Analysing the identifiability of the SEIRD model

Introduction

By fitting models of transmission dynamics to epidemiological data, we determine the combinations of model parameters which are most consistent with observations. Models of transmission dynamics, like the SEIRD model, are, however, highly nonlinear and may have many parameters. Because of these characteristics, many combinations of parameters may equally well explain the observed data. That is, the model may be unidentified given the observations at hand.

Here, we illustrate how early on during an epidemic, many parameter sets of the SEIRD model are consistent with the data (here, we assume that only cases and deaths are observed). Because of this, it is important to practice caution when making forecasts of the resultant path of the epidemic. In particular, it is essential that the uncertainty in the model fit be taken into account when making projections. Here, we illustrate this using the range of optimisation results obtained when fitting to the same data, but alternative approaches to this uncertainty quantification include Bayesian inference.

In this vignette, we begin by performing optimisations using real COVID-19 case and deaths data. Next, we use profile likelihood methods to formally determine identifiability of the SEIRD model using simulated data (i.e. data generated using known parameter values). Our approach illustrates the importance of using simulated data for assessing identification of epidemiological models.

Setting up environment with required libraries

```
library(comomodels)
library(ggplot2)
library(dplyr)
library(tidyr)
```

Choose between plotting cached data or run optimisation

```
# TRUE to plot saved data, FALSE to run optimisation
cached_bool <- TRUE
set.seed(0)
```

SEIRD model

In brief, the SEIRD model is a compartmental model that describes the transition of population groups (susceptibles, exposed, infectious, recovered and dead) between each other. The population groups are represented with S, E, I, R and D respectively. The ordinary differential equations (ODEs) that define the model are

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta SI$$
$$\frac{\mathrm{d}E}{\mathrm{d}t} = \beta SI - \kappa E$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \kappa E - (\gamma + \mu)I$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I$$
$$\frac{\mathrm{d}D}{\mathrm{d}t} = \mu I$$

where β, κ, γ and μ are the infection rate, incubation rate, recovery rate and death rate respectively. More details can be found in the SEIRD vignette from the `comomodels` package.

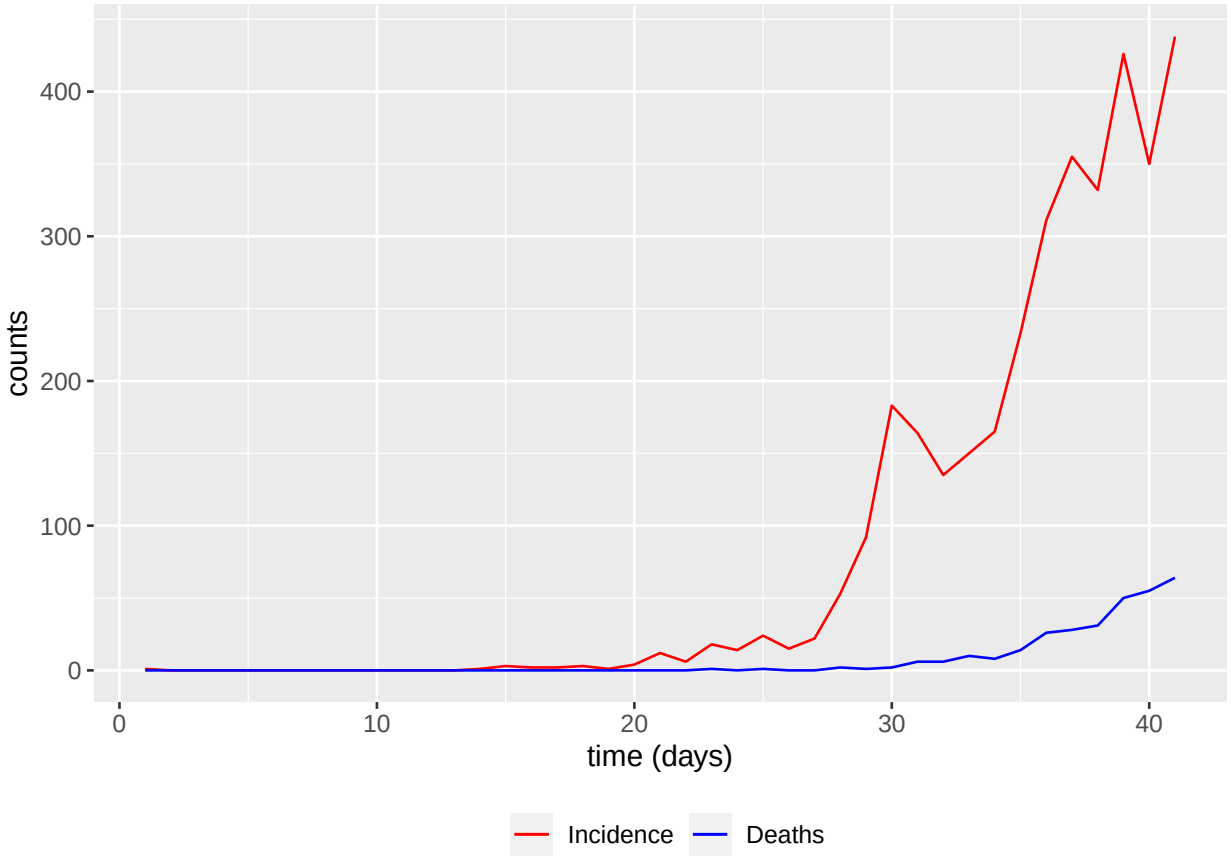
Optimisation on real COVID-19 cases and deaths data for London

We begin by fitting the SEIRD model to real COVID-19 cases and deaths data for London (UK Health Security Agency (UK HSA) 2021). To illustrate the difficulty of making predictions early on during an epidemic, we constrained our period of observation to lie between the 20$^{\text{th}}$ January 2020 to 22$^{\text{nd}}$ March 2020. We chose this period because, on the 23$^{\text{rd}}$ March, the UK government imposed the first lockdown (IFG 2021) and after that time, a model that does not allow changes to contact frequencies would not reproduce the observed dynamics. In what follows, the population size for London is assumed to be 9,002,488 (Office for National Statistics 2020).

We first plot the daily incidences and deaths in London over our chosen period, where we observe an approximately exponential growth in both observables over time.

```
# Load London COVID-19 data
df_london = read.csv('data/London_data.csv', header = TRUE)

# Extracting cases and deaths and renaming columns
cases_deaths_name_london = c('newCasesBySpecimenDate',
                             'newDailyNsoDeathsByDeathDate')
df_london = df_london[cases_deaths_name_london]
names(df_london)[names(df_london) == "newCasesBySpecimenDate"] <- "DailyCases"
names(df_london)[
  names(df_london) == "newDailyNsoDeathsByDeathDate"] <- "DailyDeaths"

# Set population size of London
population_size <- 9002488

# Extract COVID-19 data prior to lockdown
London = df_london[42:82, 1:2] # Start of lockdown (23/03/2020) is 83rd day
rownames(London) = seq(length = nrow(London))
London$time <- seq(length = nrow(London))

# Visualise the data
ggplot() +
  geom_line(data = London, aes(x = time, y = DailyCases, color = "Incidence")) +
  geom_line(data = London, aes(x = time, y = DailyDeaths, color = "Deaths")) +
  scale_color_manual(name = "",
```

```
                    values = c("Incidence" = "red", "Deaths" = "blue")) +
  labs(x = "time (days)", y = "counts") +
  theme(text = element_text(size = 12), legend.position = "bottom",
        legend.text = element_text(size = 10))
```



To estimate the parameters, we attempt maximum likelihood estimation, where we first define a log-likelihood function that is to be maximised. Here, we assume a Poisson likelihood function, where the observed cases ($\hat{C}_t$) and deaths ($\hat{D}_t$) are assumed to be sampled from distributions with the true cases (C_t) and deaths (D_t) as the Poisson means:

$$\hat{C}_t \sim \text{Pois}(C_t)$$
$$\hat{D}_t \sim \text{Pois}(D_t)$$

Note, that the above implicitly assumes that there is likely no systematic under-reporting of cases and deaths. (Thus C_t and D_t represent the SEIRD model outputs cases and deaths respectively.) Whilst this is unlikely to be true (particularly at the start of the epidemic), we do not include these processes here since it would lead to further identifiability issues of our model.

Our overall likelihood function is assumed to be the product of the likelihoods from the cases and deaths: implicitly, this assumes that these data are independent conditional on the model parameters.

