Optimisation with 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

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

Choose between plotting cached data or run optimisation

```r
# 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

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta S I$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \beta S I - \kappa E$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \kappa E - (\gamma + \mu) I$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I$$

$$\frac{\mathrm{d}D}{\mathrm{d}t} = \mu I$$

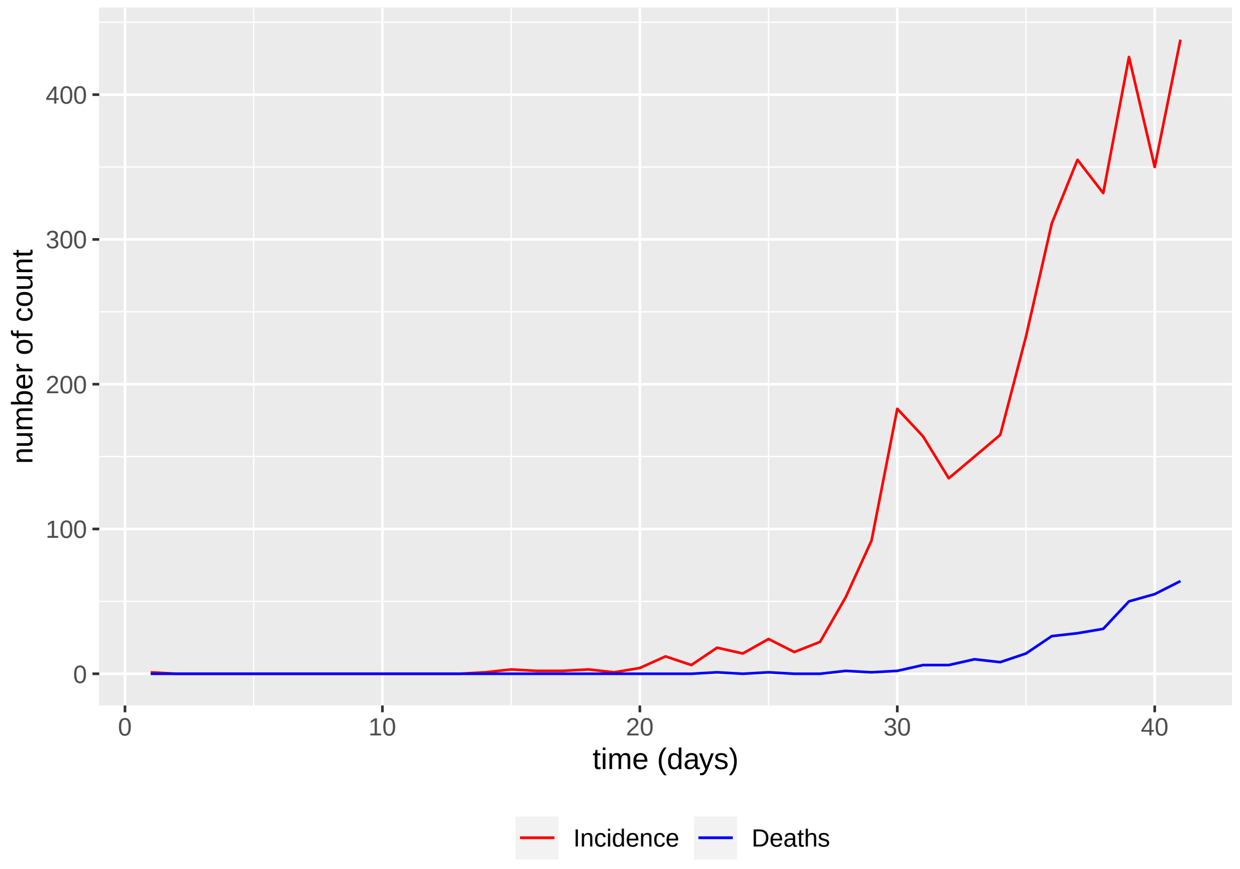
where β, κ, γ and μ are the infection rate, incubation rate, recovery rate and death rate respectively. More details can be found in the SEIRD vignette from the `comomodels` package.

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

We begin by fitting the SEIRD model to real COVID-19 case 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 20th January 2020 to 22nd March 2020. We chose this period because, on the 23rd 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.

