BayesianFitForecast: A User-Friendly R Toolbox for Parameter Estimation and Forecasting with Ordinary Differential Equations

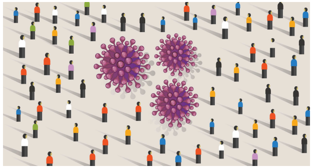
Gerardo Chowell

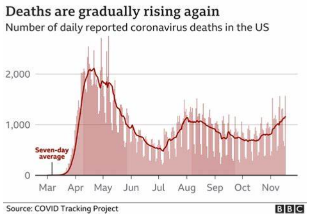
A joint work with Hamed Karami,Amanda Bleichrodt, and Ruiyan Luo
Department of Mathematics and Statistics
Department of Population Heath Sciences
Georgia State University

July 18, 2025

Background

- The use of mathematical models to create forecasts on the possible paths of epidemics and pandemics in almost real-time to inform public health interventions has gained more attention in the past ten years
- For example, we can point out the US CDC Flu Sight Challenge, the DARPA Chikungunya Challenge, the Dengue Forecasting Challenge, and the Ebola Forecasting Challenge along with epidemic and pandemic emergencies including the 2014–16 West African Ebola epidemic, the 2018–19 DRC Ebola epidemic, and the ongoing COVID-19 pandemic.




Deaths are gradually rising again
Number of daily reported coronavirus deaths in the US
2,000
1,000
0
Seven-day average
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Source: COVID Tracking Project
BBC

Bayesian toolbox and Stan

- Stan is a probabilistic programming language used for statistical modeling and inference, especially for Bayesian analysis.
- While working with Bayesian framework, coding a Stan file is necessary to define the model, ensuring that priors, likelihoods, and parameters are explicitly structured.
- `"BayesianFitForecast"` is a toolbox that automatically generates the Stan file based on the option file defined by the user.
- It also produces figures such as fit and forecasting plots, histograms of parameter estimates, and Excel files containing performance metrics, parameter estimates, and convergence diagnostics.

Frequentist and Bayesian analysis

- In the Frequentist method (the QuantDiffForecast), our treatment of parameter estimation assumes that θ is an unknown but non-random quantity. It is some fixed parameter describing the true distribution of data, and our goal was to determine this parameter.
- The Bayesian paradigm naturally incorporates our prior belief about the unknown parameter θ, and updates this belief based on observed data.
- In Bayesian analysis, before data is observed, the unknown parameter is modeled as a random variable θ having a probability distribution Θ, called the prior distribution. This distribution represents our prior belief about the value of this parameter.
- The conditional distribution of Θ given the observed data is called the posterior distribution. It represents our knowledge about the parameter Θ after having observed the data.

Bayes' rule

- Inference reverse-engineers process the data and aim to estimate parameter values given the observations. In a Bayesian framework, the set of plausible parameter values conditional on the data is characterized by the posterior distribution. The posterior distribution combines information from the data and prior knowledge and the result is obtained via Bayes' rule.
- So, based on the Bayes' rule, we have

$$p(\theta|X) \propto p(\theta)p(X|\theta).$$

Or, we can simply say

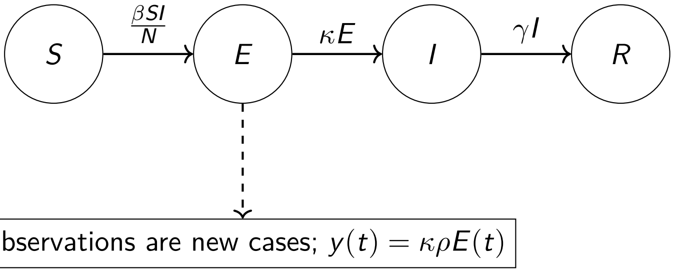
$$\text{Posterior density} \propto \text{Prior density} \times \text{Likelihood}.$$

Algorithms

MCMC and HMC

- Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain, one can obtain a sample of the desired distribution by recording states from the chain. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution.
- The Hamiltonian Monte Carlo (HMC) algorithm (originally known as hybrid Monte Carlo) is a Markov chain Monte Carlo method for obtaining a sequence of random samples that converge to being distributed according to a target probability distribution for which direct sampling is difficult. This sequence can be used to estimate integrals concerning the target distribution (expected values).

SEIR Model



$$\begin{tikzcd} S \arrow[r, "\frac{\beta S I}{N}"] & E \arrow[r, "\kappa E"] \arrow[d, dashed] & I \arrow[r, "\gamma I"] & R \\ & \boxed{\text{Observations are new cases; } y(t) = \kappa\rho E(t)} & & \end{tikzcd}$$

Figure: Compartmental diagram of the SEIR model with underreporting. Circles show the epidemiological compartments: susceptible (S), exposed (E), infectious (I), and recovered (R). Solid arrows indicate the transitions between compartments. The dashed arrow indicates the source of the observations, which are the newly reported infected individuals.

SEIR Model

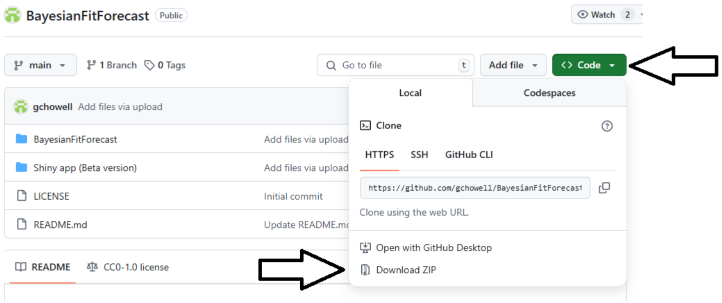
Consider the following model,

$$
\begin{aligned}
\frac{dS}{dt} &= -\frac{\beta SI}{N} \\
\frac{dE}{dt} &= \frac{\beta SI}{N} - \kappa E \\
\frac{dI}{dt} &= \kappa E - \gamma I \\
\frac{dR}{dt} &= \gamma I \\
\frac{dC}{dt} &= \kappa \rho E
\end{aligned}
$$

In this model, β is the transmission rate, κ is the incubation rate, γ is the recovery rate, and ρ is the reporting proportion.