We now define our log-likelihood function in R. The optimisation of the SEIRD model to the data is done only on the transmission parameters, which are the transition rates between the compartments, and not on the initial conditions. This is sufficient to describe the identifiability problem of the model. The initial conditions are fixed at

$$S(0) = 0.99799985, E(0) = 1 \times 10^{-7}, I(0) = 5 \times 10^{-8}, R(0) = 0.002.$$

These values are chosen from the optimisation of the SEIRD model on all parameters, including initial conditions, which is done without showing it in this vignette.

```
init_cond <- list(S0 = 0.99799985,
                  E0 = 1e-7,
                  I0 = 5e-8,
                  R0 = 0.002)
```

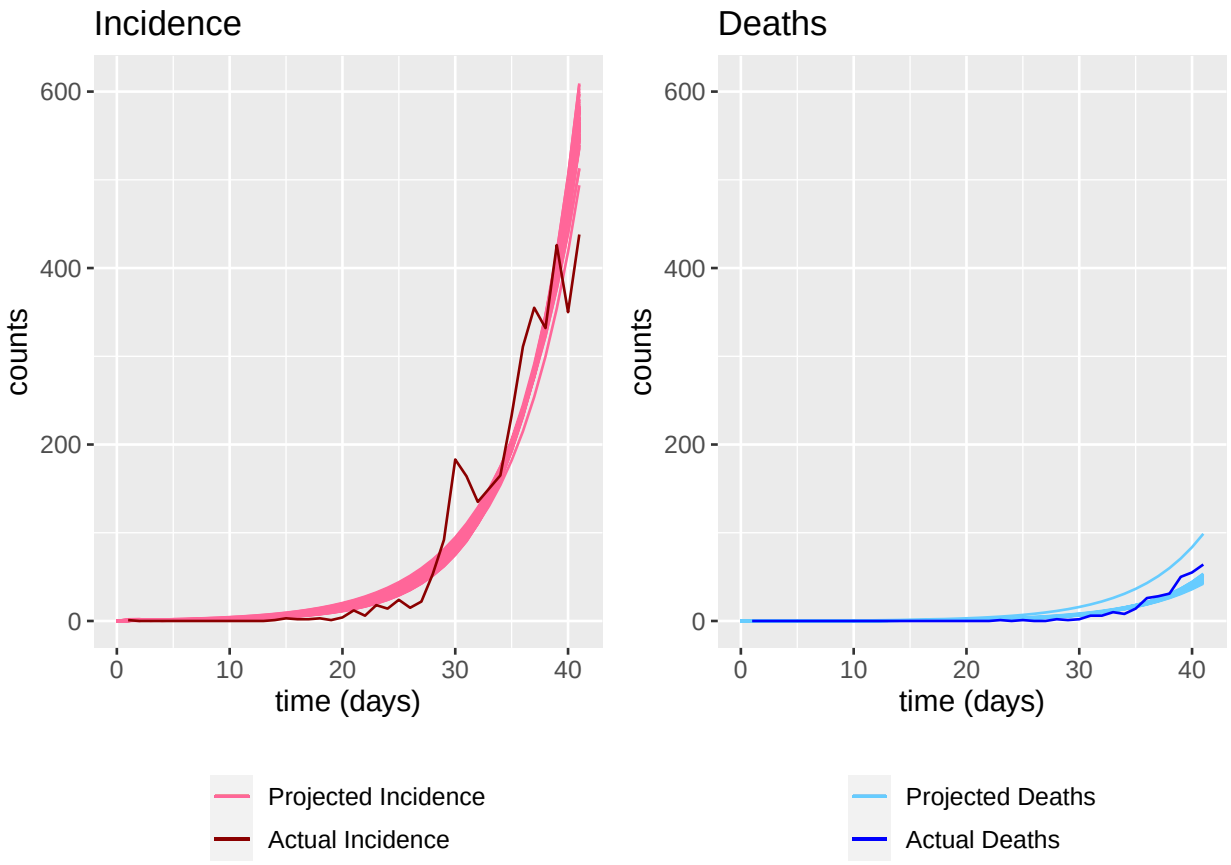
In an aim to determine the maximum likelihood estimates, we use a Nelder-Mead optimisation approach (Nelder and Mead 1965). Due to the lack of curvature of the objective function, however, obtaining the maximum likelihood estimates is difficult since many parameter sets provide a very similar fit to the observed data. We illustrate this by using a random set of parameters as an initial set for each optimisation run across 100 replicates. This results in a range of optimisation results representing different sets of the SEIRD model parameters.

```
# Set up conditions for optimisation
constraint_ui <- rbind(diag(4))
constraint_ci <- c(rep(0, 4))
repeat_num <- 100

# Run optimisation 100 times with different initial transmission parameters
for (repeats in 1:repeat_num) {
    trans_params_guess <- runif(4, min = 0, max = 1)
    result <- constrOptim(trans_params_guess,
                          RealData_LogLikelihoodFn, 'NULL', constraint_ui,
                          constraint_ci, method = "Nelder-Mead",
                          control = list(fnscale = -1, reltol = 1e-4),
                          inc_numbers = London$DailyCases,
                          death_numbers = London$DailyDeaths)

    # Create data frame to save initial guesses and optimised parameters
    if (repeats == 1) {
    optimised_para <- data.frame("beta_opt" = result$par[1],
                                 "kappa_opt" = result$par[2],
                                 "gamma_opt" = result$par[3],
                                 "mu_opt" = result$par[4],
                                 "obj_fn" = result$value)
    } else {
      optimised_para[nrow(optimised_para) + 1,] <- c(result$par, result$value)
    }
}

# save optimised parameters
write.csv(optimised_para, "data/London_parameters_100_optimisations.csv",
          row.names = FALSE)
```

```
# load saved results
optimised_para <- read.csv("data/London_parameters_100_optimisations.csv")
repeat_num <- nrow(optimised_para)
```

We now visualise the results of the optimisation by plotting the actual cases and deaths series versus the estimated series from each optimisation.

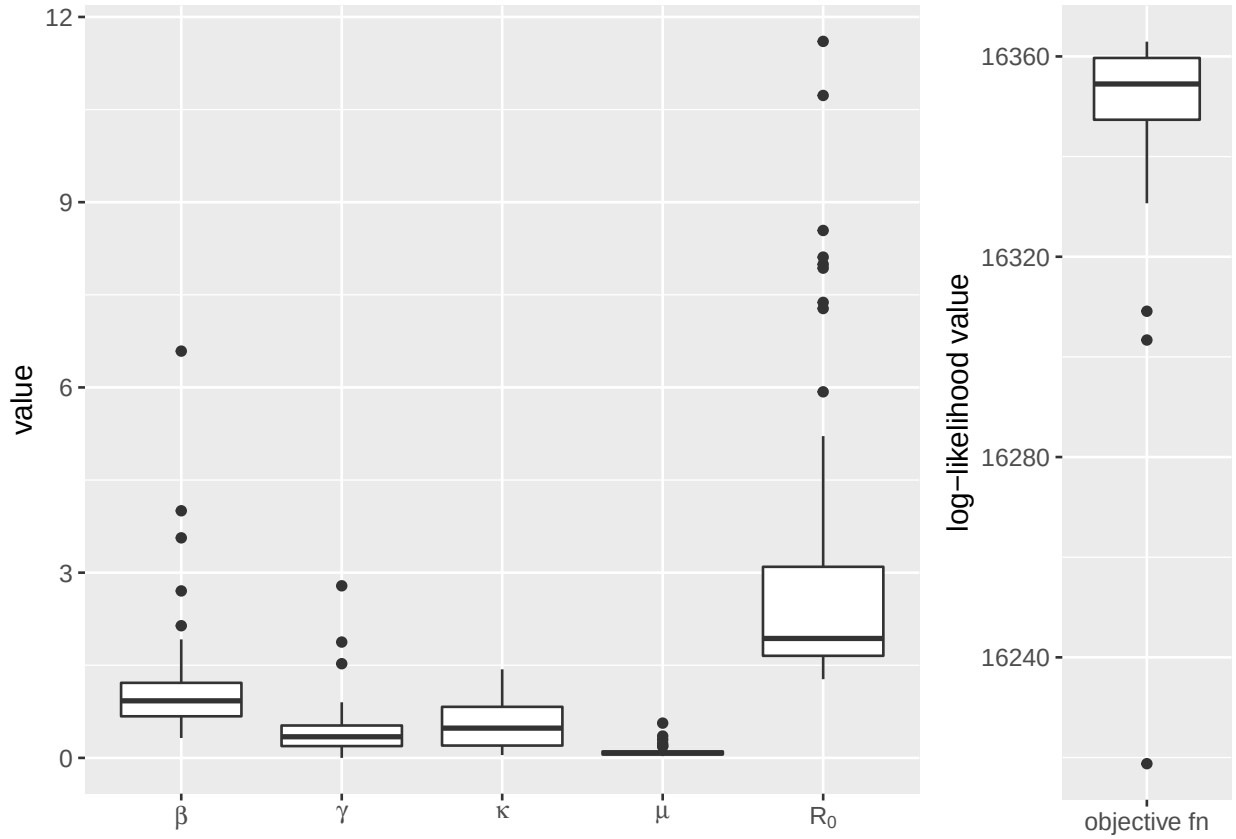
```
data_optimisation_plot(optimised_para, London)
```



