

# Bayesian workflow for disease transmission modelling in Stan

Swiss Epidemiology Winter School 2021

**Julien Riou, MD PhD**

Institute of Social and Preventive Medicine, University of Bern, Switzerland

# Preface

- Objective: fit transmission models in Stan
- Based on Grinsztajn et al., 2020 ([link](#) – v2 upcoming)
- Prerequisites:
  - introduction to infectious disease modelling
  - general understanding of Bayesian inference
  - basic programming with R and Stan 2.21
- All material is available on  
[https://github.com/jriou/bayesian\\_workflow](https://github.com/jriou/bayesian_workflow)

# Outline

## Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

Fitting a simple SIR model

Simulation-based model assessment

The Bayesian workflow

Scaling-up ODE-based models

Extensions

# Introduction

Models of disease transmission:

- Interpretability: **mechanistic**, phenomenological
- Granularity: agent-based, **population-based**
- Framework: **deterministic**, stochastic
- Data-generating mechanisms: incubation, contagion, immunity...

# Introduction

Models of disease transmission:

- Interpretability: **mechanistic**, phenomenological
- Granularity: agent-based, **population-based**
- Framework: **deterministic**, stochastic
- Data-generating mechanisms: incubation, contagion, immunity...

Mechanistic + population-based + deterministic

→ **ordinary differential equations (ODE)-based compartmental model**

# Introduction

ODE-based compartmental model:

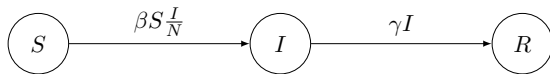
- Divide the population into homogeneous groups (**compartments**)
- Define the **flows** between compartments with ODEs
- Define initial conditions (at  $t_0$ )
- Solve for the time-dependent volume in each compartment

# 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:

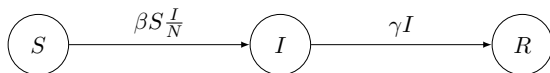


$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$

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

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

# Introduction



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

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

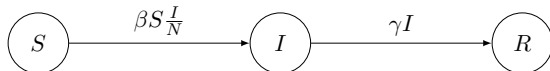
$$\frac{dR}{dt} = \gamma I$$

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  $t$ )
- $\beta$  is the **infectious contact rate** (per day per person)
- $\gamma$  is the **recovery rate** (1/infectious period)



# Introduction



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

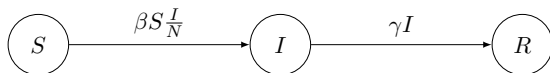
$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I$$

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

**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,  $\beta S I(t)/N$  people become infected (the **force of infection**)

# Introduction



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

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

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

## Assumptions behind the SIR model:

- homogeneous mixing
- $\beta$  and  $\gamma$  constant over time
- all infections are observed
- no incubation, exponentially-distributed recovery
- lifelong immunity
- stable population

# Introduction

Simulate in R with package `deSolve`:

- set compartments and differential equations

```
> ## Set model ----
> seir = function(t, x, parms, ...) {
+   with(as.list(c(parms, x)), {
+     dS = - beta*S*I/(S+I+R)
+     dI = beta*S*I/(S+I+R) - gamma*I
+     dR = gamma*I
+     list(c(dS, dI, dR))
+   })
+ }
```

# Introduction

Simulate in R with package `deSolve`:

- set compartments and differential equations

```
> ## Set model ----  
> seir = function(t, x, parms, ...) {  
+   with(as.list(c(parms, x)), {  
+     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  $\beta = 0.8$ ;  $\rho = 1/7$ ;  $S_0 = 100,000 - 50$ ;  $I_0 = 50$  and  $R_0 = 0$

```
> ## Set parameters ----  
> pars = c(beta = 0.8,  
+           gamma = 1/7  
+ )
```

```
> ## Set initial values ----  
> N_0 = 100000  
> I_0 = 50  
> inits = c(  
+   S = N_0 - I_0,  
+   I = I_0,  
+   R = 0  
+ )
```

# Introduction

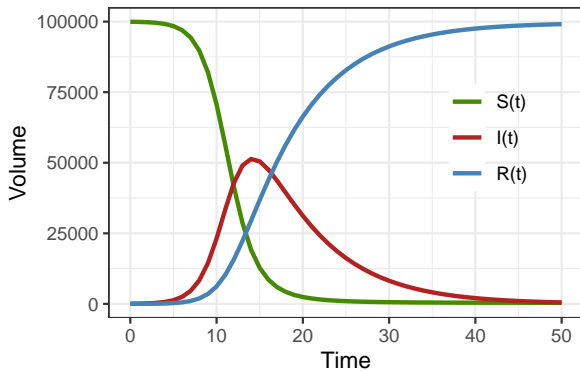
- solve the ODE system numerically (Runge-Kutta 4th order) to obtain unique solutions for  $S(t)$ ,  $I(t)$  and  $R(t)$

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