Downloading the Toolbox

- Google `BayesianFitForecast GitHub`.
- Click on the link to the GitHub repository.
- On the repository page, click the **Code** button.
- Select **Download ZIP** from the dropdown.



https://github.com/gchowell/BayesianFitForecast

Clone using the web URL.

Installation and Setup

- Install R (recommended version 4.5) and RStudio.
- Ensure both Rtools and Java are installed and up to date.
- Open RStudio and start a new R session.
- Install required packages: `bayesplot`, `readxl`, `xlsx`, `openxlsx`, `rstan`, `ggplot2`, `loo`, `stringr`, `gridExtra`.
- **Note:** Compatible with R versions **prior to 4.5** and Stan versions **prior to 2.36**.

The `prior_sltn.R` Function

Purpose

Generates prior predictive simulations by sampling from user-defined prior distributions.

Workflow

1. Draw samples from prior distributions
2. Solve the ODE system using each sample
3. Construct an ensemble of trajectories to reflect **prior uncertainty**
4. To use this function, you must define the ODE model—including parameters and state variables, and specify the prior distributions for the parameters within the options file. The structure and setup of this file will be explained in detail later.

Example: Strong Prior Distribution Using a Simulated Data

- In this example, we review the solution generated by sampled prior distribution using the SEIR model described earlier.
- We generate the simulated data using $\beta = 0.6$, $\gamma = 0.1$, $\kappa = 0.2$, $\rho = 0.9$, and $N = 550,000$.
- We use a strong normal prior distribution for the parameters with mean around the true values. Then, the solutions should be around the actual data.

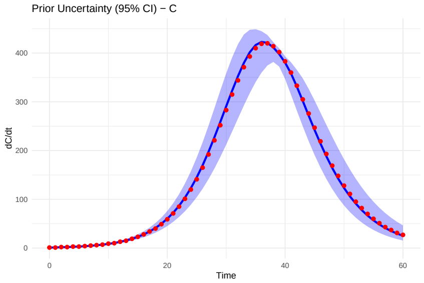
Prior Specification in R

```
params1_prior <- "normal(0.6, 0.01)T[0,]"  # beta: transmission rate
params2_prior <- "normal(0.1, 0.01)T[0,]"  # gamma: recovery rate
params3_prior <- "normal(0.2, 0.01)T[0,]"  # kappa: incubation rate
params4_prior <- "normal(0.9, 0.01)T[0,]"  # rho: reporting proportion
params5_prior <- 550000                    # N: fixed population size
```

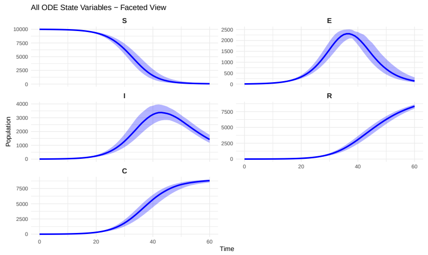
- Uses informative priors (low variance, $\sigma = 0.01$)

Prior Predictive Simulation

- Tight prior distributions yield low variability in trajectories
- Simulated trajectories represent our **initial beliefs** before observing data



(a) Incidence Cases



(b) All State Variables

Goal of the BayesianFitForecast Toolbox

Main Objective

Estimate the posterior distribution of model parameters that best explain the observed data.

What's Next

To achieve this, we must first fully explain the structure and role of the **options file**.

Next: Posterior Distribution

- We now want to combine prior knowledge with observed data
- Goal: obtain posterior distributions of parameters
- This requires the **options file** configuration

Option file

Reviewing all settings in one option file

Calibration Periods

Definition: The calibration periods are the specific time intervals used to fit the model to the data.

Input: Calibration periods should be entered as an array of numbers, where each number is selected from 1 to the total number of data points.

Important Note: Even if you want to consider only one calibration period, it should still be entered as an array with a single element.

Examples:

- **Multiple Calibration Periods:**

  ```
  calibrationperiods <- c(10, 15, 20, 25, 30)
  ```

  This indicates interest in fitting the model at several calibration periods: 10, 15, 20, 25, 30. Calibration period 10 means that we are fitting the first ten data points to the model.

- **Single Calibration Period:**

  ```
  calibrationperiods <- c(10)
  ```

Forecasting Horizon

Definition: The "forecastinghorizon" defines how far into the future predictions are made after the model is fit. It impacts the accuracy and uncertainty of predictions.
Flexibility: By choosing a large forecasting horizon initially, you gain the flexibility to generate metrics for a smaller time frame later.
Adjusting the Forecasting Horizon:

- **Example:** If you set

  ```
  forecastinghorizon <- 10
  ```

  when running "run_MCMC", you can later generate results for any horizon smaller equal than 10, such as $1, 2, \cdots, 10$. This allows you to focus on a specific part of the forecast while obtaining updated metrics. The reason to mention this point is that our toolbox is consisting of running two codes, and the first code takes much longer. So, having a large number for horizon makes our life easier for any smaller horizon later.

Adjusting the Forecasting Horizon

- **Adjusting Horizon in Post-Processing:** To zoom into a smaller segment of the forecast, change the

  `forecastinghorizon`

  in the options file after running `"run_MCMC.R"` but before executing `"run_analyzeResults.R"`.
- **Note:** Results may differ if you initially set

  `forecastinghorizon <- 5`

  However, if the chains in a larger one are convergent, this difference is tiny.
- **Important:** The `forecastinghorizon` must always be a positive integer. Zero forecastinghorizon means fitting.

Model Name

Definition: The `model_name` parameter allows the user to assign a name to the project, helping avoid confusion when extracting the results.
Usage: Assigning distinct names to projects, especially when settings are similar, prevents overwriting results in the same folder. This is particularly useful when running models with different priors or parameters.
Examples:

```
model_name <- "Bayesian-normalpriors"
model_name <- "Bayesian-niter=1000"
```

State Variables in the SEIR Model

Definition: The state variables in the SEIR model represent different compartments of the population and are defined as follows:

- `"S"`: Susceptible
- `"E"`: Exposed
- `"I"`: Infected
- `"R"`: Recovered
- `"C"`: Cumulative number of infected individuals

Variables Definition: The variables should be listed in the same order as their derivatives appear in the system of ODEs.