This plot indicates that, across the 100 replicates, there were considerable differences in the model fits to the data. To further illustrate this, we now plot the parameter estimates, the R_0 value (the basic reproduction number) and the log-likelihood across the various optimisation results.

```
R0 <- data.frame("R0" = double())
for (i in 1:nrow(optimised_para)) {
  model <- SEIRD()
  transmission_parameters(model) <- list(beta = optimised_para[i,]$beta_opt,
                                         kappa = optimised_para[i,]$kappa_opt,
                                         gamma = optimised_para[i,]$gamma_opt,
                                         mu = optimised_para[i,]$mu_opt)
  initial_conditions(model) <- init_cond
  R0[nrow(R0) + 1,] <- R0(model)}

optimised_para["R0"] <- R0

# Organise data for box plot
plotting_data <- subset(optimised_para,
                        select = c(beta_opt, kappa_opt, gamma_opt, mu_opt, R0))
colnames(plotting_data) <- c("beta", "kappa", "gamma", "mu", "R0")
plotting_data$index <- seq(1, repeat_num)
plotting_data$legend <- rep("optimised", repeat_num)
data <- gather(plotting_data, key = "parameters",
```

```
                value = "value", -c(index, legend))

# Plot initial guesses and optimised parameters from optimisation
# in box plot format
para_boxplot <- ggplot(data, aes(x = parameters, y = value)) +
  geom_boxplot() +
  labs(x = "", y = "value") +
  scale_x_discrete(labels = c("beta" = bquote(beta),
                              "kappa" = bquote(kappa),
                              "gamma" = bquote(gamma),
                              "mu" = bquote(mu),
                              "R0" = bquote(R[0]))) +
  theme(text = element_text(size = 12))

# Create box plot for log-likelihood values of all optimisations
plotting_data <- subset(optimised_para, select = c(obj_fn))
colnames(plotting_data) <- c("objective fn")
plotting_data$index <- seq(1, repeat_num)
data <- gather(plotting_data, key = "likelihood_value",
                value = "value", -c(index))
objfn_boxplot <- ggplot(data, aes(x = likelihood_value, y = value)) +
  geom_boxplot() +
  labs(x = " ", y = "log-likelihood value") +
  theme(text = element_text(size = 12))
grid.arrange(para_boxplot, objfn_boxplot, widths = c(3, 1))
```



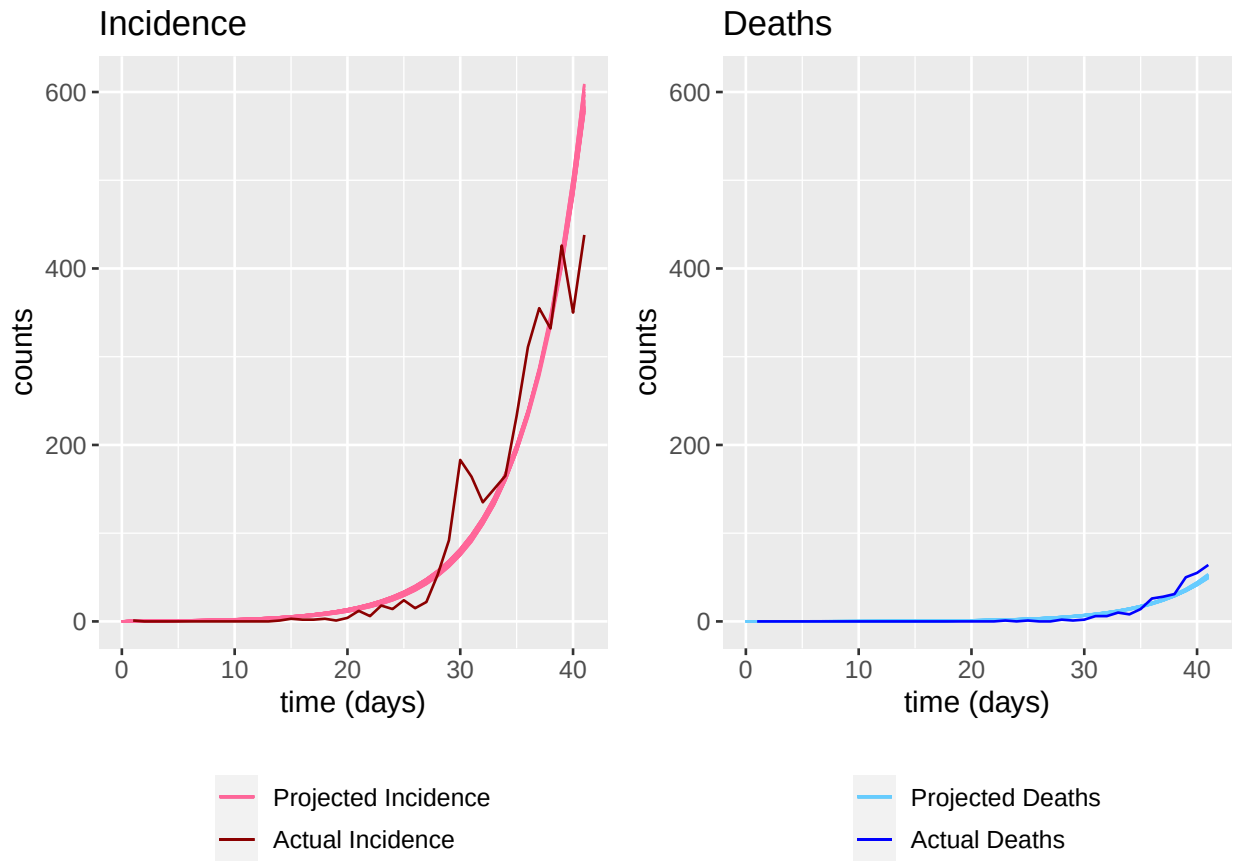

```
opt_df <- optimised_para[order(-optimised_para$obj_fn),]
print(opt_df[1,]$R0)
#> [1] 3.626846
```

This illustrates that some of the optimisation runs clearly became stuck at local minima, which is especially likely to occur when the objective function has only mild curvature near the maximum. This result also indicates the importance of performing multiple-start optimisation to improve the odds of obtaining parameter estimates closer to the true optimum.

Across the optimisations, there was notable uncertainty in R_0 indicating that, given such early data, it is a difficult quantity to estimate. But, conversely, it is precisely these early data which may offer best insights into this crucial parameter. At this stage in the epidemic for COVID-19, there was likely negligible immunity in the population and individuals were continuing to behave normally. These characteristics greatly simplify the task of estimating R_0, since the models used to fit it require fewer assumptions. There is thus a tension between waiting to obtain more data and restricting the data used for estimation to avoid having to make more assumptions about transmission processes. Our maximum likelihood fit had an R_0 value of 3.6, which is higher than the $R_0 = 2.4$ used in Imperial College's Report 9 (Ferguson et al. 2020). This illustrates that even relatively simple models like the SEIRD may be used to yield quick and potentially useful estimates of quantities of interest.