```
> ## Simulate ----
> times = seq(0,50,by=1)
> sim_data = ode(inits, times, seir, 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
```

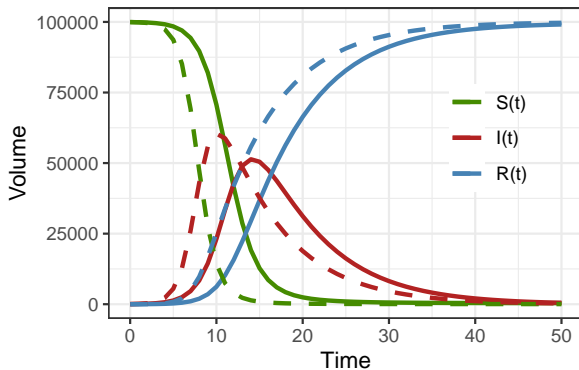
# Introduction

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



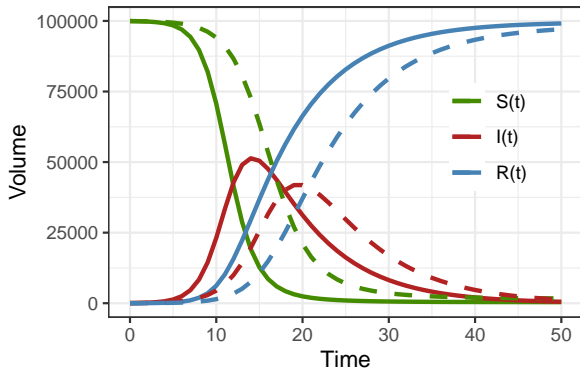
# Introduction

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



# Introduction

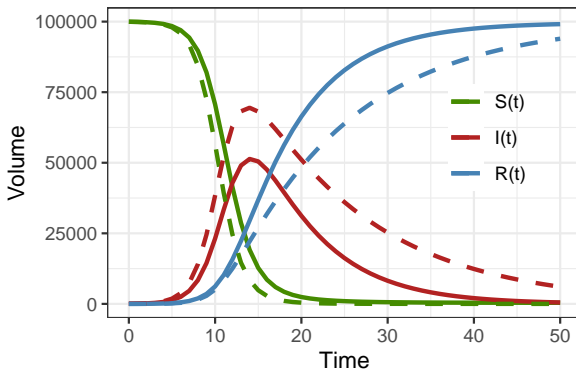
with  $\beta = 0.6$  instead of 0.8, we get





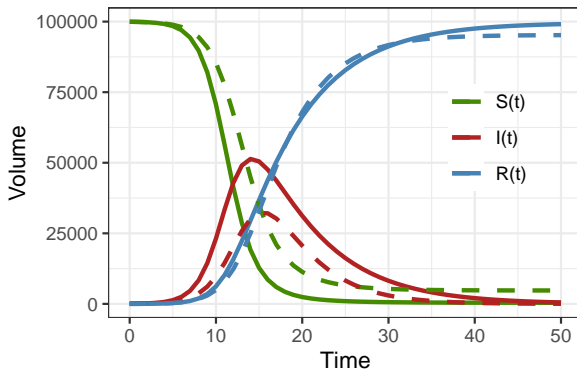
# Introduction

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



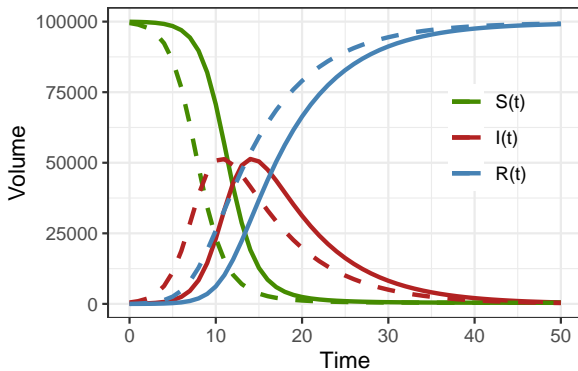
# Introduction

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



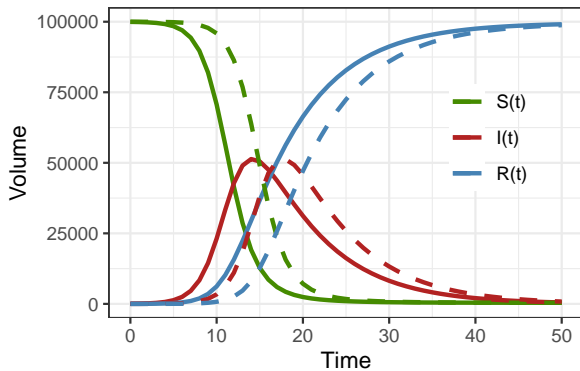
# Introduction

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



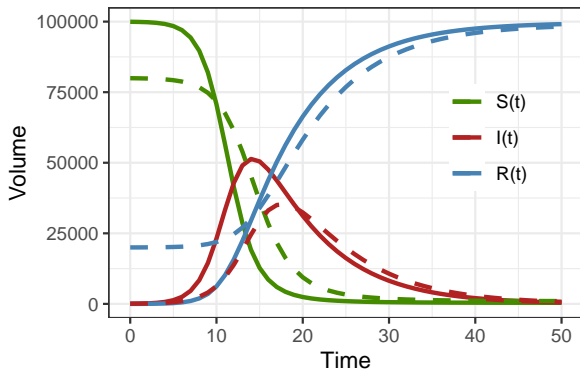
# Introduction

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



# Introduction

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



# Introduction

Compartmental models have many uses:

- formalize and put numerical values on **general concepts** (herd immunity, vaccination threshold...)
- get **mechanistic insight** about an epidemic (transmissibility levels, drivers of transmission)

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

- produce precise **forecasts** (based on mechanisms)

# Introduction

Compartmental models have many uses:

- formalize and put numerical values on **general concepts** (herd immunity, vaccination threshold...)
- get **mechanistic insight** about an epidemic (transmissibility levels, drivers of transmission)

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

- produce precise **forecasts** (based on mechanisms)

→ all these uses are based on **numerical values** for  $\beta$ ,  $\rho$  and the initial conditions and their **uncertainty**

# Introduction

Enters **Bayesian inference**:

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



# Introduction

Enters **Bayesian inference**:

- infer parameter values by **integrating information from data and from 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

Introduction

**Basics of Bayesian inference**

Bayesian modelling with Stan

Fitting a simple SIR model

Simulation-based model assessment

The Bayesian workflow

Scaling-up ODE-based models

Extensions

# Bayesian inference

General principle of Bayesian inference<sup>1,2</sup>:

**(1)** specify a complete Bayesian model

- consider data  $y = \{y_1, \dots, y_n\}$  that are noisy measurement of an unknown quantity (parameter)  $\theta$
- encode the data-generating processes in an **observation model** (likelihood)

$$p(y|\theta) = \prod_n^N \text{normal}(y_n|\theta, 1)$$

- encode domain knowledge in a **prior distribution**

$$p(\theta) = \text{normal}(0, 1)$$

---

<sup>1</sup>Gelman et al. *Bayesian Data Analysis*

<sup>2</sup>Michael Betancourt, *Introduction to Stan*

# Bayesian inference

(2) reverse the question:

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

- the answer comes from Bayes' rule:

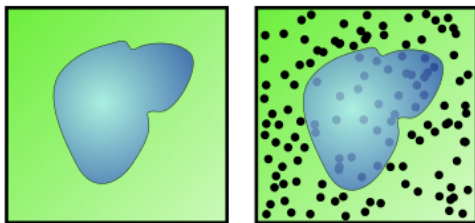
$$p(\theta \mid y) \propto p(\theta)p(y \mid \theta)$$

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

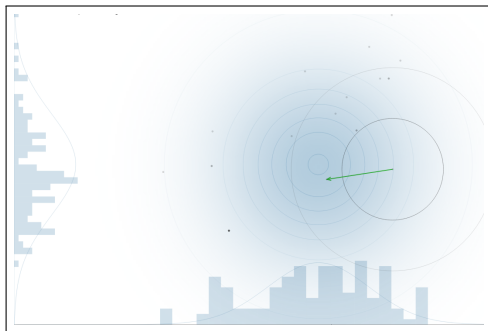
- Monte Carlo methods



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

# Bayesian inference

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



**Figure:** An MCMC sampler exploring a bivariate normal distribution (<https://chi-feng.github.io/mcmc-demo/app.html>).

# Outline

Introduction

Basics of Bayesian inference

**Bayesian modelling with Stan**

Fitting a simple SIR model

Simulation-based model assessment

The Bayesian workflow

Scaling-up ODE-based models

Extensions

# Bayesian modelling with Stan

Stan is a probabilistic programming framework for Bayesian inference

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



# Bayesian modelling with Stan

Programming in Stan is structured in **blocks**:

- the **data** block defines known variables and constants

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

- the **parameters** block defines unknown variables (parameters)

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

- the **model** block defines the prior distribution(s) and the observation model(s)

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

- save in `model_linear.stan`

# Bayesian modelling with Stan

We then explore the target with Stan's MCMC **sampler**:

- load `rstan` package

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

- simulate  $N = 50$  data points with  $\theta = 0.7$

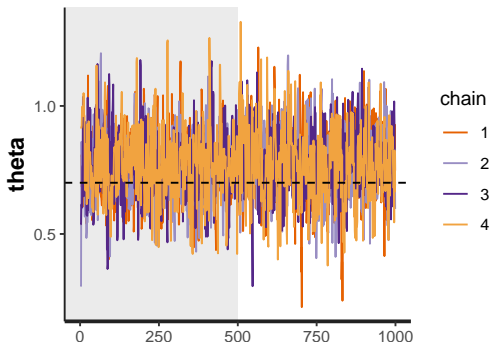
```
## Simulate data ----  
N = 50  
theta = 0.7  
y = rnorm(N,theta,1)  
input_data = list(N=N,y=y)
```

- run MCMC sampling

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

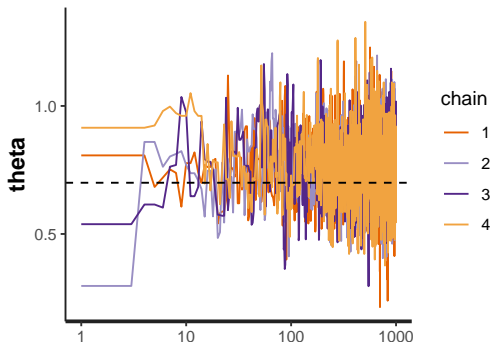
# Bayesian modelling with Stan

We use **multiple chains** that should converge after warm-up



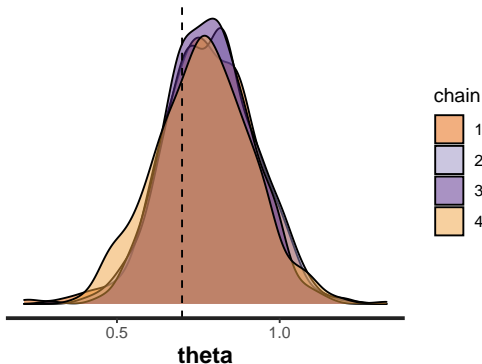
# Bayesian modelling with Stan

We use **multiple chains** that should converge after warm-up



# Bayesian modelling with Stan

The post-warm-up samples of  $\theta$  approximate its **posterior distribution**



# Bayesian modelling with Stan

We run **basic diagnosis tools**: divergences, tree depth, energy