Example:

```
vars <- c("S", "E", "I", "R", "C")
```

Model Parameters

Definition: The parameters in the SEIR model represent values that affect the dynamics of the system. They can be either constants or values to be estimated during model fitting.

Parameters:

- `"beta"`: Transmission rate
- `"gamma"`: Recovery rate
- `"kappa"`: Incubation rate
- `"rho"`: Reporting proportion
- `"N"`: Total population size

Parameters Definition: All parameters must be included in the vector `params`, and each parameter should be defined as a string.

Example:

```
params <- c("beta", "gamma", "kappa", "rho", "N")
```

Time-Dependent Parameters

- Time-dependent parameters are defined in the `time_dependent_templates` object.
- Each is a string function of time `t` and elements from the `params` vector.

Example: Exponential Decay

```
time_dependent_param1 <- "
return (params1 * exp(-params2 * t));
"
```

Common Time-Dependent Forms

Piecewise Linear

```
time_dependent_param2 <- "
if (t < params1) {
return(params2);
} else if (t < params3) {
return(params2 + (params4 -
    params2) *
(t - params1) / (params3 - params1
    ));
} else {
return(params4);
}"
```

Mathematical Expression

$$\beta(t) = \begin{cases} \beta_0 & \text{if } t < t_1 \\ \beta_0 + (\beta_1 - \beta_0)\frac{t-t_1}{t_2-t_1} & \text{if } t_1 \leq t < t_2 \\ \beta_1 & \text{if } t \geq t_2 \end{cases}$$

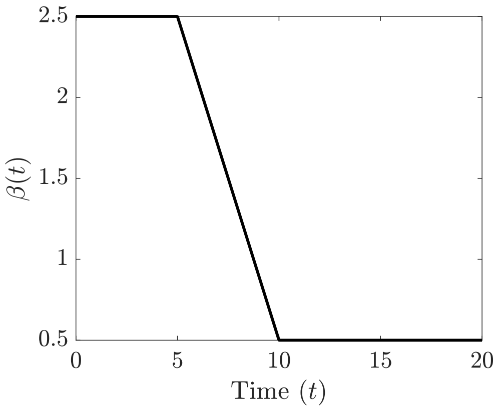


Figure: Piecewise linear function with the parameters $\beta_0 = 2.5$, $\beta_1 = 0.5$, $t_1 = 5$, and $t_2 = 10$.

Defining the ODE System

Overview: The `"ode_system"` defines the system of ODEs by specifying the relationships between parameters, variables, and their derivatives with respect to time.

Notation:

- `"paramsi"` refers to the `i`-th parameter in the `params` list.
- `"varsi"` refers to the `i`-th variable in the `vars` list.
- `"diff_vari"` refers to the derivative for the `i`-th variable.

Formatting Rules:

- The first line of the `ode_system` string should be left empty.
- The last line of the string must end with the final equation, followed by the closing quotation mark.

Defining the ODE System

Example: Assume the following definitions for variables and parameters:

```
vars <- c("S", "E", "I", "R", "C")
params <- c("beta", "gamma", "kappa", "rho", "N")
```

Then, the ODE system can be defined as:

```
ode_system <- '
diff_var1 = -params1 * vars3 * vars1 / params5
diff_var2 = params1 * vars3 * vars1 / params5 - params3 * vars2
diff_var3 = params3 * vars2 - params2 * vars3
diff_var4 = params2 * vars3
diff_var5 = params4 * params3 * vars2'
```

Fixing Parameters in the Model

Overview: Users may wish to keep certain parameters fixed. For example, the population size is often treated as a constant and does not require estimation.
Defining Fixed Parameters: In this toolbox, you can designate any parameter as constant using the `paramsfix` array.
Array Definition: - The `paramsfix` array should have the same length as the `params` array. - If an element in `paramsfix` is set to 1, the corresponding parameter in `params` is fixed (constant). - If it is set to 0, the parameter will be estimated.
Example:

- `params <- c("beta", "gamma", "kappa", "rho", "N")`
- `paramsfix <- c(1, 0, 0, 0, 1)`

In this example, β and N are constants, while γ, κ, and ρ will be estimated.

Composite Expressions

Definition: Sometimes, users need to estimate derived parameters, such as the basic reproduction number or recovery time. To facilitate this, users should include these parameters in the `composite_expressions` list. Each entry should pair the parameter name on the left with its corresponding formula on the right, expressed as a string and a function of the model's parameters.

Example: In the SEIR model, the parameters might be defined as follows:

```
composite_expressions <- list(
R0 = "beta/gamma",
recovery_time = "1/gamma"
)
```

fitting_index and fitting_diff

Definition: These parameters map the data to model variables. For example, in the SEIR model, if data represents the infectious individuals, set `fitting_index` to `c(3)`, where I is the 3rd variable in `vars`. For the cumulative cases in the SEIR model, set `fitting_index` to `c(5)` for C.

Derivatives: To fit data to a variable's derivative, set `fitting_diff` to `c(1)`. For instance, `fitting_index = c(5)` and `fitting_diff = c(1)` in SEIR fits data to $\frac{dC}{dt}$ (incidence cases), while `fitting_index = c(3)` and `fitting_diff = c(1)` fits to $\frac{dI}{dt}$ (newly infectious).

Multiple datasets: For multiple time-series, use arrays. For example, `fitting_index = c(3,5)` and `fitting_diff = c(0,1)` fit the data to I and $\frac{dC}{dt}$.

errstrc

Definition: The `errstrc` parameter is an integer that specifies the error structure used when fitting the model to data, which describes the variability or noise.

Options:

1. Negative binomial: accounts for overdispersion in count data.
2. Normal: assumes constant variance.
3. Poisson: assumes variance equals the mean.

Selection: Choose the error structure by setting `errstrc` to the corresponding number.