```r
# 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 = "number of count") +
  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 in this vignette.

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

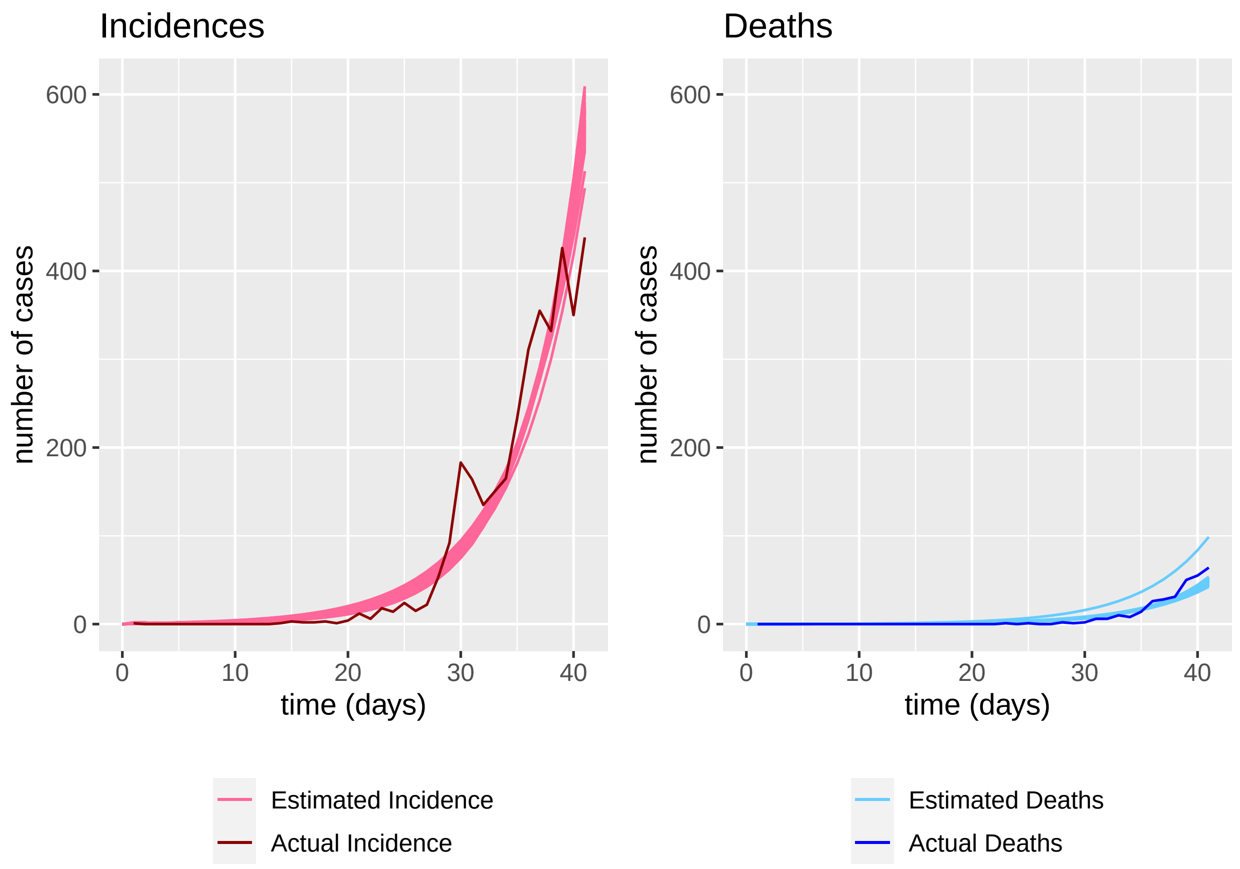
In an aim to determine the maximum likelihood estimates, we use a Nelder-Mead optimisation approach (Nelder and Mead 1965). Due to the lack of curvature of the objective function, however, obtaining the maximum likelihood estimates is difficult since