```
> 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.
```

# Bayesian modelling with Stan

Printing the object gives:

- **diagnostics**: effective sample size, Gelman-Rubin  $\hat{R}$
- **inference**: full posterior distribution of  $\theta$

```
> print(fit)
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  0.78    0.01 0.14   0.51   0.69   0.78   0.87   1.05   760    1
lp__ -18.30    0.03 0.73 -20.28 -18.44 -18.03 -17.86 -17.81   799    1

Samples were drawn using NUTS(diag_e) at Thu Nov 12 19:15:33 2020.
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).
```

# Outline

Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

**Fitting a simple SIR model**

Simulation-based model assessment

The Bayesian workflow

Scaling-up ODE-based models

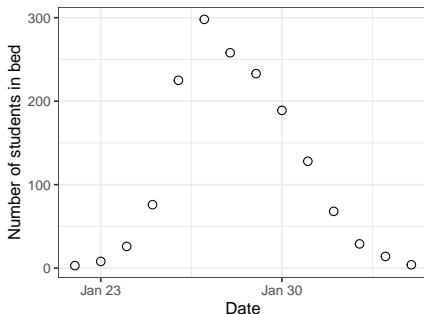
Extensions



# Fitting a simple SIR

Example data: 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** data)



# Fitting a simple SIR

Specifying the model:

- prevalence data:  $\mathbb{I}_t$  with  $t \in \{1, \dots, 14\}$
- parameters to estimate:  $\theta = \{\beta, \gamma, \phi\}$
- parameters that will remain fixed:  $\{S_0 = 762, I_0 = 1, R_0 = 0\}$
- map data  $\mathbb{I}_t$  to SIR model output  $I(t)$  using an observation model with an appropriate **probability distribution**:

$$\Pr(\mathbb{I}|\theta) = \prod_{t=1}^{14} \text{neg-bin}(\mathbb{I}_t | I(t), \phi)$$

- **prior distributions**

$$\Pr(\beta) = \text{exponential}(1)$$

$$\Pr(1/\gamma) = \text{normal}(2, 0.5)$$

$$\Pr(1/\phi) = \text{exponential}(5)$$

## Fitting a simple SIR

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!

## Fitting a simple SIR

We declare the data variables in the data block

```
data {  
  int<lower=1> T;  
  real y0[3];  
  real t0;  
  real ts[T];  
  int N;  
  int cases[T];  
}
```

## Fitting a simple SIR

We declare the data variables in the data block

```
data {  
  int<lower=1> T;  
  real y0[3];  
  real t0;  
  real ts[T];  
  int N;  
  int cases[T];  
}
```

and define additional data variables in transformed data

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

## Fitting a simple SIR

Similarly, parameters are declared in the `parameters` block

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

⚠ It sometimes makes more sense to transform some parameters (e.g., recovery rate  $\gamma$  and overdispersion  $\phi$ ) to improve interpretability

## Fitting a simple SIR

In transformed parameters, we define additional parameters and solve the ODE system

```
transformed parameters{  
  real y[T,3];  
  real phi = 1. / phi_inv;  
  real gamma = 1. / recovery_time;  
  real theta[2];  
  theta[1] = beta;  
  theta[2] = gamma;  
  
  y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);  
}
```

# Fitting a simple SIR

```
y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);
```

Two crucial points:

- be careful about the **formats and signatures**
  - the ODE output `y` is an array of size  $T \times 3$  (number of time steps and number of compartments)
  - `sir` is the name of the function defined in the `function` block
  - `y0` is an array of size 3 defined in the `data` block
  - `ts` is an array of size  $T$  defined in the `data` block
  - `theta` is an array of size 2 storing the parameters
  - `x_r` is defined as empty in transformed data, but can be used to store fixed real values
  - `x_i` is an array of size 1 storing the population size  $N$  (can also be used to store fixed integer values)



# Fitting a simple SIR

```
y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);
```

Two crucial points:

- two ODE solvers are available:
  - `integrate_ode_rk45` uses the Runge-Kutta method (quicker but non-adapted to stiff systems)
  - `integrate_ode_bdf` uses the backward differentiation method (slower but adapted to stiff systems)

## Fitting a simple SIR

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

```
model {  
  // priors  
  beta ~ exponential(1);  
  recovery_time ~ normal(2,0.5);  
  phi_inv ~ exponential(5);  
  
  // observation model  
  cases ~ neg_binomial_2(col(to_matrix(y),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`

## Fitting a simple SIR

Last, we add a `generated quantities` block that does not influence sampling and can be used for “post-processing”:

- reproduction number  $\mathcal{R}_0 = \beta/\gamma$
- model predictions of prevalence from the negative binomial

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

# Fitting a simple SIR

In summary:

- `functions`: define the ODE system (⚠ signature and formats)
- `data`: declare data variables that will be provided
- `transformed data`: additional quantities that can be computed internally or from data variables
- `parameters`: declare parameters (⚠ boundaries)
- `transformed parameters`: quantities that can be computed internally or from data or parameters variables, including the ODE output (⚠ signature and format)
- `model`: priors and observation model
- `generated quantities`: additional quantities that can be computed without influencing the sampling

## Fitting a simple SIR

As before, we conduct the inference from R with the package `rstan`:

```
## Format input ----  
# prevalence data  
cases = influenza_england_1978_school$in_bed  
N = 763  
n_days = 14  
t0 = 0  
t = 1:n_days  
  
# initial conditions  
i0 = 1  
s0 = N - i0  
r0 = 0  
y0 = c(s0, i0, r0)  
  
# put into list  
input_data = list(T = n_days, y0 = y0, t0 = t0, ts = t, N = N, cases = cases)
```

⚠ data is put in a list with names matching the data block in Stan

# Fitting a simple SIR

Hit the inference button!