Additional Parameters:

- Negative binomial requires estimating dispersion ϕ.
- Normal requires estimating standard deviation σ.

cadfilename1 and caddisease

cadfilename1: This is a string representing the name of the Excel file (without the extension `.xlsx`). The Excel file should have at least two columns:

- First column: Sequential time points (e.g., 0, 1, 2, ...) representing days, weeks, or years.
- Second column: Temporal incidence data related to the state variables or their derivatives. The column name should start with `cases1`, followed by `cases2`, `cases3`, etc., for additional columns.

caddisease: This string defines the name of the disease or process. It is used to generate a unique name for the results folder, preventing overwriting results from different diseases.

series_cases and datetype

series_cases: This string parameter defines the time series label(s) for the y-axis. For example:

- If set to `"Cases"`, the y-axis label will be `"Cases"` for a single time series.
- If set to `"infected", "recovered"`, there are two time series, and the y-axis labels will be `"infected"` for cases1 and `"recovered"` for cases2.

datetype: This parameter specifies the x-axis label. For instance, setting `"datetype"` to `"Days"` labels the x-axis as `"Days"`.

Priors - Definition and Considerations

Definition: This section allows users to specify prior distributions for parameters. If a parameter is fixed (indicated by `"paramsfix"` being 1), its prior is a constant value. For estimable parameters, enter a distribution as a string.
Parameter Considerations: Pay attention to the range of parameters. For instance, parameters like β, γ, and κ are positive. When using normal distributions, truncate at 0 (e.g., $T[0,])$.

Priors - Example

Example: Given:

- `params <- c("beta", "gamma", "kappa", "rho","N")`

Prior beliefs:

1. `beta`: 2 (constant)
2. `gamma`, `kappa`: `"normal(1,2)T[0,]"` (truncated normal)
3. `rho`: `"uniform(0,1)"` (uniform)
4. `N`: 10000 (constant)

The prior distribution setup:

- `params1_prior <- 2`
- `params2_prior <- "normal(1,2)T[0,]"`
- `params3_prior <- "normal(1,2)T[0,]"`
- `params4_prior <- "uniform(0,1)"`
- `params5_prior <- 10000`

Lower and Upper Bounds - Definition

Definition: This section allows users to define lower and upper bounds for parameters, restricting the range of values during estimation.
Bounds:

- `"LB"`: Minimum value a parameter can take.
- `"UB"`: Maximum value a parameter can take.
- Use `"NA"` for no restrictions.

Lower and Upper Bounds - Example

Example: Given:

- `params <- c("beta", "gamma", "kappa", "rho","N")`

Parameter bounds:

- `beta`, `gamma`, `kappa`: Lower bounds 0, no upper bounds.
- `rho`: Lower bound 0, upper bound 1.
- `N`: No lower or upper bounds.

Bound definitions:

- `params1_LB <- 0`, `params1_UB <- NA`
- `params2_LB <- 0`, `params2_UB <- NA`
- `params3_LB <- 0`, `params3_UB <- NA`
- `params4_LB <- 0`, `params4_UB <- 1`
- `params5_LB <- NA`, `params5_UB <- NA`

Prior Distribution for Additional Parameters in Error Structure

Additional Parameters: For the negative binomial and normal error structures, we define:

- ϕ for the negative binomial.
- σ for the normal distribution.

Specifying Priors: The prior for the parameter σ is defined using:

- `normalerror1_prior <- "cauchy(0, 2.5)T[0,]"`

(Cauchy distribution with location 0 and scale 2.5).

Prior Distribution for Additional Parameters in Error Structure

Multiple Datasets: For three time series, define:

- `normalerror1_prior <- "cauchy(0, 2.5)T[0,]"`
- `normalerror2_prior <- "cauchy(0, 2.5)T[0,]"`
- `normalerror3_prior <- "cauchy(0, 2.5)T[0,]"`

Error Structure Setting:

- If `errstrc` $= 1$, define only `negbinerror_prior`.
- If `errstrc` $= 2$, define only `normalerror_prior`.

Initial Conditions (Ic)

Definition: The vector `Ic` represents the initial conditions for a Bayesian method in Stan, corresponding to initial state variable values. The order must match `vars`.

Example (SEIR Model):

$$S(0) = N - i_0, \quad E(0) = 0, \quad I(0) = i_0, \quad R(0) = 0, \quad C(0) = i_0$$

where i_0 is the first observed data point and N is the population size.

If $i_0 = 1$:

$$Ic = c(params5\_prior - 1, 0, 1, 0, 1)$$

Alternatively, specify N directly:

$$Ic = c(10000 - 1, 0, 1, 0, 1)$$

`vars.init`

Purpose: Sometimes, the initial conditions (e.g., initial number of infected or exposed individuals) are unknown and need to be estimated. This can be done by setting `vars.init = 0`. If the values are known, set `vars.init = 1`.

When `vars.init = 0`:

- `params`: Add the new parameter (e.g., i_0) to `params`.
- `paramsfix`: Update to include the new parameter, with its entry set to 0 (for estimation).
- `priors`: Define the prior distribution for the new parameter.
- `Ic`: Specify initial conditions. Example for SEIR:

$$\texttt{Ic = ("N-}i_0\texttt{", 0,"}i_0\texttt{", 0, "}i_0\texttt{")}$$

When `vars.init = 1`: No changes are needed.

niter and num_chain

Definition:

- niter: This specifies the number of iterations for MCMC sampling.
- num_chain: This sets the number of MCMC chains.

Importance: Both parameters are crucial in Bayesian analysis using Stan to ensure convergence of the MCMC algorithm.
Example: To set the values for 2000 iterations and 4 chains:

```
niter = 2000,    num_chain = 4
```

These values can be adjusted based on the complexity of the model or specific needs for ensuring convergence. The convergence occurs when the parameter "Rhat" is less than 1.1.

Process

What is the process after configuring the option file?

run_MCMC

Purpose: The file run_MCMC is designed to fit the model to the data using user-defined parameters.

Configuration:

- The user must update the options in the options file as needed.
- The user must also ensure that line 5 in the run_MCMC file matches the name of the configured options file.

Importance: Updating the options and ensuring correct file references is crucial for running the MCMC fitting process smoothly.