many parameter sets provide a very similar fit to 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.

```r
# 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)
```

```r
# 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.

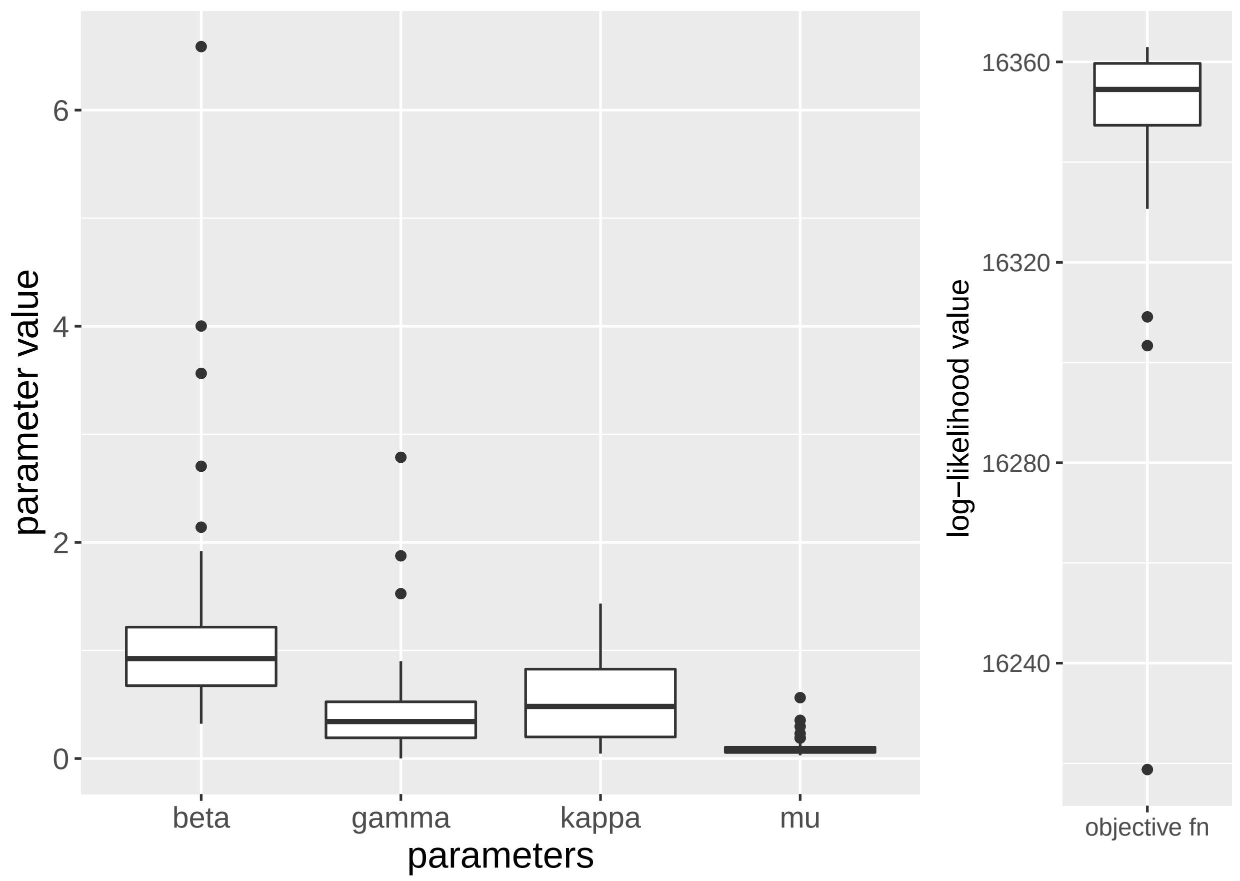
```r
data_optimisation_plot(optimised_para, London)
```



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 estimated parameter values and log-likelihood value from each of the replicates for each of the four model parameters.