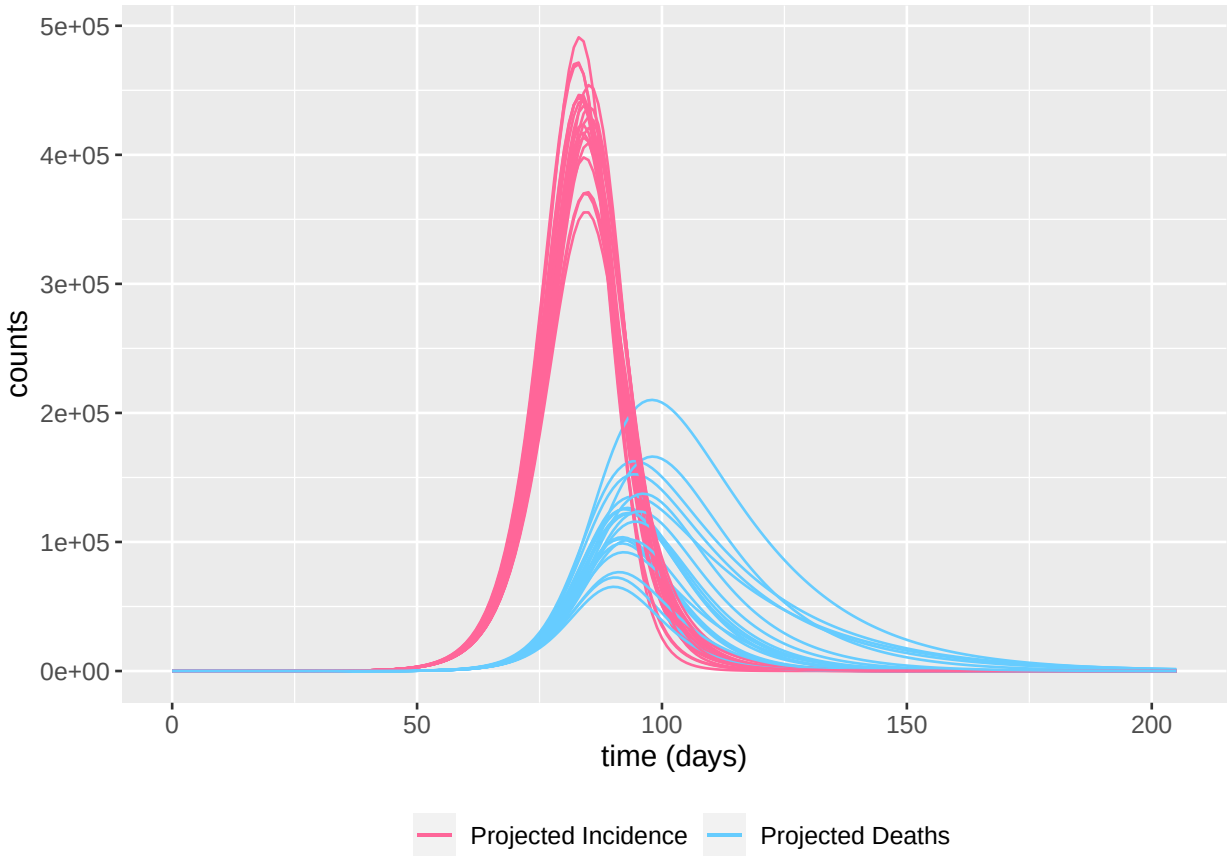
To provide a measure of uncertainty in our model fits, we chose the 20 optimisation parameter sets with the highest log-likelihood as representing possible model fits to the data. We recognise that a more formal uncertainty analysis like that obtained from Bayesian inference would yield a different measure of uncertainty. Our point is rather to illustrate that a number of parameter sets – here those obtained from different optimisation runs – provide a similar visual fit to the observed data.

```
# Sort data in order of highest log-likelihood value
chosen_repeats <- 20
top_opt_df <- opt_df[1:chosen_repeats,]

# Plot trajectories of optimisation with highest log-likelihood value
data_optimisation_plot(top_opt_df, London)
```



We now take those 20 optimisation parameter sets and use the SEIRD model to project forward cases and deaths beyond the period of observation.

```
# Plot prediction of the epidemic
optimisation_plot(top_opt_df, London,
                  time_length_scale = 5)
```



The figure above shows that even when the optimisation fits the real data well during early stages of the pandemic, very different outcomes are predicted. Indeed, here, the maximum difference in peak values of all projections is more than 1×10^5 in both cases and deaths. This result indicates the importance of including measures of uncertainty when making projections over the eventual course of the epidemic. It also indicates that raw cases and deaths data (particularly at the start of the epidemic) do not provide sufficient information to constrain model estimates.

Synthetic data studies and profile likelihood

Using real data, it is unclear whether the estimation results obtained are due to model misspecification or, because the data contains insufficient information to identify the causative parameters. As such, we now perform a simulated data study: where we first generate simulated cases and deaths series using known parameter values. We then use these data to perform a profile likelihood analysis (Cole, Chu, and Greenland 2014) to assess the identifiability of the system. Briefly, a single profile likelihood value is the maximum likelihood value obtained when fixing a parameter of interest to a given value and optimising across the remaining parameters. When this exercise is repeated across a grid of such fixed values, a profile likelihood trace is obtained, and its shape can be used to assess the identifiability of the system.

Generating synthetic data

We first generate cases and deaths series which qualitatively replicate the London COVID-19 data series which we investigate above. The initial conditions used to generate the synthetic data are taken as the values set previously when optimising transmission parameters for real data. The transmission parameters,

on the other hand, are taken to be the parameter set which yielded the maximum log-likelihood value from the optimisations on real data. Here, we assume that we know the exact population size. The population size is set to the population size of London.

```
transmission_para_name <- c("beta", "kappa", "gamma", "mu")
initial_conditions_name <- c("S0", "E0", "I0", "R0")

# Set up SEIRD model
model <- SEIRD()
simulating_para <- list(beta = top_opt_df[1,]$beta_opt, kappa = top_opt_df[1,]$kappa_opt,
                        gamma = top_opt_df[1,]$gamma_opt, mu = top_opt_df[1,]$mu_opt,
                        S0 = 0.99799985, E0 = 1e-7,
                        I0 = 5e-8, R0 = 0.002)
transmission_parameters(model) <- simulating_para[transmission_para_name]
initial_conditions(model) <- simulating_para[initial_conditions_name]

# Simulate the model to create synthetic data
times <- seq(0, 150, by = 1)
out_df <- run(model, times)
```

The model outputs do not well-approximate the stochasticity seen in the real data: as such, we use a Poisson distribution to generate (simulated) observed case and deaths data. To do so, we assume that Poisson mean for cases and deaths are the model outputs for each of those quantities:

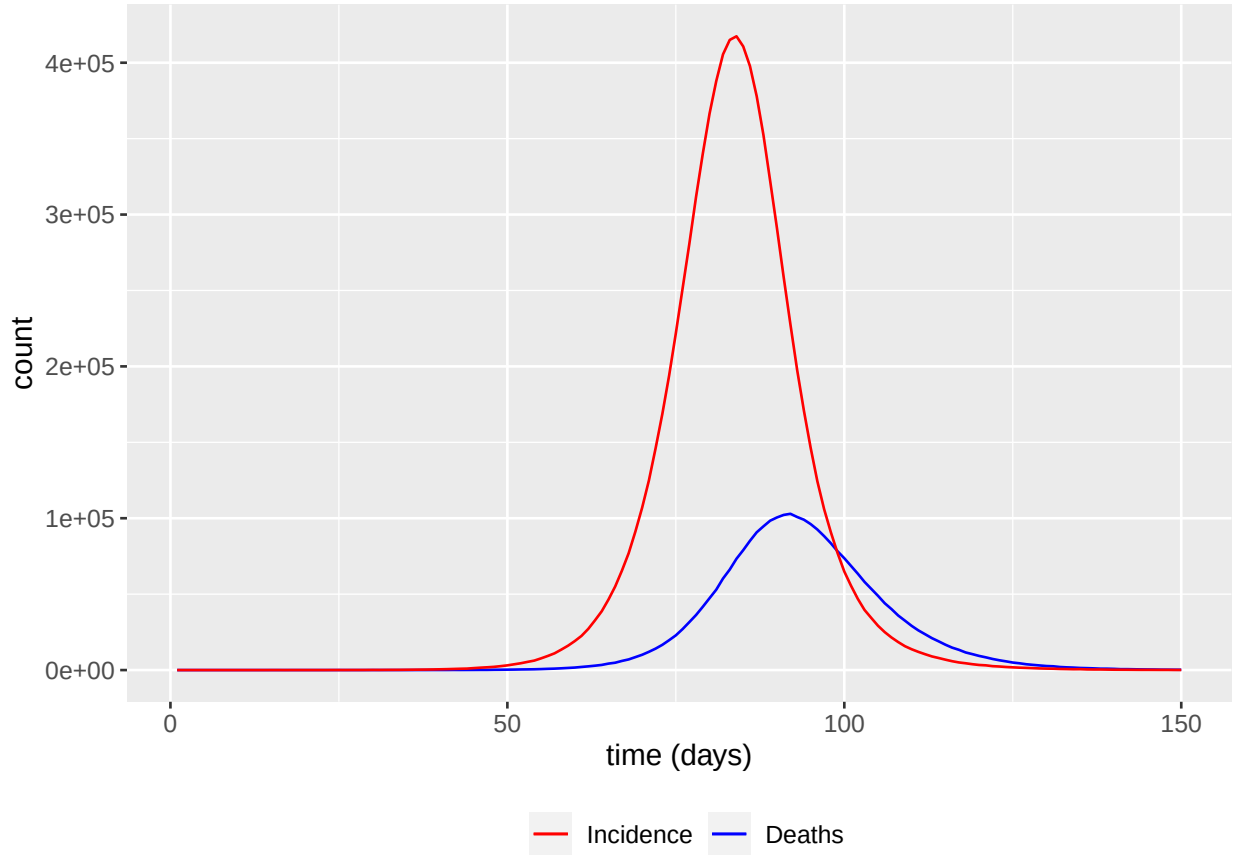
$$C^*(t) \sim \text{Pois}(C(t))$$
$$D^*(t) \sim \text{Pois}(D(t))$$

where $C^*(t)$ and $D^*(t)$ represent the observed cases and deaths at time t, and $C(t)$ and $D(t)$ are the true cases and deaths at the same time.

```
# add Poisson noise to synthetic data
testing_data <- spread(out_df$changes, compartment, value)
testing_data$Incidence <- testing_data$Incidence * population_size
testing_data$Deaths <- testing_data$Deaths * population_size
inc_noise <- rpois(1, testing_data$Incidence[1])
death_noise <- rpois(1, testing_data$Deaths[1])
for (i in 2:length(testing_data$Incidence)) {
  inc_rand <- rpois(1, testing_data$Incidence[i])
  inc_noise <- c(inc_noise, inc_rand)

  death_rand <- rpois(1, testing_data$Deaths[i])
  death_noise <- c(death_noise, death_rand)
}
testing_data$IncNoise <- inc_noise
testing_data$DeathNoise <- death_noise
testing_data <- testing_data[-1,]

# save results
write.csv(testing_data, "data/synthetic_data.csv", row.names = FALSE)
```

```
# load saved data
testing_data <- read.csv("data/synthetic_data.csv")
```

```r
# Plot synthetic data
ggplot() +
  geom_line(data = testing_data,
            aes(x = time, y = DeathNoise, color = "Deaths")) +
  geom_line(data = testing_data,
            aes(x = time, y = IncNoise, color = "Incidence")) +
  scale_color_manual(name = "",
                     values = c("Incidence" = "red", "Deaths" = "blue")) +
  labs(x = "time (days)", y = "count") +
  theme(text = element_text(size = 12), legend.position = "bottom",
        legend.text = element_text(size = 10))
```