Saved Data

Output: After running run_MCMC, the generated samples will be saved as an .Rdata file in the toolbox folder.
File Naming: For each period in calibrationperiods, a corresponding .Rdata file is generated. The filename is based on:

- model_name
- calibrationperiod
- forecastinghorizon
- errstrc
- caddisease

Recommendation: If two option files share identical settings, the second run will overwrite the first file. It is recommended to modify model_name to avoid this.

run_MCMC_Parallel

- The file run_MCMC_Parallel.R is designed to accelerate MCMC computation by running multiple chains in parallel.
- Within the R environment, the rstan package utilizes R's built-in parallel package to run chains concurrently.
- Users can set the number of cores via:

```
options(mc.cores = parallel::detectCores())
```

- This command automatically detects the maximum number of available cores and assigns one core per chain.
- Significantly reduces computation time.
- Helps MCMC chains converge faster by running them independently and simultaneously.

run_analyzeResults

Purpose: By running run_analyzeResults, users can generate and analyze results based on the previously configured options.
Key Points: Ensure the option file remains the same as in run_MCMC (except for the forecasting horizon). Modify line 10 to specify the option file name.
File Naming: Generated files will follow the format:

```
model_name + caddisease + errstrc +
calibrationperiod + forecastinghorizon
```

Convergence and Forecast Files

Convergence File: Includes the convergence parameter `Rhat`. Use results with `Rhat < 1.1`. To achieve this, increase `niter` or `num_chain`.
Example format:
`convergence-Bayesian-weak-sanfrancisco-normal-cal-10-fcst-10`
Forecast File: Includes columns for Date, Data, Median, Lower/Upper Bounds of 95% PIs. Example format:
`forecast-Bayesian-weak-sanfrancisco-normal-cal-10-fcst-10`

Parameters and Performance Metrics

Parameters File: Provides estimates with median, mean, lower/upper bounds of 95% CIs. For normal error structure: σ, For negative binomial: ϕ.
Example format:
`parameters-Bayesian-weak-sanfrancisco-normal-cal-10-fcst-10`
Performance Metrics File: Includes MAE, MSE, WIS, and Coverage for calibration/forecasting periods. Example format: `performance metrics-Bayesian-weak-sanfrancisco-normal-cal-10-fcst-10`

Generated PDF Files

In addition to Excel files, several PDF files will be generated:

- **Histograms**: Save histograms of parameters, including σ or ϕ. Example: `"beta-histogram-Bayesian-weak-\sanfrancisco-normal-cal-10-fcst-10"`.
- **Forecast**: Shows median fit and 95% PI margins. Example: `"Forecast-Bayesian-weak-sanfrancisco-normal-\cal-10-fcst-10"`.
- **Trace plot**: Visualizes parameter values over MCMC iterations. Example: `"traceplot-Bayesian-weak-sanfrancisco-normal-cal-10-fcst-10"`.

Example 1

Overview

In this example, the SEIR model is applied to the SF 1918 flu dataset with the parameter $\rho = 1$. The model is fitted using a Poisson error structure, and the options file is modified to achieve the desired results.

How to start

We begin by creating a new file `"options_SEIR_sanfrancisco_Ex1.R"` based on the original `"options_SEIR_sanfrancisco_normalprior"` file.

Key parameters in the new file

```
calibrationperiods <- c(17)
forecastinghorizon <- 10
vars <- c("S", "E", "I", "R", "C")
ode_system <- '
 diff_var1 = -params1 * vars3 * vars1 / params5
 diff_var2 = params1 * vars3 * vars1 / params5 - params3 * vars2
 diff_var3 = params3 * vars2 - params2 * vars3
 diff_var4 = params2 * vars3
 diff_var5 = params4 * params3 * vars2'
paramsfix <- c(0,1,1,1,1)
```

Settings and execution

Important Settings

Ensure the following settings are correct for fitting and error structure:

- `fitting_index = 5`
- `fitting_diff = 1`
- `errstrc = 3`

Execution

To run the MCMC algorithm:

- Modify line 5 in `run_MCMC` to use the new options file: `"options_SEIR_sanfrancisco_Ex1.R"`.
- Execute the script in RStudio and analyze results using `run_analyzeResults.R`.

Fitting using Poisson error structure

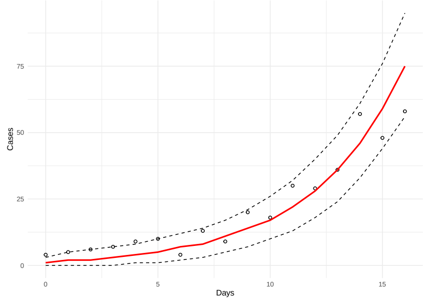
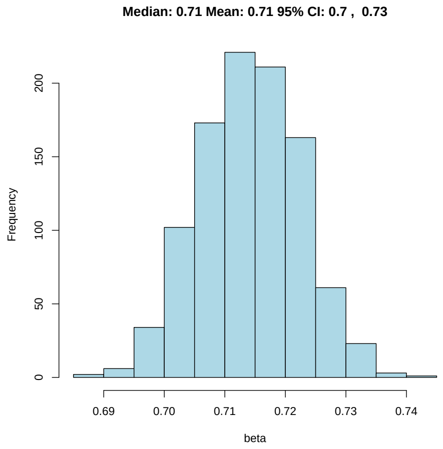
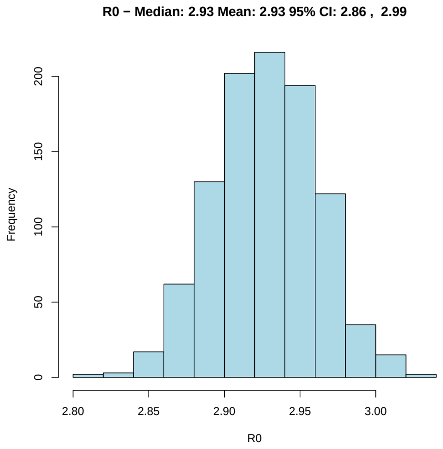


Figure: Bayesian fitting of the SEIR model to the 1918 influenza dataset from San Francisco was performed using a Poisson error structure, calibrated over 17 days. The model captures the dynamics of newly infected individuals, represented by $\frac{dC}{dt}$. The parameters $\kappa = \frac{1}{1.9}$, $\gamma = \frac{1}{1.4}$, and $\rho = 1$ are held constant, while the prior distribution for β is uniform(0,10). All parameters were constrained to have non-negative values, with a total population size fixed at $550,000$. Initial conditions were set as (549996, 0, 4, 0, 4), based on the first recorded data point. The MCMC algorithm was executed with 1,000 iterations across two independent chains to ensure robust convergence.