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

# Fitting a simple SIR

Run basic diagnosis tools:

```
> check_hmc_diagnostics(fit)
```

Divergences:

0 of 2000 iterations ended with a divergence.

Tree depth:

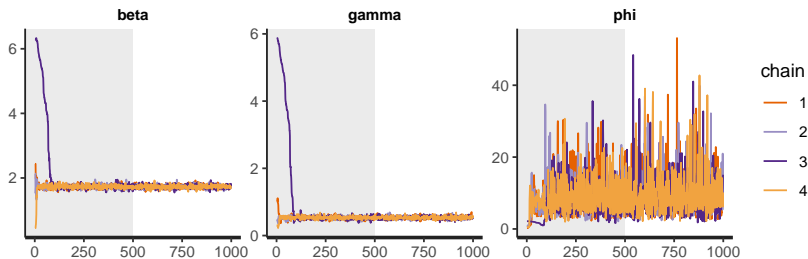
0 of 2000 iterations saturated the maximum tree depth of 10.

Energy:

E-BFMI indicated no pathological behavior.

# Fitting a simple SIR

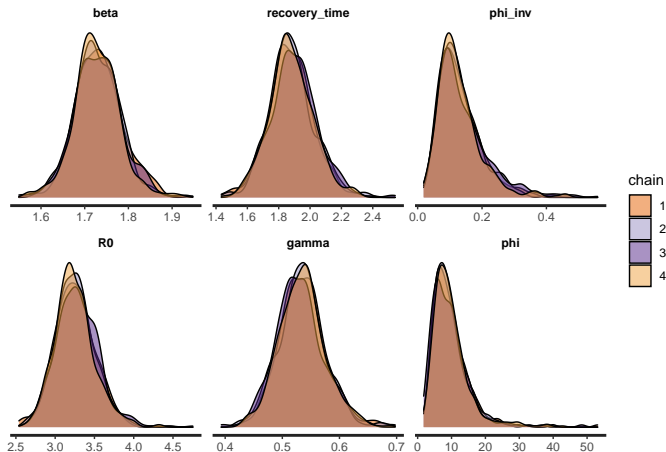
Examine trace plots:





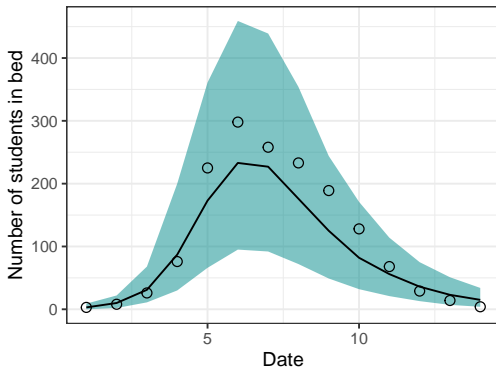
# Fitting a simple SIR

Examine chain mixing:



# Fitting a simple SIR

Posterior predictive checking (always show!):



# Fitting a simple SIR

Print the results:

```
> print(fit, pars=c("beta", "gamma", "phi", "R0", "recovery_time"))
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	0.06	1.61	1.70	1.73	1.76	1.85	1049	1
gamma	0.53	0.00	0.04	0.44	0.50	0.53	0.56	0.63	1382	1
phi	9.61	0.22	6.13	2.94	5.72	8.31	11.80	23.40	743	1
R0	3.27	0.01	0.29	2.79	3.09	3.25	3.42	3.96	1403	1
recovery_time	1.89	0.00	0.16	1.59	1.79	1.88	1.98	2.25	1410	1

Samples were drawn using NUTS(diag\_e) at Thu Nov 12 19:10:02 2020.  
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).

## Fitting a simple SIR

### Conclusions:

- we estimate  $\mathcal{R}_0$  to 3.3 (95% credible interval: 2.8 to 4.0)
- this corresponds to the direct estimation from the final size of the epidemic  $q = 512/763 = 0.67$

$$\mathcal{R}_0 = 1/(1 - q) = 3.03$$

- based on **many assumptions**:
  - common to all SIRs (homogeneous mixing, no incubation...)
  - prior distributions (especially on the recovery period)
  - complete ascertainment, no asymptomatics
  - no initial immunity

# Outline

Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

Fitting a simple SIR model

**Simulation-based model assessment**

The Bayesian workflow

Scaling-up ODE-based models

Extensions

# Simulation-based model assessment

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

- incomplete data, insufficient domain knowledge
- uncertainty on **necessary model features**

**Fake data** can be used to probe the model and better understand its behaviour:

- prior predictive checking<sup>3</sup>
- simulation study

---

<sup>3</sup>Gabry et al. *Visualization in Bayesian workflow*

# Simulation-based model assessment

Prior predictive checking consists in simulating data from the priors:

- visualize priors (especially after transformation)
- this shows the range of data compatible with the model
- it helps understand the adequacy of the chosen priors, as it is often easier to elicit expert knowledge on measureable quantities of interest rather than abstract parameter values

# Simulation-based model assessment

**Prior predictive checking** consists in simulating data from the priors:

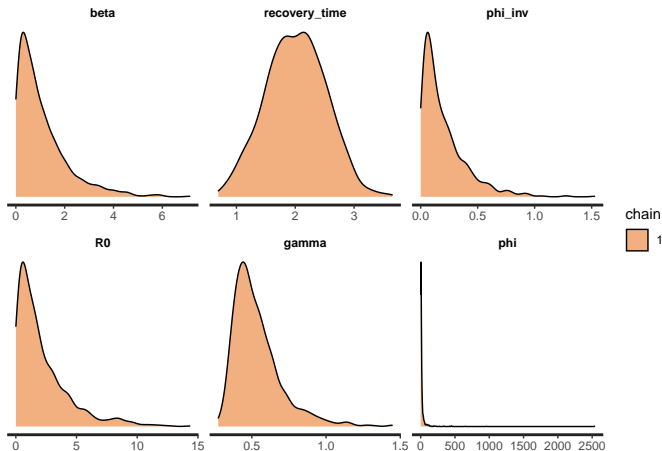
- visualize priors (especially after transformation)
- this shows the range of data compatible with the model
- it helps understand the **adequacy of the chosen priors**, as it is often easier to elicit expert knowledge on measureable quantities of interest rather than abstract parameter values
- remove (or switch off) the likelihood from the `model` block