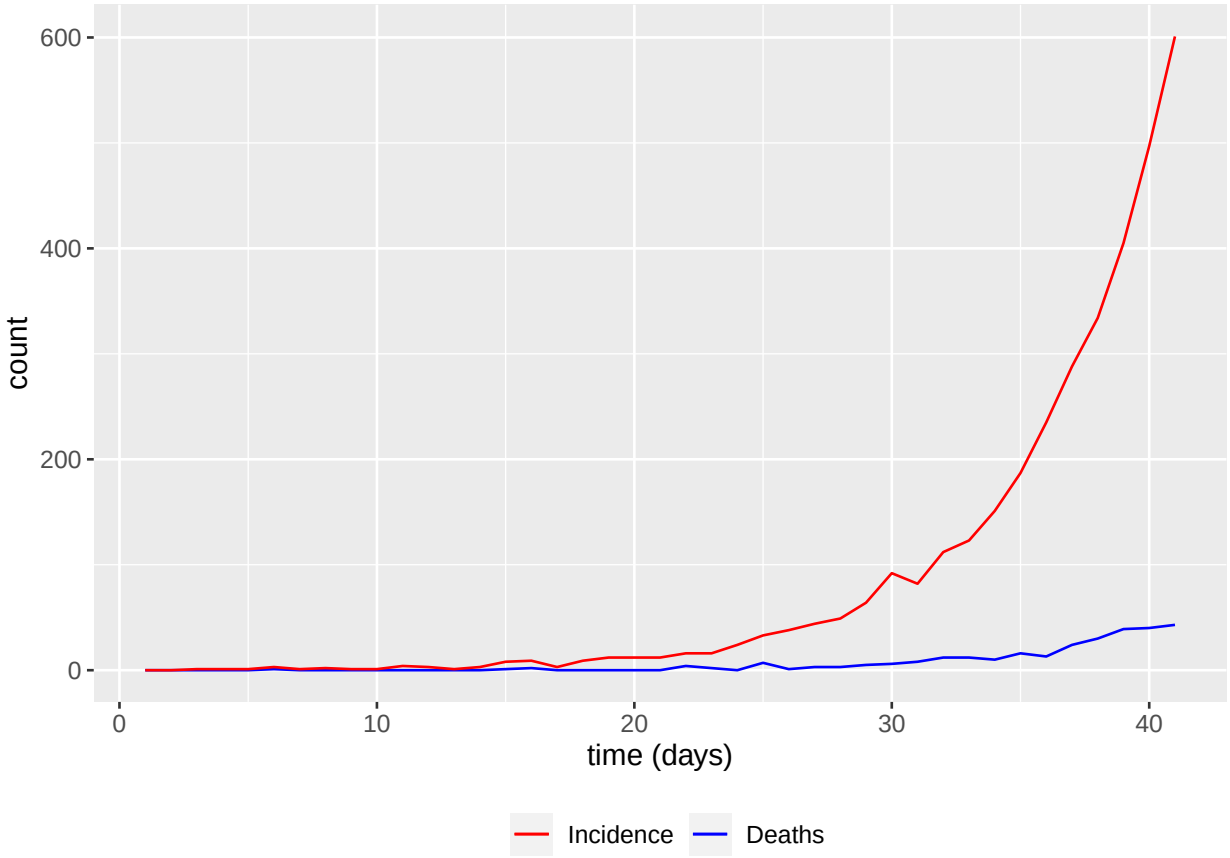


The figure above shows the synthetic data simulated for the full pandemic.

The synthetic data is extracted to have the same amount of time points (days) as the London COVID-19 data. The chosen data is visualised in the figure below.

```r
# Extract data prior to peak
testing_data_short <- testing_data[1:nrow(London),]

# Visualise data
ggplot() +
  geom_line(data = testing_data_short,
            aes(x = time, y = DeathNoise, color = "Deaths")) +
  geom_line(data = testing_data_short,
```

```
          aes(x = time, y = IncNoise, color = "Incidence")) +
  scale_color_manual(name = "",
                     values = c("Incidence" = "red", "Deaths" = "blue")) +
  labs(x = "time (days)", y = "count") +
  theme(text = element_text(size = 12), legend.position = "bottom",
        legend.text = element_text(size = 10))
```



We now use a Poisson log-likelihood to fit the SEIRD model to the simulated data. The profile likelihood is then constructed for the transmission parameters, while the initial conditions are fixed at their true values.

Profile likelihood of β and γ

We first suppose that only β and γ are unknown and apply the profile likelihood approach to β and γ only, fixing the other transmission parameters (κ and μ) at their known values. So when considering β, we scan across a range of different values from this parameter and perform optimisation to find the γ value which maximises the log-likelihood for that particular β value; when considering γ, we perform optimisation for β.

Our first step is to set the grid ranges over which to scan for each of β and γ. In both cases, the parameter ranges include the causative parameter values.

```
# Set interested parameters for profile likelihood
profile_parameters <- c('beta', 'gamma')

# Set range of values for interested parameters
```

```
beta_range <- c(seq(simulating_para$beta,
                    max(0.01, simulating_para$beta - 0.24),
                    by = -0.02),
                seq(simulating_para$beta,
                    simulating_para$beta + 0.5,
                    by = 0.02))
gamma_range <- c(seq(simulating_para$gamma,
                     max(0.01, simulating_para$gamma - 0.5),
                     by = -0.02),
                 seq(simulating_para$gamma,
                     simulating_para$gamma + 0.5,
                     by = 0.02))

# Create a data frame for the range of values
range_transmission <- data.frame(parameter = rep('beta', length(beta_range)),
                                 fixed_value = beta_range)
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('gamma', length(gamma_range)), fixed_value = gamma_range))
```

We now run the optimisations required for our profile likelihood determination. The function is defined in a separate R script for better readability.

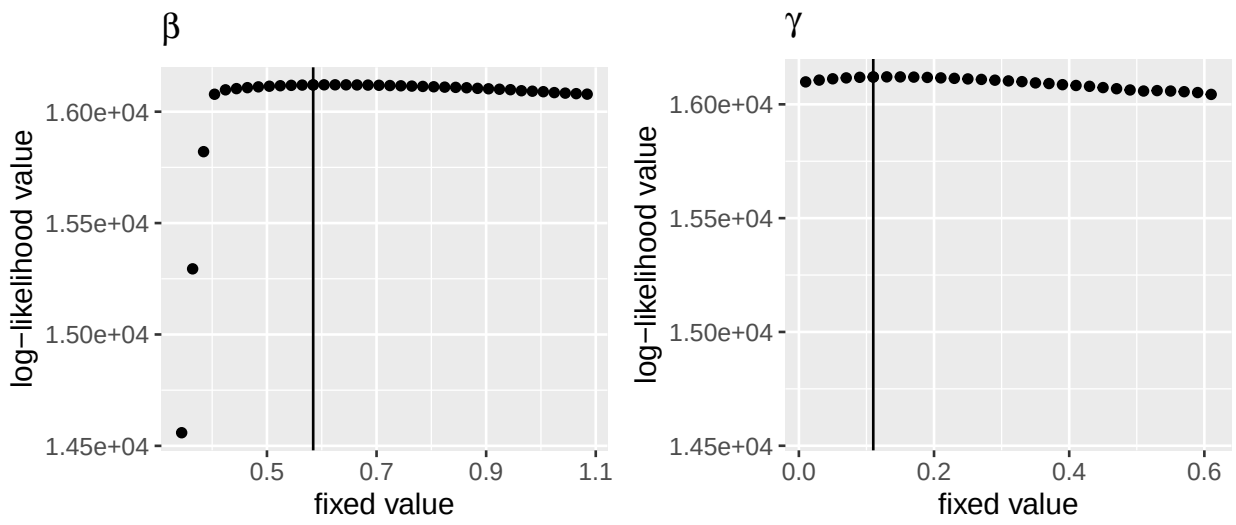
```
# Run optimisations to create profile likelihood
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data_short)

# Save results
write.csv(profile_likelihood,
          "data/synthetic_halftrend_2param_profilelikelihood.csv",
          row.names = FALSE)
```

While working through the optimisations for different fixed values of the parameter of interest, the previously optimised parameters were used as the initial guesses for the next round of optimisation.

```
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_halftrend_2param_profilelikelihood.csv")
```

We now plot the profile likelihoods for β and γ below. In this figure, we indicate the true parameter values by vertical lines. For both of these parameters, the profile likelihood is only gently curved near the maximum, and the parameters are only weakly identified.

