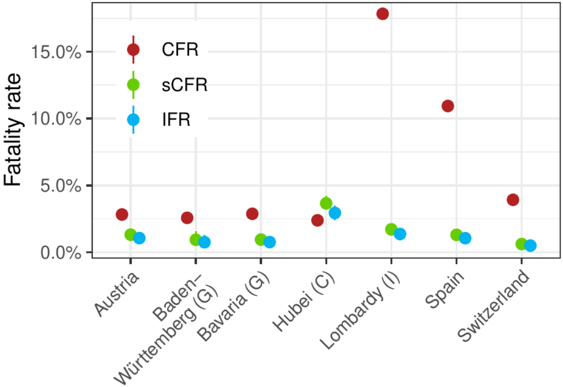
Example

Posterior predictive check (Hubei):



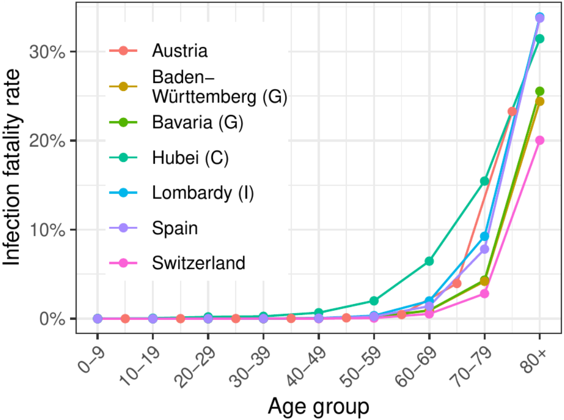
Results in all regions

IFR estimates (compared to CFR and sCFR)



Results in all regions

IFR estimates by age



Conclusions

Location (limit date)	Estimated attack rate	CFR	sCFR	IFR
Hubei, China (11 February)	0.2% (0.2-0.3)	2.0%	3.1% (2.7-3.5)	2.5% (2.1-2.9)
Austria (14 April)	0.8% (0.6-0.9)	2.8%	1.3% (1.1-1.6)	1.1% (0.8-1.3)
Baden-Württemberg, Germany (16 April)	1.9% (1.7-2.2)	2.6%	0.9% (0.6-1.6)	0.7% (0.5-1.3)
Bavaria, Germany (16 April)	2.0% (1.7-2.3)	2.9%	0.9% (0.7-1.3)	0.8% (0.5-1.1)
Lombardy, Italy (25 April)	11.5% (10.1-13.4)	17.8%	1.7% (1.5-2.0)	1.4% (1.1-1.6)
Spain (16 April)	5.7% (5.0-6.6)	10.9%	1.3% (1.2-1.5)	1.0% (0.9-1.2)
Switzerland (23 April)	3.6% (2.9-4.5)	3.9%	0.6% (0.5-0.8)	0.5% (0.4-0.6)

- IFR estimates adjusted for under-ascertainment and right-censoring are more similar across countries than CFR
- still some degree of heterogeneity
- clear increase of mortality with age

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

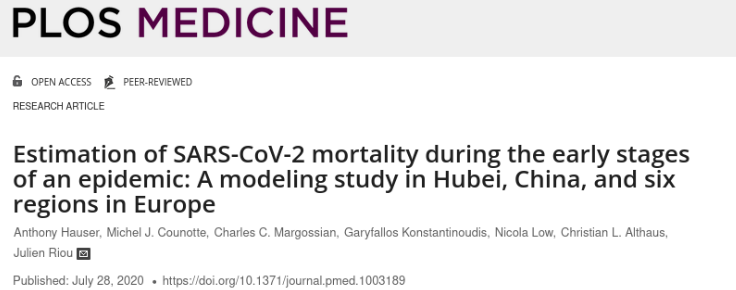
Limitations to the SIR model

In practice, the boarding school example quickly reaches its limits:

- epidemics are not often observed in such a controlled environment (under-ascertainment)
- epidemics are not always left uncontrolled
- data generally consist of daily counts of new cases (incidence) rather than counts of currently sick people (prevalence)
- most infectious diseases have an incubation period (SEIR instead of SIR)
- transmission is generally not homogeneous in the full population (stratification by age, sex...)
- ...

Example

Example with SARS-CoV-2 from Hauser et al. (2020):



PLOS MEDICINE

OPEN ACCESS PEER-REVIEWED

RESEARCH ARTICLE

Estimation of SARS-CoV-2 mortality during the early stages of an epidemic: A modeling study in Hubei, China, and six regions in Europe

Anthony Hauser, Michel J. Counotte, Charles C. Margossian, Garyfallos Konstantinoudis, Nicola Low, Christian L. Althaus, Julien Riou

Published: July 28, 2020 • https://doi.org/10.1371/journal.pmed.1003189

Background

The case fatality ratio (CFR) is computed as the number of deaths divided by the number of reported cases at time t.