```
// observation model  
if(switch_likelihood==1) cases ~ neg_binomial_2(col(to_matrix(y),2), phi);
```



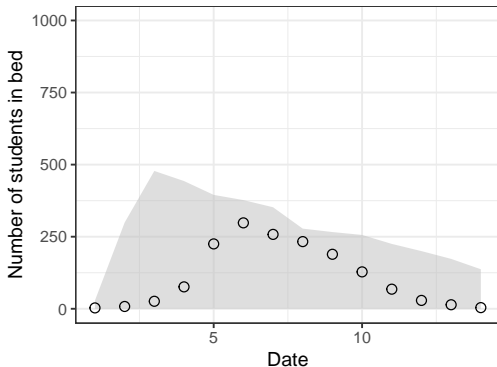
# Simulation-based model assessment

Simulating priors in the boarding school example:



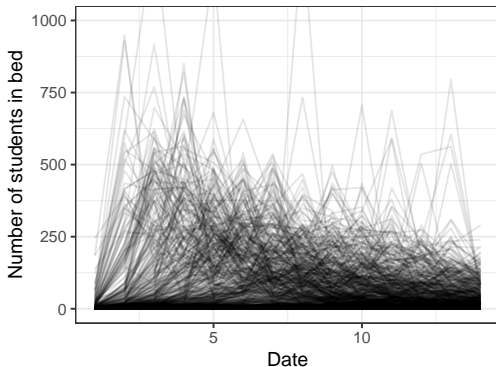
# Simulation-based model assessment

Prior predictive checking: simulating epidemic trajectories



# Simulation-based model assessment

Prior predictive checking: simulating epidemic trajectories



# Simulation-based model assessment

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  $\beta$  is high, the epidemic will stop rapidly by lack of susceptibles
  - if  $\beta$  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

# Simulation-based model assessment

A **simulation study** consists in two steps:

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

# Simulation-based model assessment

A **simulation study** consists in two steps:

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

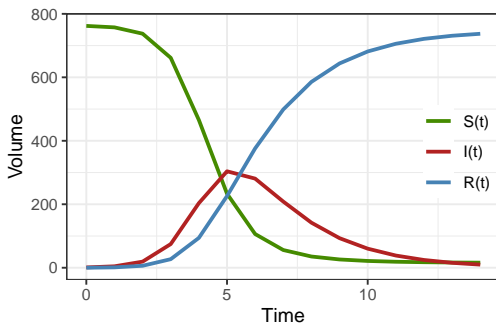
Many advantages:

- 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 assessment

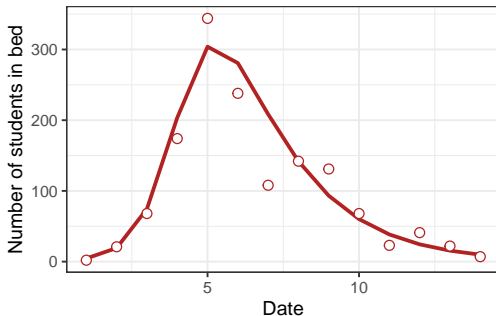
Let's go back to the **simple SIR example** from the beginning:

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



# Simulation-based model assessment

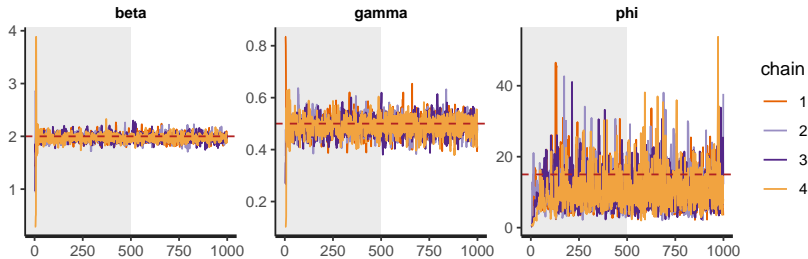
Add noise with a **negative binomial** distribution with  $\phi = 15$





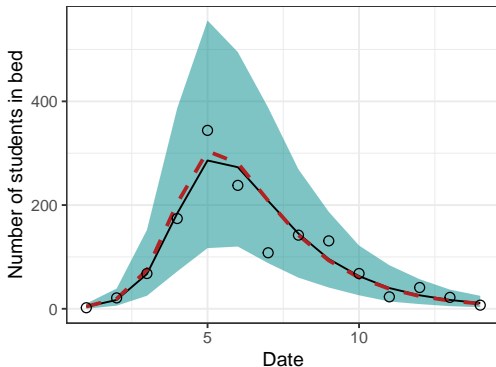
# Simulation-based model assessment

Fit the **same model** as for the boarding school example



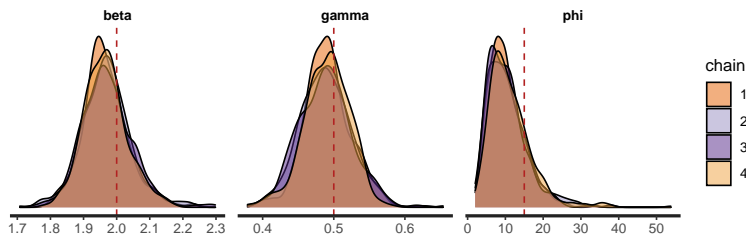
# Simulation-based model assessment

## Posterior predictive checking



# Simulation-based model assessment

Compare the posterior distributions of the parameters with the “truth”



- no identifiability issue
- $\beta$  and  $\gamma$  are well estimated, but  $\phi$  is not
- try with other values to understand when does the model break<sup>4</sup>

<sup>4</sup>Talts et al. *Simulation-based calibration*

# Outline

Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

Fitting a simple SIR model

Simulation-based model assessment

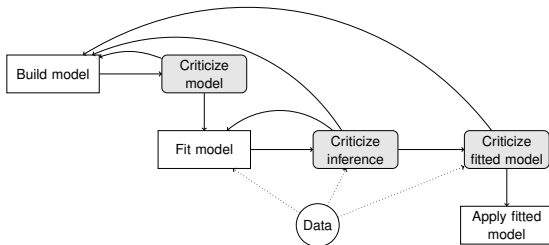
**The Bayesian workflow**

Scaling-up ODE-based models

Extensions

# The Bayesian workflow

Bayesian **workflow**: model development as an iterative process<sup>5</sup>



<sup>5</sup>Gelman et al. *Bayesian workflow*

# The Bayesian workflow

## (1) Build a complete model:

- encode the data-generating processes with observation models
- encode domain knowledge with prior distributions

## (2) *Criticize the model* with fake data

- **prior predictive checking**: is the model adequate?
- **simulation-based calibration**: does the model work as intended?

## (3) Fit the model to real data

## (4) *Criticize the inference*

- **chain mixing** (trace plots, posteriors by chain,  $\hat{R}$ )
- **basic diagnosis tools** (divergences, energy,  $n_{\text{eff}}$ )

## (5) *Criticize fitted model*

- **posterior predictive checking**

## (6) Application (inference, forecast...)

# Outline

Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

Fitting a simple SIR model

Simulation-based model assessment

The Bayesian workflow

**Scaling-up ODE-based models**

Extensions

# Scaling up ODE-based models

Inference with ODE-based models is **computationally intensive**

Be attentive to the model structure:

- HMC requires to compute the **gradient** of the log joint density multiple times for each iteration

$$\nabla_{\theta} \log \Pr(y, \theta)$$

- each block is treated differently
  - transformed data and generated quantities are evaluated once per iteration
  - parameters, transformed parameters and model are evaluated multiple times for each iteration
- put everything that **does not influence the inference** (e.g.  $\mathcal{R}_0$  or predicted values) in generated quantities



# Scaling up ODE-based models

Limiting the load of the ODE solver:

```
y = integrate_ode_rk45(sir, y0, t0, ts, theta, x_r, x_i);
```

- the computational cost of solving the ODEs scales with  $N + NK$ 
    - $N$ : the number of compartments
    - $K$ : the number of parameters in `y0` and `theta`
- remove unnecessary compartments (e.g.  $R(t)$ )
- reparametrize initial conditions

# Scaling up ODE-based models

Picking **the right ODE solver**:

- two are available:
    - `integrate_ode_rk45` uses the Runge-Kutta method (quicker but non-adapted to stiff systems)
    - `integrate_ode_bdf` uses the backward differentiation method (slower but adapted to stiff systems)
  - there is no formal definition of **stiffness**
  - intuitively, it occurs when the time step of the integrator needs to be very small to keep the solution stable (e.g. large variations in magnitude in time)
- start with `rk45`, move to `bdf` if there are problems (*folk theorem of statistical computing*)

# Scaling up ODE-based models

Tuning the ODE solver:

- additional options in the solver function

```
y_hat = integrate_ode_bdf(sho, y0, t0, ts, theta, x_r, x_i,  
                          rel_tol, abs_tol, max_steps);
```

- rel\_tol is the relative tolerance
- abs\_tol is the absolute tolerance
- max\_steps is the maximum number of steps
- can be adjusted depending on the level of **precision** needed

**⚠** be cause of the tolerance, the ODE solver may sometimes give negative values when too close to zero, causing issues

→ this can be solved by **adding 5-10 times the absolute tolerance** to the ODE output

# Outline

Introduction

Basics of Bayesian inference

Bayesian modelling with Stan

Fitting a simple SIR model

Simulation-based model assessment

The Bayesian workflow

Scaling-up ODE-based models

**Extensions**

## Extensions

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...)
- ...

# Extensions


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

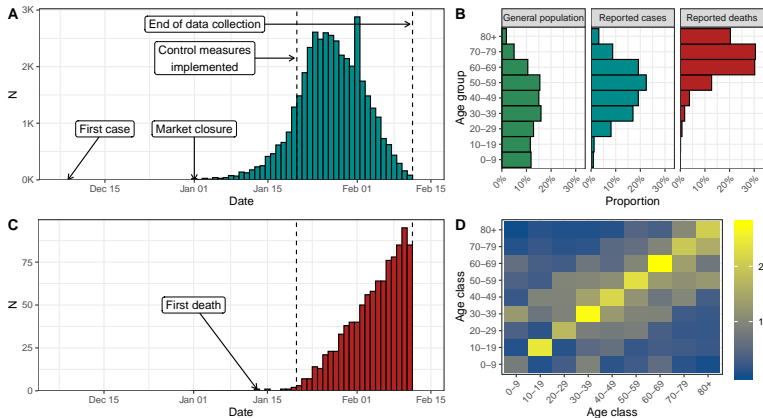
- (1) Simulate the dynamics of transmission and mortality of SARS-CoV-2 using publicly available surveillance data
- (2) Provide **overall and age-stratified estimates of IFR** for SARS-CoV-2 infection adjusted for right-censoring and preferential ascertainment

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):



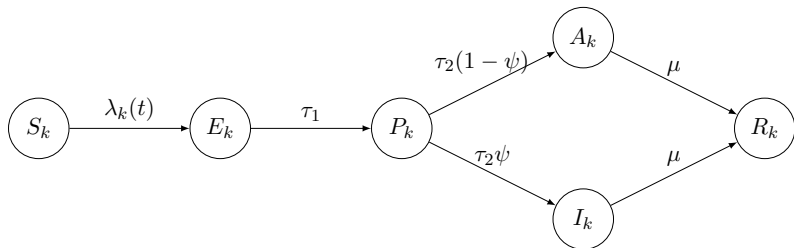
# Model development

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  $20.2 \pm 11.6$  days (fit to mortality data)

# Model development

Final model:



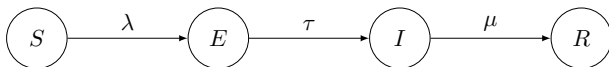
$S_k$	Susceptible (for age group $k$ )
$E_k$	Exposed
$P_k$	Presymptomatic
$A_k$	Infected asymptomatic
$I_k$	Infected symptomatic
$R_k$	Removed

$\lambda_k(t)$	Force of infection (time-dependent)
$1/\tau_1 + 1/\tau_2$	Incubation period (split in two)
$\psi$	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



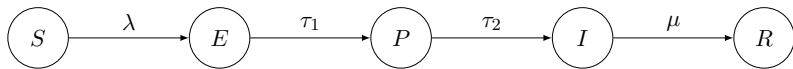
- $\tau$  is the inverse of the incubation period
- **individuals are infectious from symptom onset** (when entering  $I$ )

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