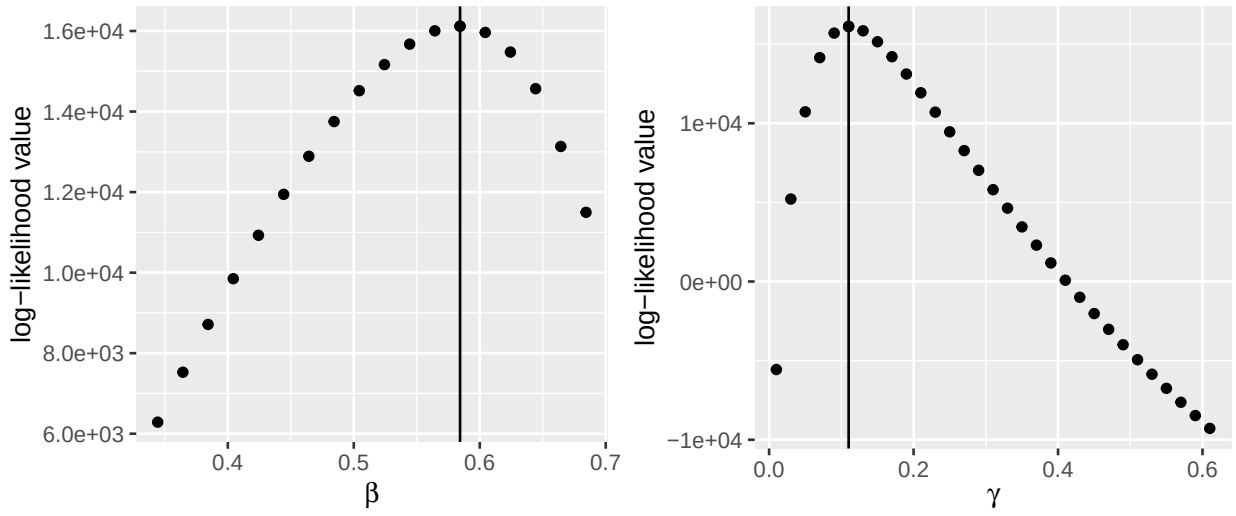
```
profilelikelihood_plot(profile_likelihood, profile_parameters)
```



We next consider an easier inference problem, when all of the parameters bar the parameter under consideration are fixed at their known values. In the plot below, in the left panel, the log-likelihood values for different values of β while fixing the remaining parameters to the true parameter values. A similar plot is shown in the right panel for γ. Both parameters are relatively well identified in this case.

```
beta_ll_range <- c(seq(simulating_para$beta,
                       max(0.01, simulating_para$beta - 0.24),
                       by = -0.02),
                   seq(simulating_para$beta,
                       simulating_para$beta + 0.1,
                       by = 0.02))

beta_loglikelihood_values <- data.frame(beta = double(),
                                        likelihood_value = double())
gamma_loglikelihood_values <- data.frame(gamma = double(),
                                        likelihood_value = double())
for (beta in beta_ll_range) {
        parameters <- list(beta = beta)
        ll_value <- LogLikelihoodFn(parameters, model = SEIRD(),
                                    inc_numbers = testing_data_short$IncNoise,
                                    death_numbers = testing_data_short$DeathNoise,
                                    profile_parameters = c('beta'),
                                    fixed_parameter = FALSE,
                                    fixed_parameter_value = 0)
        beta_loglikelihood_values[
          nrow(beta_loglikelihood_values) + 1,] <- c(beta, ll_value)}
for (gamma in gamma_range) {
        parameters <- list(gamma = gamma)
        ll_value <- LogLikelihoodFn(parameters, model = SEIRD(),
                                    inc_numbers = testing_data_short$IncNoise,
                                    death_numbers = testing_data_short$DeathNoise,
                                    profile_parameters = c('gamma'),
                                    fixed_parameter = FALSE,
                                    fixed_parameter_value = 0)
```

```
        gamma_loglikelihood_values[
          nrow(gamma_loglikelihood_values) + 1,] <- c(gamma, ll_value)}

beta_plot <- ggplot() +
  geom_point(data = beta_loglikelihood_values,
             aes(x = beta, y = likelihood_value)) +
  geom_vline(xintercept = simulating_para$beta) +
  labs(x = bquote(beta), y = "log-likelihood value") +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE))
gamma_plot <- ggplot() +
  geom_point(data = gamma_loglikelihood_values,
             aes(x = gamma, y = likelihood_value)) +
  geom_vline(xintercept = simulating_para$gamma) +
  labs(x = bquote(gamma), y = "log-likelihood value") +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE))
grid.arrange(beta_plot, gamma_plot, nrow = 1, ncol = 2)
```



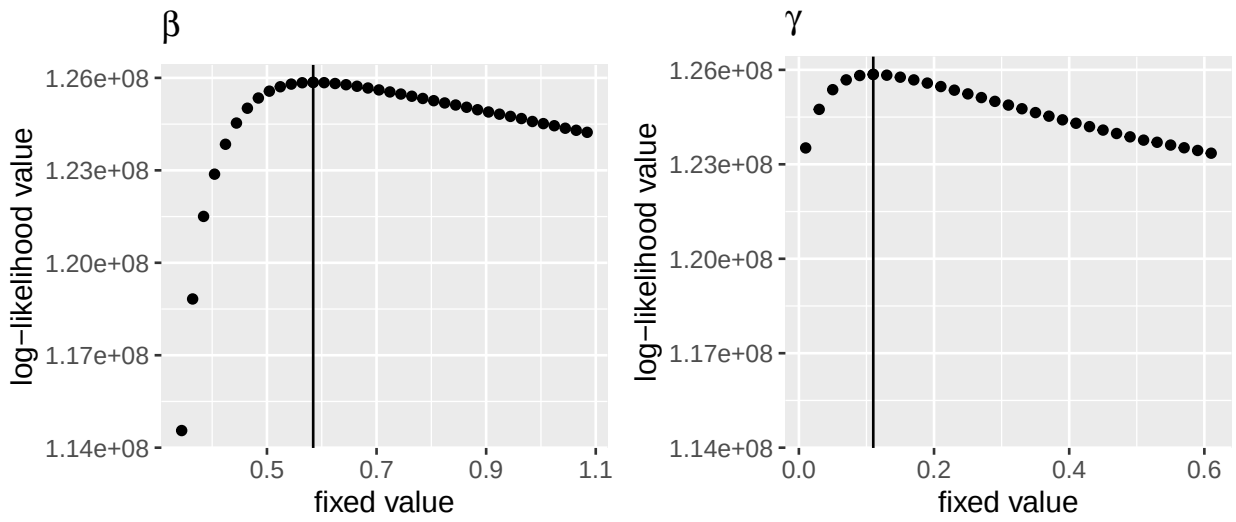
Profile likelihood with more information

If more data are collected, it may be possible to better identify a model's parameters, and here we investigate this for the SEIRD model, supposing that all parameters apart from β and γ are known.. We show an example by constructing the profile likelihood using synthetic data with time points that cover the first wave of the pandemic. The data used is daily and taken to run from the start of the simulated data until $t = 150$ days, which is well after the epidemic wave has passed.

```
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data,
                                            reltol = 1e-6)

# Save results
write.csv(profile_likelihood,
          "data/synthetic_fulltrend_2param_profilelikelihood.csv",
          row.names = FALSE)
```

```r
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_fulltrend_2param_profilelikelihood.csv")
```

```r
# Plot profile likelihood
profilelikelihood_plot(profile_likelihood, profile_parameters)
```



The figure above shows the profile likelihood of β and γ, with the same notations as the previous figure. This plot shows that the profile likelihoods for β and γ are both peaked at their true values, indicating that the model is identified. In this case, providing more data, encompassing the whole of the epidemic has helped us to estimate the parameters. Note that, for COVID-19, this quality of data would not typically be available because many countries put in place interventions aiming to divert the course of the epidemic. A byproduct of this is that the models required to reproduce the observed dynamics become more complicated: increasing the difficulty of the inference problem.