Histogram of posterior distributions: Poisson error structure





Posterior inference and convergence analysis: Poisson error structure

Table: Posterior inference and convergence analysis for the example applying the SEIR model to the San Francisco 1918 influenza dataset with a Poisson error structure.

Calibration	**Parameter**	**Mean**	**Median**	**CI_95**	**N_eff**	**Rhat**
17	β	0.71	0.71	(0.7 , 0.73)	316.35	1

Performance metrics: Poisson error structure

Table: The performance metrics resulting from Bayesian fitting of the SEIR model to the 1918 influenza dataset from San Francisco, using a Poisson error structure and 17 days of calibration data.

	Calibration	**Forecasting**
MAE	5.24	NA
MSE	45.35	NA
WIS	3.33	NA
Coverage of 95% CI	88.24	NA

Example 2

This example compares the performance metrics of the SEIR model using the SF 1918 flu dataset with two error structures: negative binomial and Poisson. The top panel shows the model fit, while the bottom panels display histograms of the estimated parameters: β, ϕ, and R_0.
Additionally, the performance metrics and convergence analysis are presented in tables for a detailed comparison of the two error structures. The option file for this example can be found under the name `"options_SEIR_sanfrancisco_Ex2"`.

Fitting using negative binomial error structure

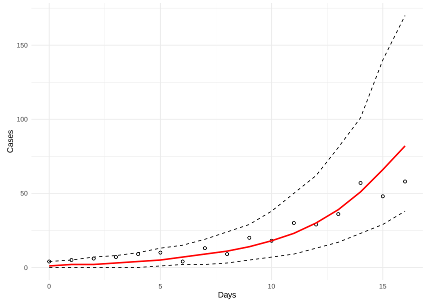
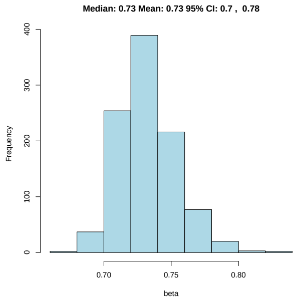
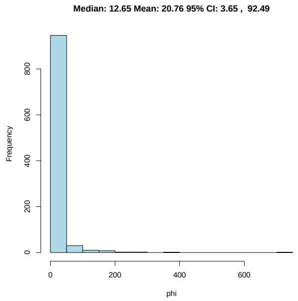
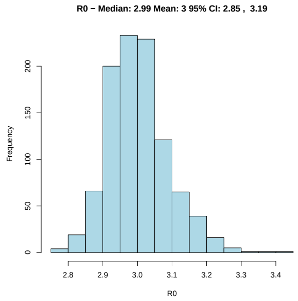


Figure: The Bayesian fitting of the SEIR model to the first 17 days of the 1918 influenza pandemic in San Francisco was performed using a negative binomial error structure. The model is fitted to the newly infected individuals, represented by $\frac{dC}{dt}$. The parameters $\kappa = \frac{1}{1.9}$, $\gamma = \frac{1}{1.4}$, and $\rho = 1$ are constants, while the prior distribution for β is uniform(0,10) and for ϕ is exponential(5), with all parameters having a lower bound of zero. The population size is 550,000, with an initial condition of (549996, 0, 4, 0, 4) based on the first recorded data point. The MCMC algorithm was run with 1,000 iterations across two chains.

Parameter histograms: negative binomial error structure







Performance metrics: negative binomial error structure

Table: Performance metrics for the SEIR model with a negative binomial error structure, based on the San Francisco 1918 influenza dataset, calibrated over 17 days.

	Calibration	**Forecasting**
MAE	5.76	NA
MSE	68.24	NA
WIS	3.57	NA
Coverage of 95% PI	100.00	NA

Convergence: negative binomial error structure

Table: Convergence and other statistics for the SEIR model applied to the San Francisco 1918 influenza dataset with a negative binomial error structure.

Parameter	Mean	Median	CI_95	N_eff	Rhat
β	0.73	0.73	(0.7 , 0.78)	356.83	1
ϕ	20.76	12.65	(3.65 , 92.49)	395.63	1

Comparison of performance metrics in calibrating

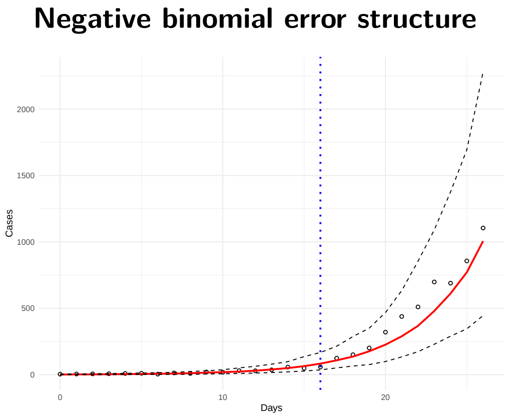
Table: Comparison of performance metrics between the Poisson and negative binomial error structures for the SEIR model, based on the SF 1918 influenza dataset.