Estimated in real time, the CFR is a misleading indicator of mortality due to SARS-CoV-2 because of two opposing biases:

- preferential ascertainment of severe cases: severe cases are both more likely to die and more likely to be diagnosed and reported

→ overestimates mortality

- right-censoring of deaths: there is a long delay between infection and deaths, so that part of the cases at time t will die in the future

→ underestimates mortality

Background

This limits the interpretability of the CFR:

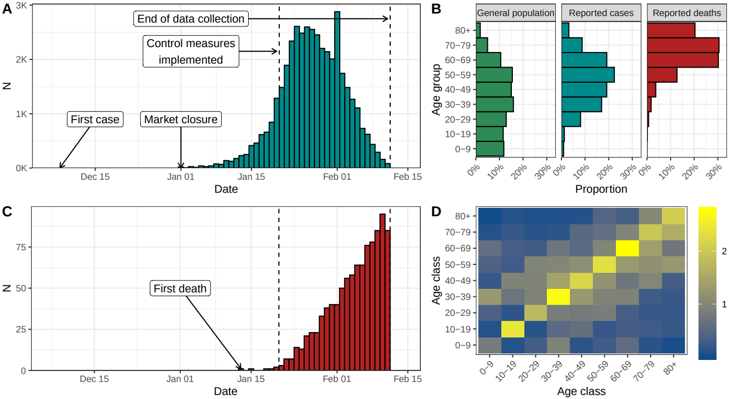
- varies in time (ascending or descending phase)
- varies across countries (depending on surveillance system)

→ April 2020: from 2.4% in Wuhan to 17.8% in Lombardy)

- the real indicator of interest is the infection fatality ratio (IFR), i.e. the total number of deaths that occur among people infected with SARS-CoV-2

Data

In Hubei province (China):



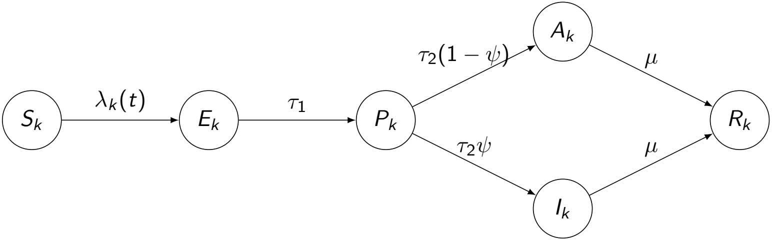
Model development

Specific features and natural history of SARS-CoV-2 infection:

- Incubation period of 5 days (SEIR)
- Pre-symptomatic transmission accounting for 44-48% (SEPIR)
- Symptoms in 81% (95%CrI 71%–89%) of cases (SEPIAR)
- Respiratory virus transmitted through contacts (age stratification)
- Effect of control measures (time-dependent force of infection)
- Mortality is delayed by 20.2 ± 11.6 days (fit to mortality data)

Model

Final model:

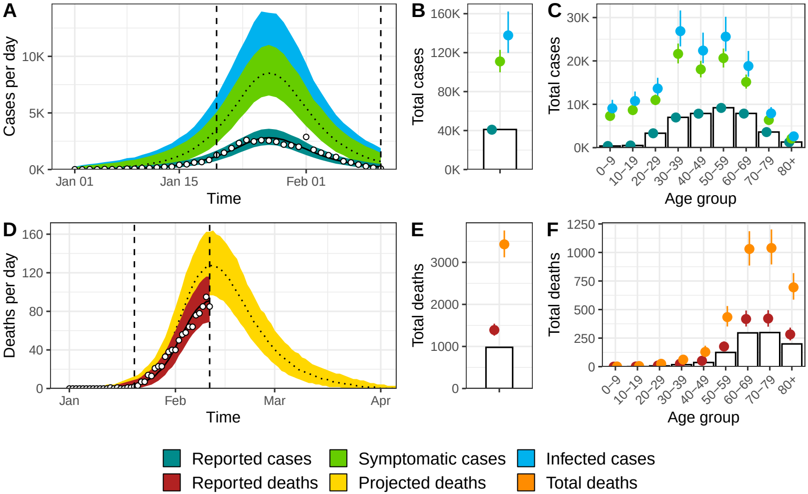


S_k	Susceptible (for age group k)
E_k	Exposed
P_k	Presymptomatic
A_k	Infected asymptomatic
I_k	Infected symptomatic
R_k	Removed