# 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  $\tau_1$  and  $\tau_2$
- individuals are infectious **before symptom onset** (entering  $P$ )

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

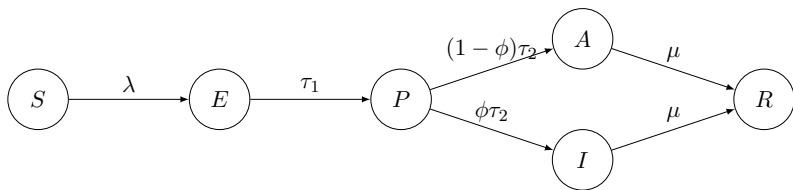
- we can introduce **reduced transmissibility** before symptom onset

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

# Model development

Asymptomatic infections:

- **SEPIAR**: adding a compartment **A** for asymptomatic



- we introduce the proportion of symptomatics  $\phi$
- we can introduce **reduced transmissibility** for asymptomatics

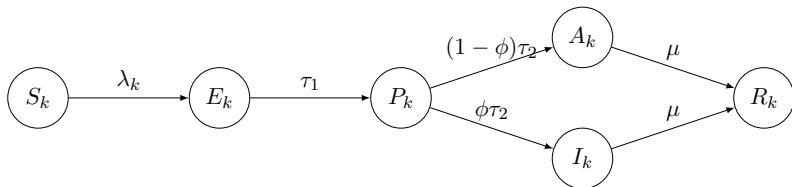
$$\lambda = \beta \frac{\kappa P + I + \kappa A}{N}$$

# 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  $k \in \{1, \dots, 9\}$  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  $\beta$  can be split in a **contact rate**  $c$  times a **probability of transmission upon contact**  $\beta$ , so that

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

**⚠** the same symbol  $\beta$  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**

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

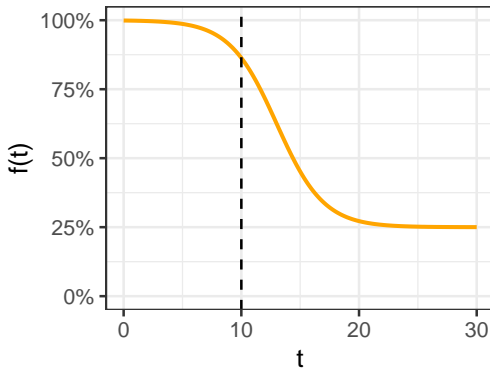
- to model the effect of control measures, we want a downward function that maps onto the interval  $[0, 1]$ , e.g. a **logistic function**:

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

- $\eta$  is the **relative reduction in transmission** after control measures
- $\xi$  is the **slope** of implementation of the control measures
- $\nu$  is the **delay** until the control measures are 50% effective (in days after  $t_c$ , the date of introduction of control measures).

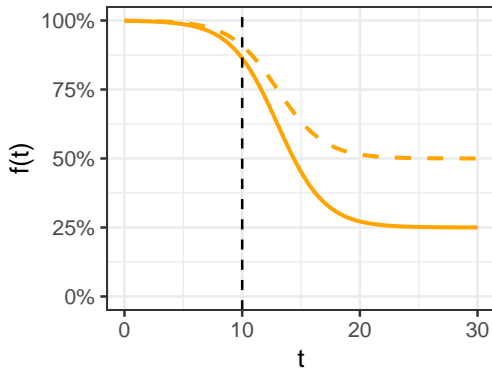
# Model development

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



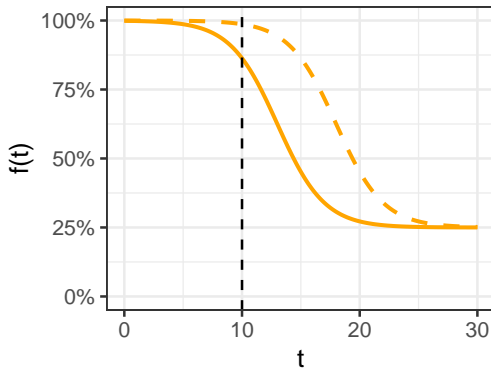
# Model development

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



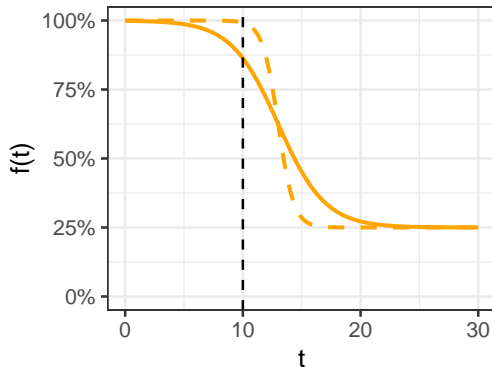
# Model development

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



# Model development

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



# Model development

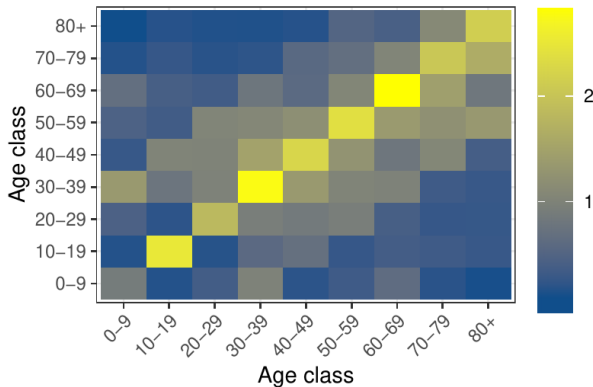
- we account for **behaviour differences across age groups**:

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

- 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

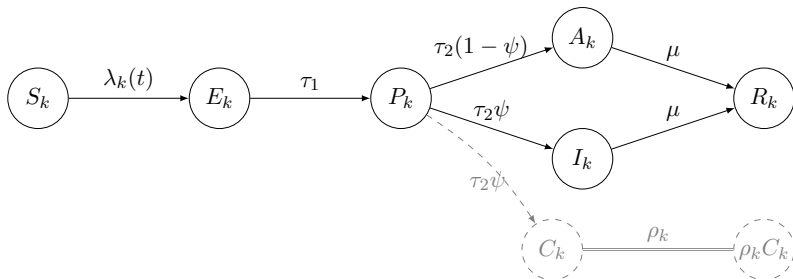




# Model development

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  $\rho_k$ :

$$A_t = \sum_k^9 \rho_k \Delta C_{k,t}^I$$

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

$$B_k = \frac{\rho_k C_k^I(t_{\max})}{\sum_k^9 \rho_k C_k^I(t_{\max})}$$

# Model development

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

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

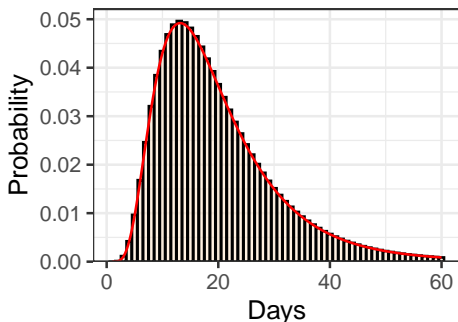
- $B_k$  can be mapped to the **age distribution of reported cases**  $\mathbb{B}$  using a multinomial likelihood:

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