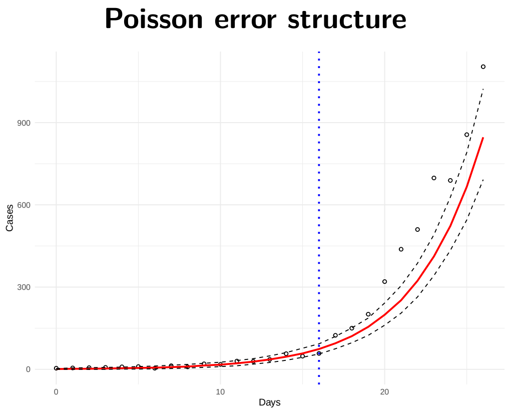
Model	**MAE**	**MSE**	**Coverage 95% PI**	**WIS**
Bayesian, Poisson error	5.24	45.35	88.24	3.33
Bayesian, Negative binomial error	5.76	68.24	100	3.57

Example 3: Forecasting with Poisson vs. negative binomial

In this example, we present the best model for forecasting 10 days after calibrating 17 days of the SF 1918 flu dataset. During the calibration period, the Poisson error structure performs better in terms of MAE, MSE, and WIS metrics, while the negative binomial model excels in 95% PI coverage. However, the negative binomial model demonstrates superior performance across all metrics in the forecasting period. The forecast for both Poisson and negative binomial error structures is presented, along with a comparison of their performance metrics. The related option files for this example are provided as `"options_SEIR_sanfrancisco_Ex3_Negbin"` and `"options_SEIR_sanfrancisco_Ex3_Poisson"`.

Forecast negative binomial vs. Poisson





Comparison of performance metrics in forecasting

Table: A comparison of forecasting and model fit performance metrics between the Poisson and negative binomial error structures.

Error structure	MAE	MSE	Coverage 95% PI	WIS
Calibration performance				
Negative binomial	5.85	68.25	100.0	3.62
Poisson	5.00	41.71	88.24	3.28
Forecasting performance				
Negative binomial	92.10	12325.90	100.0	56.84
Poisson	149.75	29976.22	10.0	122.28

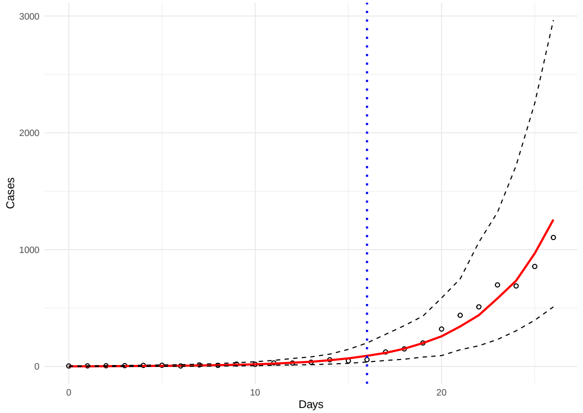
Example 4: Comparison of SEIR and exponential growth models

In this practice example, we compare the SEIR model to the exponential growth model using the SF 1918 flu data, with a negative binomial error structure. The models were calibrated for 17 days, followed by a forecast of 10 days.

The forecasting results, along with the histogram of the parameter r from the exponential growth model, are presented.

The results indicate that the SEIR model outperforms the exponential growth model in calibration performance across all metrics. However, the exponential growth model demonstrates superior performance in forecasting for all metrics.

Forecast using exponential growth model



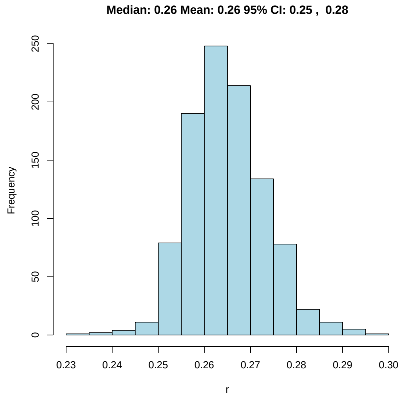


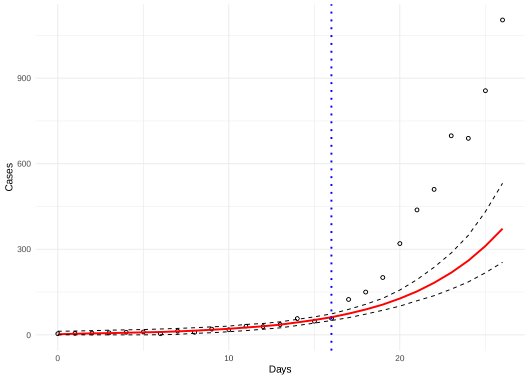
Table: A comparison of performance metrics between the SEIR model and the exponential growth model using Bayesian fitting to the San Francisco 1918 flu dataset with negative binomial error structure, using 17 days of calibration data with a forecasting horizon 10.

Model	MAE	MSE	Coverage 95% PI	WIS
Calibration Performance				
SEIR model	5.85	68.25	100.0	3.62
Exponential growth	6.53	104.88	100.0	4.11
Forecasting Performance				
SEIR model	92.10	12325.90	100.0	56.84
Exponential growth	66.50	6939.35	100.0	53.98

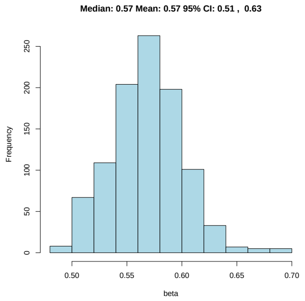
Example 5: Forecasting the 1918 flu pandemic in San Francisco estimating the initial number of infected people

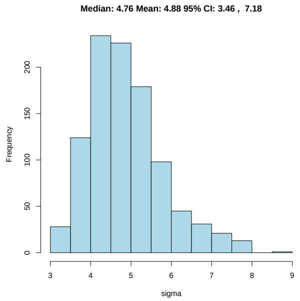
In this example, we utilize data from the 1918 influenza pandemic in San Francisco to forecast the number of infected individuals using a normal error structure. The model is calibrated over a 17-day period, followed by a 10-day forecast.
In this scenario, we estimate the parameter i_0, which represents the initial number of infected individuals. The estimation reveals that i_0 is approximately 11.
The results demonstrate the model's effectiveness in capturing the dynamics of the outbreak and providing insights into the initial spread of the infection.