Profile likelihood of all transmission parameters

We now investigate the identifiability of the model when all four transmission parameters: β, κ, γ and μ, are unknown. For each of these parameters, we plot a profile likelihood assuming that we have access to data for the entire epidemic wave.

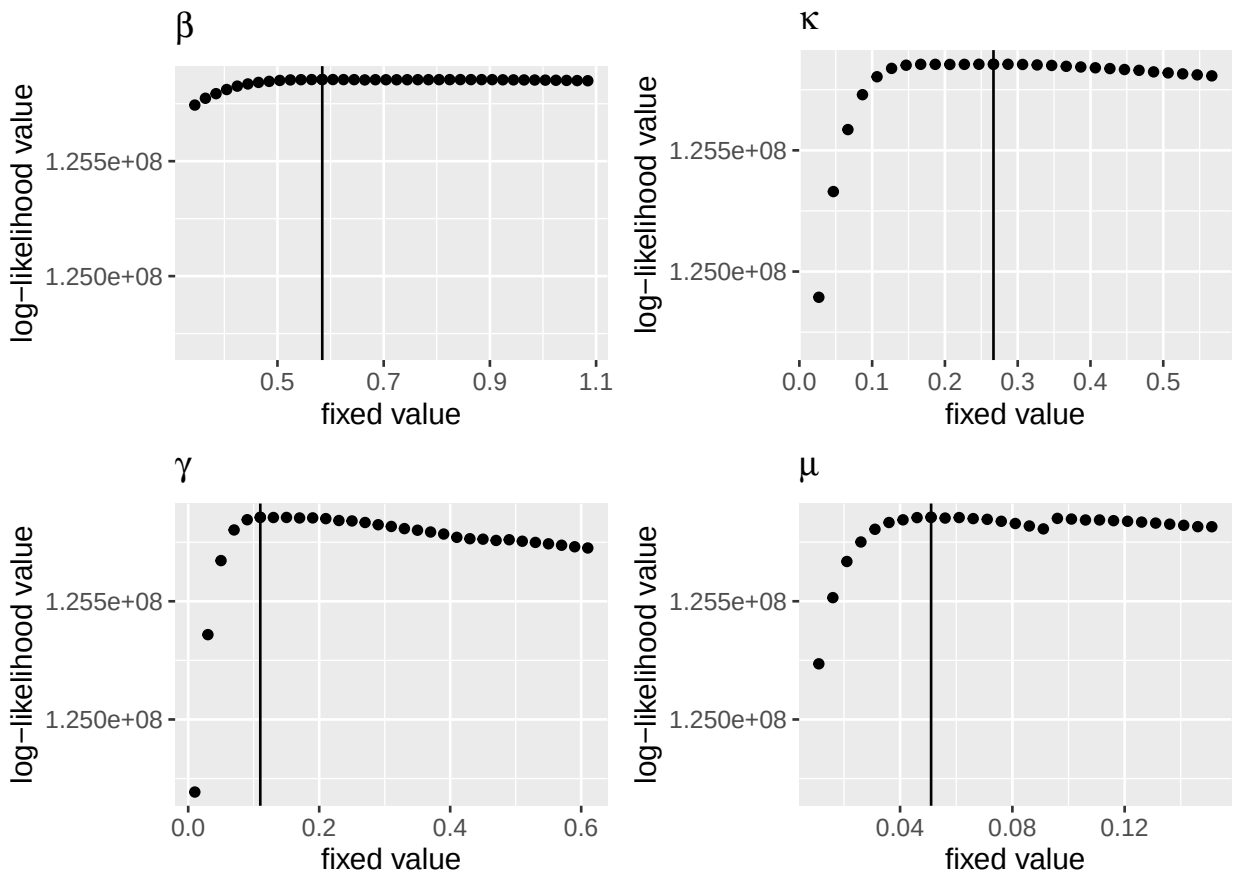
```r
# Set interested parameters for profile likelihood
profile_parameters <- c('beta', 'kappa', 'gamma', 'mu')

# Set range of values for interested parameters
kappa_range <- c(seq(simulating_para$kappa,
                     max(0.01, simulating_para$kappa - 0.3),
                     by = -0.02),
                 seq(simulating_para$kappa,
                     simulating_para$kappa + 0.3,
                     by = 0.02))
mu_range <- c(seq(simulating_para$mu,
                  max(0.01, simulating_para$mu - 0.1),
                  by = -0.005),
              seq(simulating_para$mu,
```

```
                  simulating_para$mu + 0.1,
                  by = 0.005))

# Create a data frame for the range of values
range_transmission <- data.frame(parameter = rep('beta', length(beta_range)),
                                 fixed_value = beta_range)
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('kappa', length(kappa_range)), fixed_value = kappa_range))
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('gamma', length(gamma_range)), fixed_value = gamma_range))
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('mu', length(mu_range)), fixed_value = mu_range))
```

```
# Run optimisations to create profile likelihood
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data,
                                            reltol = 1e-6)
# Save results
write.csv(profile_likelihood,
          "data/synthetic_fulltrend_4param_profilelikelihood.csv",
          row.names = FALSE)
```

```
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_fulltrend_4param_profilelikelihood.csv")
```

```
# Plot profile likelihood
profilelikelihood_plot(profile_likelihood, profile_parameters)
```



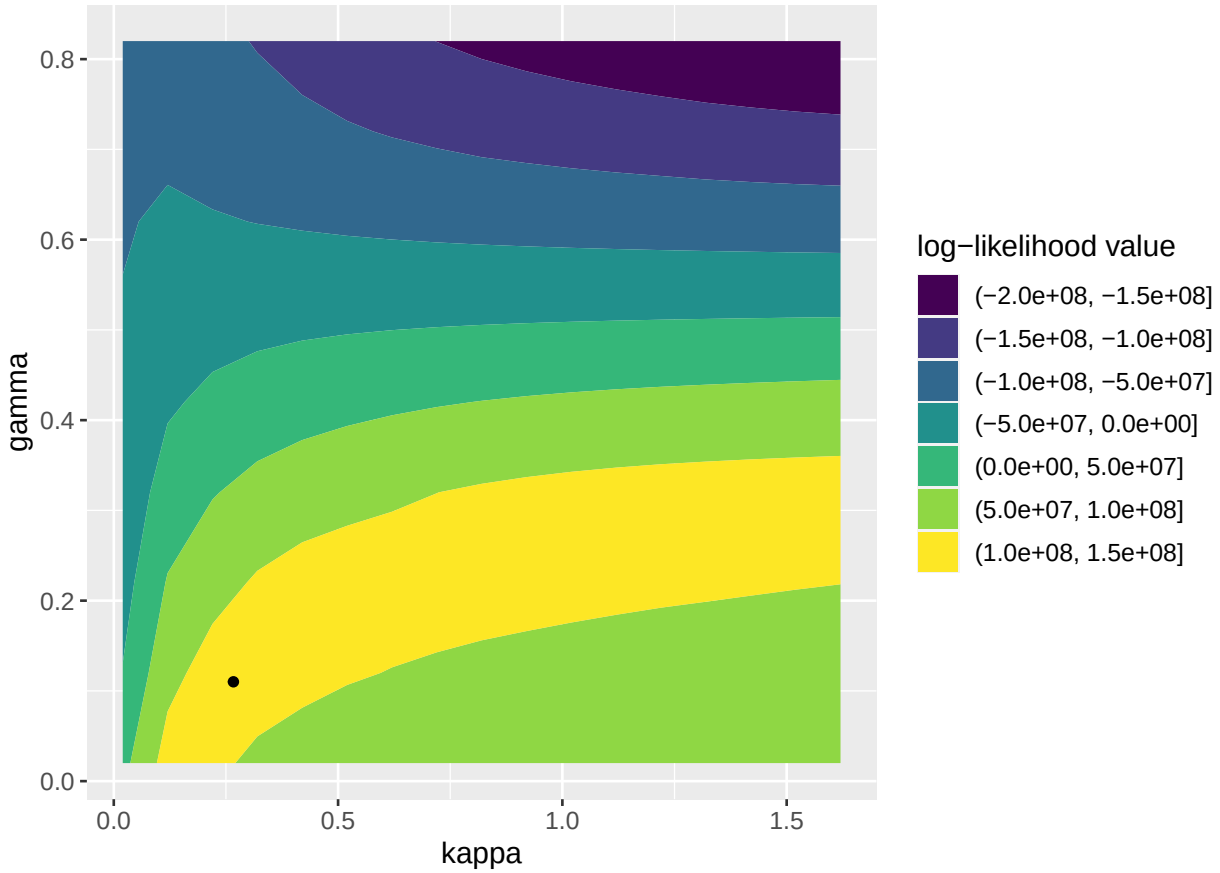
All transmission parameters, except for γ, are not identifiable since there is not a clear maximum to the profile likelihood in each case. Clearly, by adding more unknown parameters, this has increased the difficulty of the inference problem. Intuitively, many combinations of the parameters yield similar fits to the data (as when we initially searched for the maximum likelihood estimates for the real data), meaning that there are wide ranges of each parameter that may yield comparable log-likelihoods.