# Model development

## Model fit to deaths:

- mortality is considered outside of the system of ODEs, using an age-specific **mortality parameter**  $\varepsilon_k$  (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  $t$  ( $1 \leq t \leq t_{\max} + 60$ ) 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  $t_{\max}$ :

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

## 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:

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

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  $\rho$** 
  - fix  $\rho_9$  (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  $\rho_9$ ...)
  - sensitivity analyses

# Inference

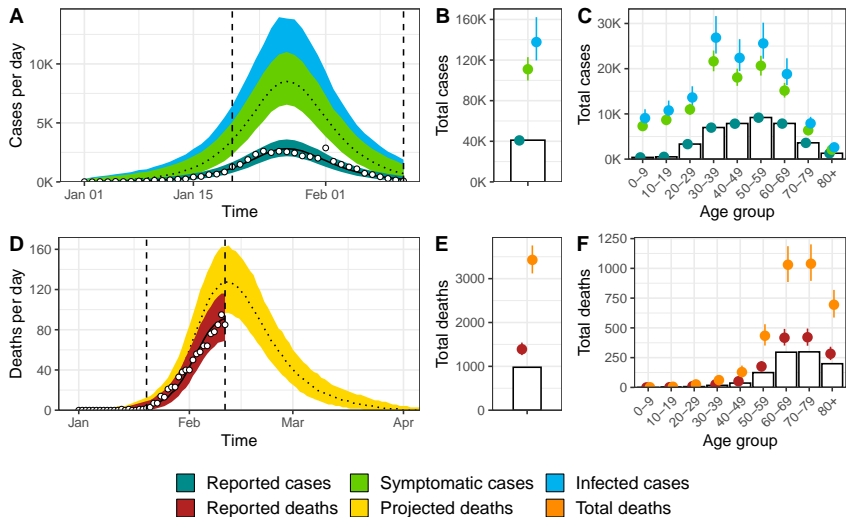
You know the drill:

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



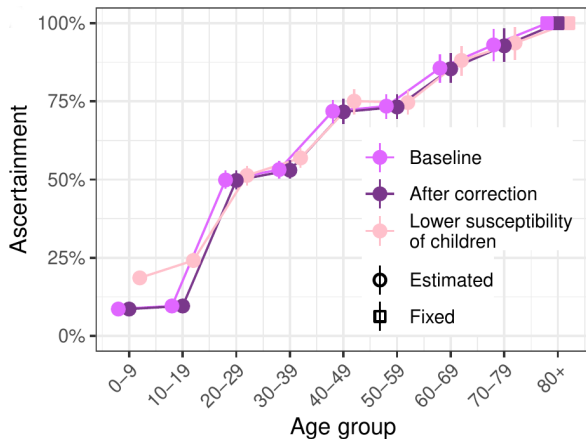
# Results in Hubei

## Posterior predictive check (Hubei):



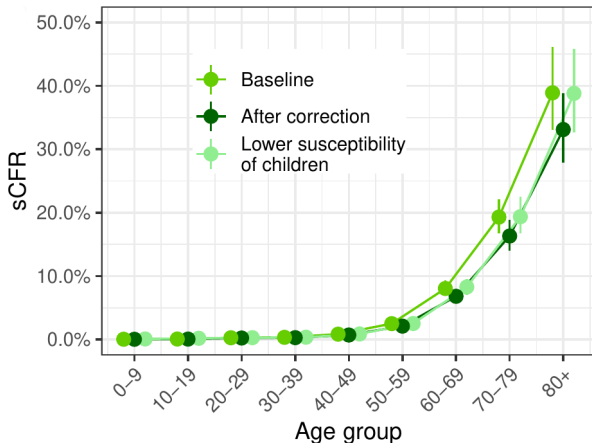
# Results in Hubei

Ascertainment (posteriors of  $\rho_k$ ):



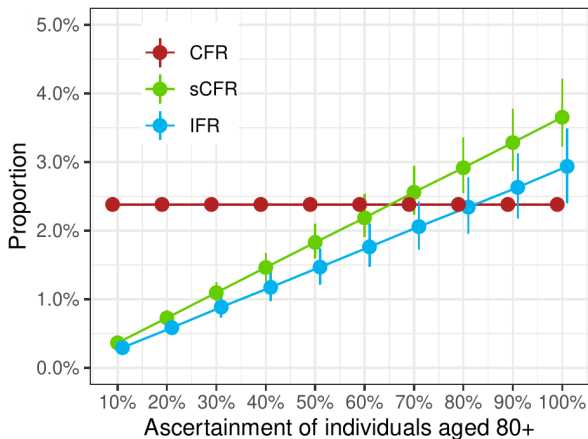
# Results in Hubei

Mortality among symptomatics or sCFR (posteriors of  $\varepsilon_k$ ):



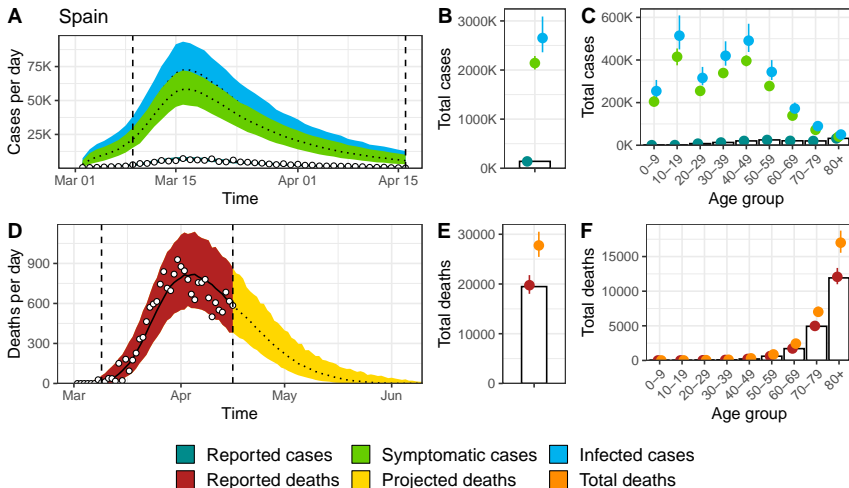
# Results in Hubei

Effect on the assumption on  $\rho_9$  on IFR estimate:



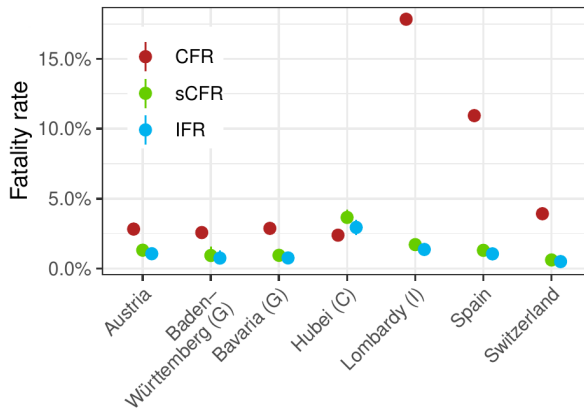
# Results in all regions

Posterior predictive check (Spain):



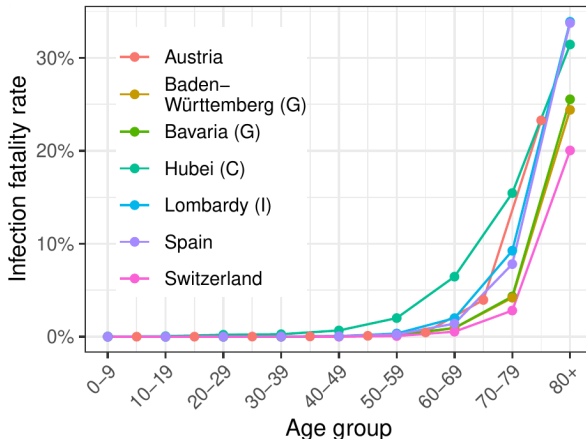
# Results in all regions

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**



# Conclusions

## General comments:

- knowing the **data-generating mechanisms** to avoid misinterpretation
- this knowledge can be used to build a model to adjust for known biases
- estimates of IFR obtained **early in the epidemic**, mostly confirmed in later seroprevalence studies

# Acknowledgements & ressources

## General Bayesian inference & Stan:

- Andrew Gelman et al. *Bayesian Data Analysis*  
<http://www.stat.columbia.edu/~gelman/book/>
- Michael Betancourt, *Introduction to Stan*  
[https://betanalpha.github.io/assets/case\\_studies/stan\\_intro.html](https://betanalpha.github.io/assets/case_studies/stan_intro.html)
- Andrew Gelman et al., *Bayesian workflow*  
<https://arxiv.org/abs/2011.01808>
- Jonah Gabry et al., *Visualization in Bayesian workflow*  
<https://arxiv.org/abs/1709.01449>
- Sean Talts et al., *Simulation-based calibration*  
<https://arxiv.org/abs/1804.06788>

# Acknowledgements & ressources

## Stan and ODEs or infectious disease models:

- Daniel Lee, *ODEs in Stan*  
[https://youtu.be/hJ34\\_xJhYeY](https://youtu.be/hJ34_xJhYeY)
- Chatzilena et al., *Contemporary statistical inference for ID models*  
<https://www.sciencedirect.com/science/article/pii/S1755436519300325>
- Bob Carpenter, *Population dynamics: the Lotka-Volterra model*  
<https://mc-stan.org/users/documentation/case-studies/lotka-volterra-predator-prey.html>
- Grinsztajn et al., *Bayesian workflow for disease transmission modeling in Stan*  
<https://arxiv.org/abs/2006.02985>