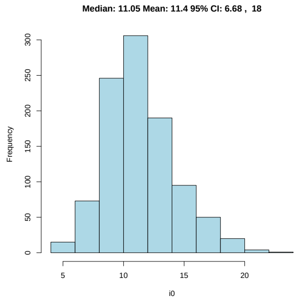
Forecast while estimating the initial number of infected people



Parameter histograms







Example 6: Simulation of SEIR using multiple datasets

In this example, we simulate data using the forward solution of the SEIR model with the following parameters:

- $\beta = 0.5$
- $\gamma = 0.25$
- $\kappa = 1$
- $\rho = 1$
- $N = 100{,}000$

Normal noise with a standard deviation of 5 is added to the simulation.
The simulated data includes:

- Day (Column 1)
- Number of infectious individuals I (Column 2)
- Derivative of recovered individuals $\frac{dR}{dt}$ (Column 3)
- Derivative of cumulative cases $\frac{dC}{dt}$ (Column 4)

Model Fitting Results and Analysis

Three modeling scenarios are considered:

1. Fitting the SEIR model to all data
2. Fitting the model to I and $\frac{dC}{dt}$
3. Fitting the model to only $\frac{dC}{dt}$

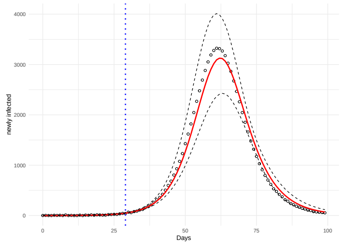
The calibration period spans 70 days, followed by a 30-day forecast. The fitted curves of the incidence, along with prediction bands for the three scenarios, highlight the following:

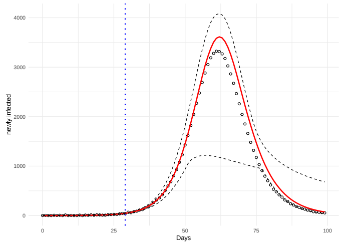
- Scenario 1, using all data, outperforms Scenario 2, which uses I and $\frac{dC}{dt}$.
- Scenario 2 performs better than Scenario 3, which relies solely on $\frac{dC}{dt}$.

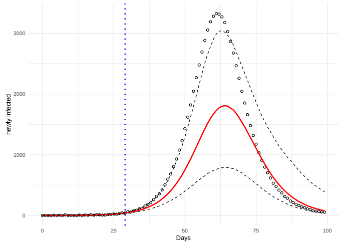
Notably, while Scenario 3 performs better during calibration, it underperforms in forecasting. This suggests:

- Overfitting during calibration due to limited data.
- Scenario 1 provides more accurate parameter estimates, improving forecasting performance.

Forecast using three scenarios







Performance metrics using three scenarios

Table: Performance metrics for the rate of new cases, $\frac{dC}{dt}$, obtained by fitting the SEIR model to simulated data generated using the forward solution of the SEIR model.

Scenario	Period	MAE	MSE	Coverage 95% PI	WIS
Calibration Performance					
(i) $I, \frac{dR}{dt}, \frac{dC}{dt}$	30	3.06	15.59	96.67	1.99
(ii) $I, \frac{dC}{dt}$	30	3.22	16.02	93.33	2.02
(iii) $\frac{dC}{dt}$	30	3.04	14.08	96.67	1.90
Forecasting Performance					
(i) $I, \frac{dR}{dt}, \frac{dC}{dt}$	70	100.98	17216.70	87.14	65.78
(ii) $I, \frac{dC}{dt}$	70	131.93	35634.15	84.29	93.13
(iii) $\frac{dC}{dt}$	70	431.25	464089.54	60.0	287.49

Example 7: Time-Dependent SEIR Model

Goal: Model interventions that change transmission over time

Time-dependent transmission rate:

$$\beta(t) = \begin{cases} \beta_0, & \text{if } t < t_{\text{int}} \\ \beta_1 + (\beta_0 - \beta_1)e^{-q(t - t_{\text{int}})}, & \text{if } t \geq t_{\text{int}} \end{cases} \tag{1}$$

- β_0: initial transmission rate
- β_1: final transmission rate
- t_{int}: intervention time
- q: transition rate

Simulated Data

Parameters used:

- $\beta_0 = 0.5$
- $\beta_1 = 0.1$
- $q = 0.1$
- $t_{\text{int}} = 10$
- $\gamma = 1/7$
- $\kappa = 1/5$
- $\rho = 0.6$
- $N = 1{,}000{,}000$

Time period: 100 days **Intervention:** 80% reduction in transmission after day 10

Dataset Structure

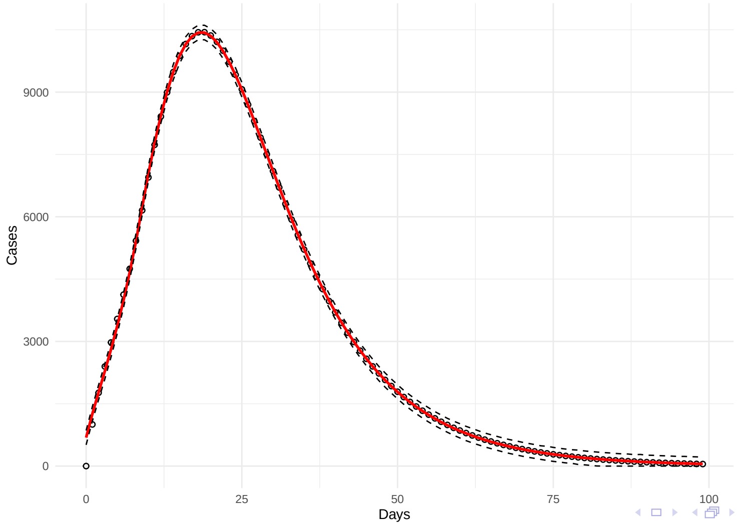
File: `seir_simulated.xlsx`

- Column 1: Days
- Column 2: $\frac{dC}{dt}$ (new cases)

Options file: `options_seir_timedep_Ex7.R`

Task: Estimate β_0, β_1, q, t_{int} from data

Model Fit Results



Parameter Estimation Results

Parameter	Mean	Median	CI_{95}	N_{eff}	$\hat{R}$
β_0	0.91	0.67	(0.55, 1.62)	26.08	1.01
β_1	0.34	0.06	(0.02, 1.20)	25.77	1.01
q	0.16	0.10	(0.10, 0.45)	33.43	1.00
κ	0.11	0.12	(0.08, 0.14)	30.95	1.01
γ	0.14	0.11	(0.07, 0.25)	28.97	1.00
ρ	0.45	0.48	(0.33, 0.55)	29.37	1.00

THANK YOU :)