Profile likelihood of β, γ and μ

A larger value of κ, implying a shorter latent period, means that an epidemic will appear sooner and that the peak of the wave will be higher. This could be partly offset by an increased value of γ, meaning that individuals are infectious for less time, resulting in a smaller epidemic that appears later. Because of this similarity in the effect of each of these parameters on the outputs, there is a positive correlation in a log-likelihood plot in (κ, γ) space, explaining why it is difficult to estimate both of these parameters from data.

```
profile_parameters <- c('kappa', 'gamma')
kappa_ll_range <- seq(0.02, 1.7, by = 0.1)
gamma_ll_range <- seq(0.02, 0.9, by = 0.1)
true_parameters <- data.frame(simulating_para[profile_parameters])
loglikelihood_values <- data.frame(kappa = double(), gamma = double(),
                                   likelihood_value = double())

# Calculate log-likelihood values for a range of kappa and gamma values
for (kappa in kappa_ll_range) {
```

```
  for (gamma in gamma_ll_range) {
    parameters <- list(kappa = kappa, gamma = gamma)
    ll_value <- LogLikelihoodFn(parameters, model = SEIRD(),
                                inc_numbers = testing_data$IncNoise,
                                death_numbers = testing_data$DeathNoise,
                                profile_parameters = profile_parameters,
                                fixed_parameter = FALSE,
                                fixed_parameter_value = 0)
            loglikelihood_values[
              nrow(loglikelihood_values) + 1,] <- c(kappa, gamma, ll_value)
            }}

# Plot log-likelihood values for a range of kappa and gamma values

ggplot() +
  geom_contour_filled(data = loglikelihood_values,
            aes(x = kappa, y = gamma, z = likelihood_value)) +
  geom_point(data = true_parameters,
             aes(x = kappa, y = gamma)) +
  labs(fill = "log-likelihood value") +
  theme(text = element_text(size = 12),
        legend.text = element_text(size = 10))
```



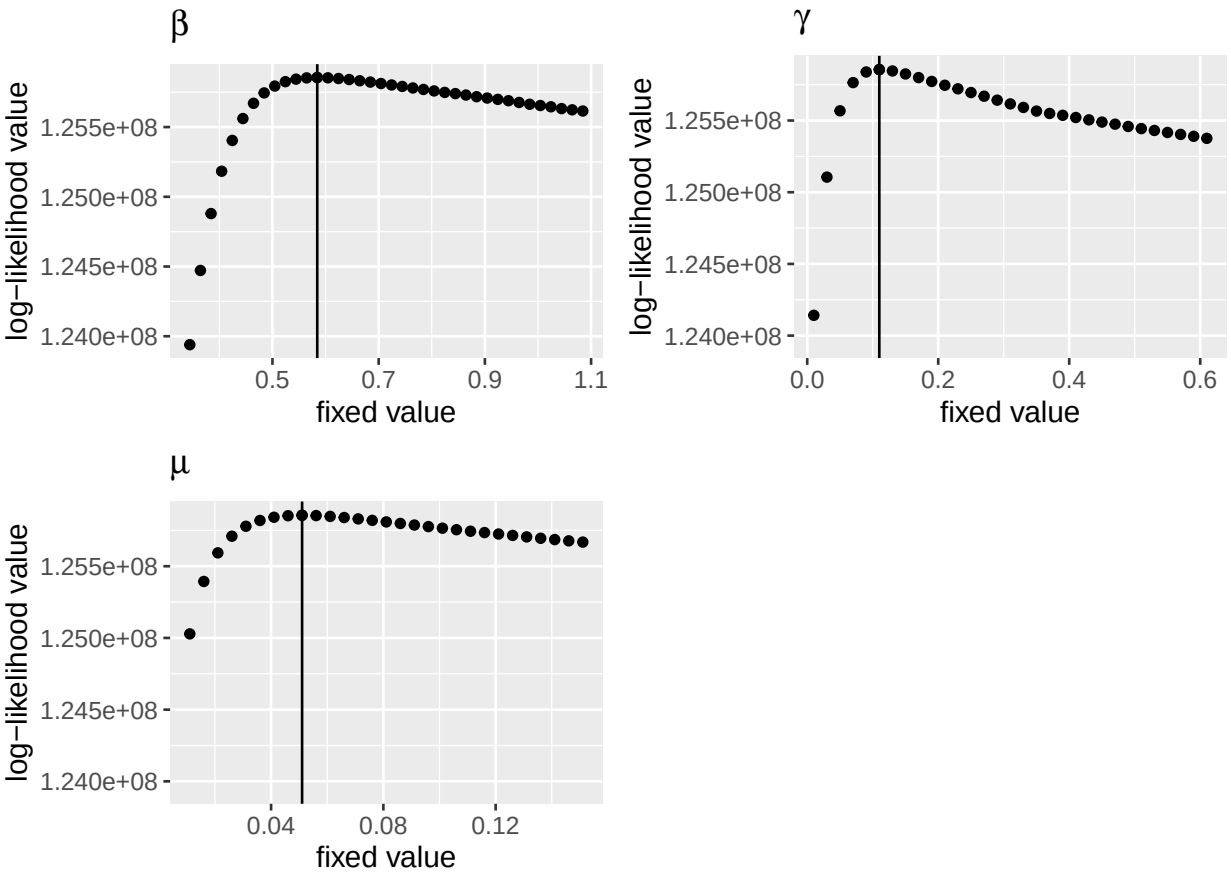
Because κ and γ cannot uniquely be identified by the cases and deaths data, we fix κ at its true value and repeat the profile likelihood analysis.

```r
# Set interested parameters for profile likelihood
profile_parameters <- c('beta', 'gamma', 'mu')

# Create a data frame for the range of values
range_transmission <- data.frame(parameter = rep('beta', length(beta_range)),
                                 fixed_value = beta_range)
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('gamma', length(gamma_range)), fixed_value = gamma_range))
range_transmission <- rbind(range_transmission, data.frame(
  parameter = rep('mu', length(mu_range)), fixed_value = mu_range))
```

```r
# Run optimisations to create profile likelihood
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data,
                                            reltol = 1e-6)
# Save results
write.csv(profile_likelihood,
          "data/synthetic_fulltrend_3param_profilelikelihood.csv",
          row.names = FALSE)
```

```r
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_fulltrend_3param_profilelikelihood.csv")
```

```r
# Plot profile likelihood
profilelikelihood_plot(profile_likelihood, profile_parameters)
```



With only β, γ and μ unknown, the model is identified given data for the full epidemic.

Conclusion

Using both real data on disease cases and deaths from before the peak of the 1st epidemic wave of COVID-19 in London and simulated data, we have demonstrated that the SEIRD model is not fully identified. Further, even if data on a complete (and unmitigated) epidemic are available, the model remains unidentified. This identifiability issue is because κ, the rate at which infected individuals become infectious, and γ, the rate of recovery, both have similar impacts on the resultant dynamics, meaning it is difficult to separate one's effect from the other's. This means that, in reality, one or both of these parameters is typically estimated using other sources of data – usually laboratory studies. These parameters are then either fixed at these outside estimates when inference is performed, or strongly informative priors (incorporating information from these external studies) are used.

Overall, our results show that performing inference for transition dynamics models can be plagued with identifiability issues. Here, we illustrated this using the relatively simple SEIRD model. This model is the simplest in our package and has substantially fewer parameters than the complex models that have often been used to inform policymaking. This means that assessing their identifiability using simulated data and approaches such as the profile likelihood method before performing inference on real data is likely to have great utility. These larger models are also likely to rely more on information from other studies, and, due to having more parameters, may have many more combinations of parameters that yield similar fits to the data. As such, it is particularly key to present sensitivity analyses when publishing predictions using these models.

References

Cole, S R, H Chu, and S Greenland. 2014. "Maximum Likelihood, Profile Likelihood, and Penalized Likelihood: A Primer." *American Journal of Epidemiology* 179 (2): 252–60.

Ferguson, N, D Laydon, G Nedjati Gilani, N Imai, K Ainslie, M Baguelin, S Bhatia, et al. 2020. "Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand."

IFG. 2021. "Timeline of UK coronavirus lockdowns, March 2020 to March 2021." https://www.instituteforgovernment.org.uk/sites/default/files/timeline-lockdown-web.pdf.

Nelder, J A, and R Mead. 1965. "A Simplex Method for Function Minimization." *The Computer Journal* 7 (4): 308–13.

Office for National Statistics. 2020. "Census Output Area Population Estimates - London, England."

UK Health Security Agency (UK HSA). 2021. "Coronavirus (COVID-19) in the UK." https://coronavirus.data.gov.uk/.