$\lambda_k(t)$	Force of infection (time-dependent)
$1/\tau_1 + 1/\tau_2$	Incubation period (split in two)
ψ	Proportion of symptomatic
$1/\tau_2$	Presymptomatic infectious period
$1/\mu$	Symptomatic infectious period

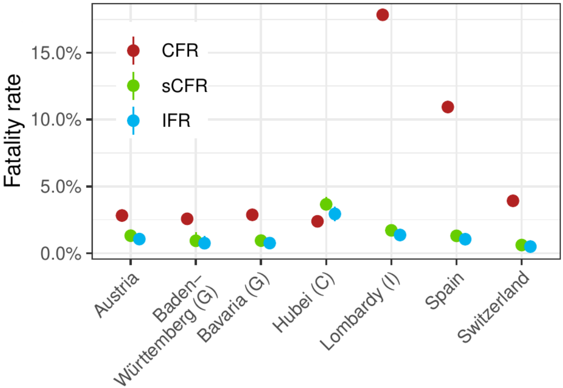
Posterior predictive check

Posterior predictive check (Hubei):



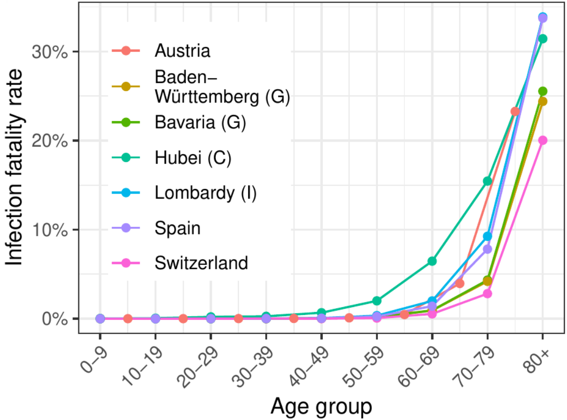
Results

IFR estimates (compared to CFR and sCFR)



Results

IFR estimates by age



Results

Location (limit date)	Estimated attack rate	CFR	sCFR	IFR
Hubei, China (11 February)	0.2% (0.2-0.3)	2.0%	3.1% (2.7-3.5)	2.5% (2.1-2.9)
Austria (14 April)	0.8% (0.6-0.9)	2.8%	1.3% (1.1-1.6)	1.1% (0.8-1.3)
Baden-Württemberg, Germany (16 April)	1.9% (1.7-2.2)	2.6%	0.9% (0.6-1.6)	0.7% (0.5-1.3)
Bavaria, Germany (16 April)	2.0% (1.7-2.3)	2.9%	0.9% (0.7-1.3)	0.8% (0.5-1.1)
Lombardy, Italy (25 April)	11.5% (10.1-13.4)	17.8%	1.7% (1.5-2.0)	1.4% (1.1-1.6)
Spain (16 April)	5.7% (5.0-6.6)	10.9%	1.3% (1.2-1.5)	1.0% (0.9-1.2)
Switzerland (23 April)	3.6% (2.9-4.5)	3.9%	0.6% (0.5-0.8)	0.5% (0.4-0.6)

- IFR estimates adjusted for under-ascertainment and right-censoring are more similar across countries than CFR
- still some degree of heterogeneity
- clear increase of mortality with age

Outline

Models of disease transmission (day 1)

Bayesian inference

Bayesian inference with Stan

Fitting a simple SIR model

Beyond inference: generated quantities

Simulation-based model validation

The Bayesian workflow

Extending from the simple SIR model (long version)

Extending from the simple SIR model (short version)

Conclusions

Conclusions

General comments:

- develop models from the data-generating mechanisms
- use Bayesian inference to propagate uncertainty from the data (and priors) into the results (and forecasts)
- carefully examine the modelling process (Bayesian workflow)
- be transparent about assumptions (open code)

Try by yourself!

- `https://github.com/ISPMBern/winter-school-modeling-course-2022/tree/main/day2`
- julien.riou@ispm.unibe.ch

Acknowledgements & ressources

Textbooks:

- Gelman et al., *Bayesian data analysis* (book available online)
- Richard McElreath, *Statistical rethinking* (book and (video)

Articles:

- Grinsztajn et al., *Bayesian workflow for disease transmission modeling in Stan* (2021)
- Betancourt, *An introduction to Stan* (2020)
- Zelner et al., *Accounting for uncertainty during a pandemic* (2021)
- Gabry et al., *Visualization in Bayesian workflow* (2019)
- Talts et al., *Validating Bayesian inference algorithms with simulation-based calibration* (2018)
- Gelman et al., *Bayesian workflow* (2020)

Ressources:

- Stan forums `https://discourse.mc-stan.org/`
- Chi Feng, *MCMC interactive gallery* `https://chi-feng.github.io/mcmc-demo/app.html`