```r
# Organise data for box plot
plotting_data <- subset(optimised_para,
                        select = c(beta_opt, kappa_opt, gamma_opt, mu_opt))
colnames(plotting_data) <- c("beta", "kappa", "gamma", "mu")
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(y = "parameter value") +
  theme(text = element_text(size = 15))

# 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")
```

```r
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))
```

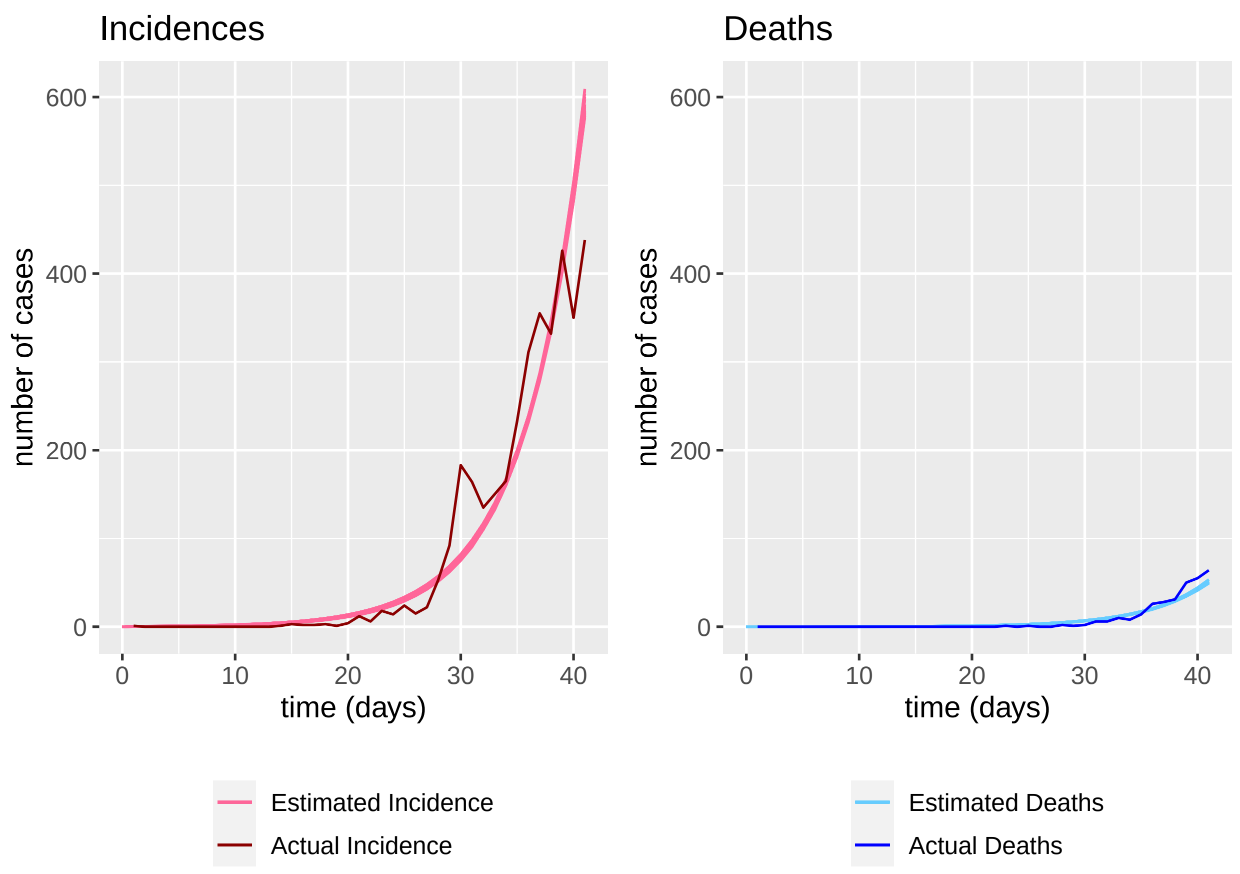


This illustrates that some of the optimisation runs clearly became stuck at local minima, which is especially likely to occur when the objective function has only mild curvature near the maximum. This result also indicates the importance of performing multiple-start optimisation to improve the odds of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

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.

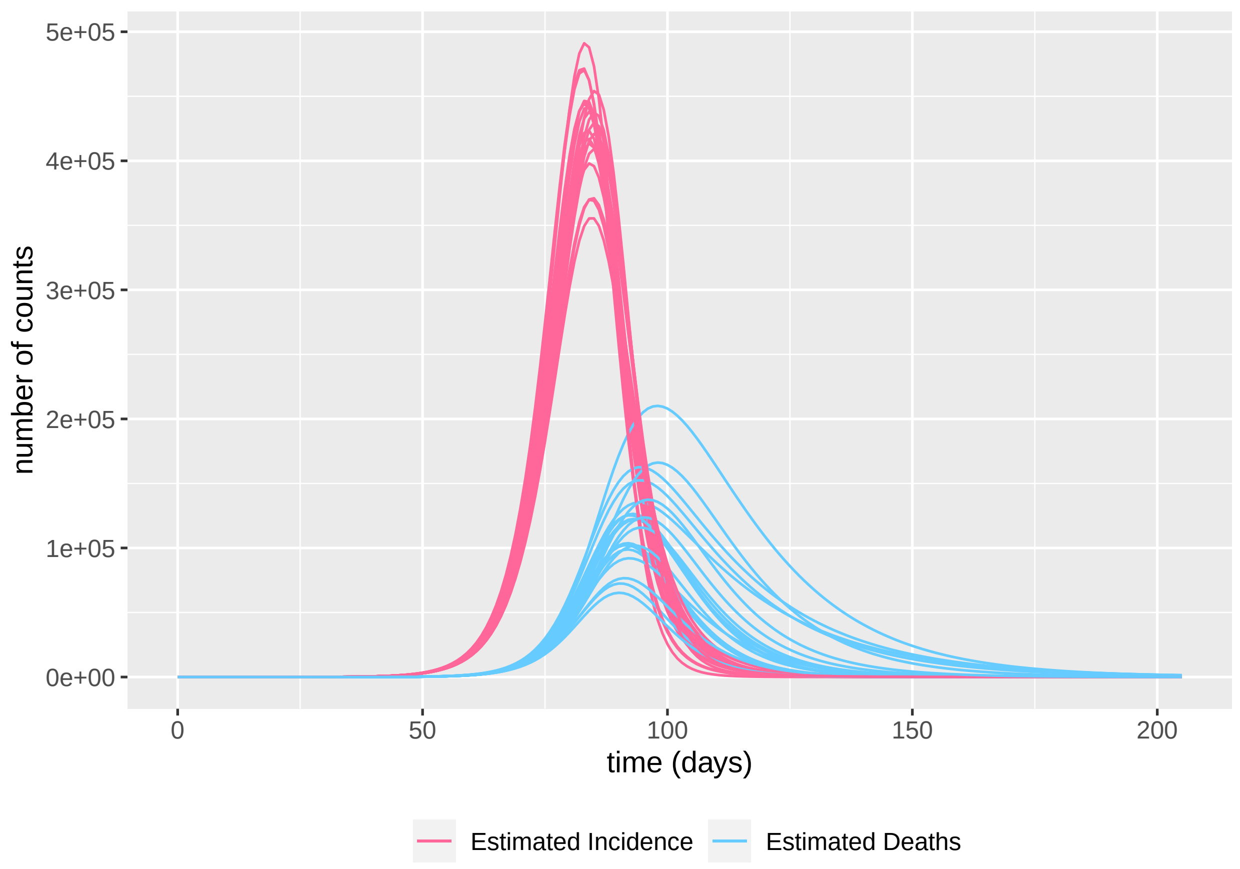
```r
# Sort data in order of highest log-likelihood value
chosen_repeats <- 20
opt_df <- optimised_para[order(-optimised_para$obj_fn),]
top_opt_df <- opt_df[1:chosen_repeats,]
```

```r
# 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.

```r
# 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, indeed, 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 case and deaths series which qualitatively replicate the London COVID-19 data series which we investigate above. The initial conditions and transmission parameters used to generate the synthetic data are as follows:

$$S(0) = 0.9999988, E(0) = 1 \times 10^{-7}, I(0) = 1 \times 10^{-7}, R(0) = 1 \times 10^{-6},$$
$$\beta = 0.91, \kappa = 0.87, \gamma = 0.53, \mu = 0.097.$$

Here, we assume that we know the exact population size. The population size is set arbitrarily to 1×10^4.

```r
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 = 9.1e-1, kappa = 8.7e-1,
                        gamma = 5.3e-1, mu = 9.7e-2,
                        S0 = 9.999988e-1, E0 = 1e-7,
                        I0 = 1e-7, R0 = 1e-6)
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:

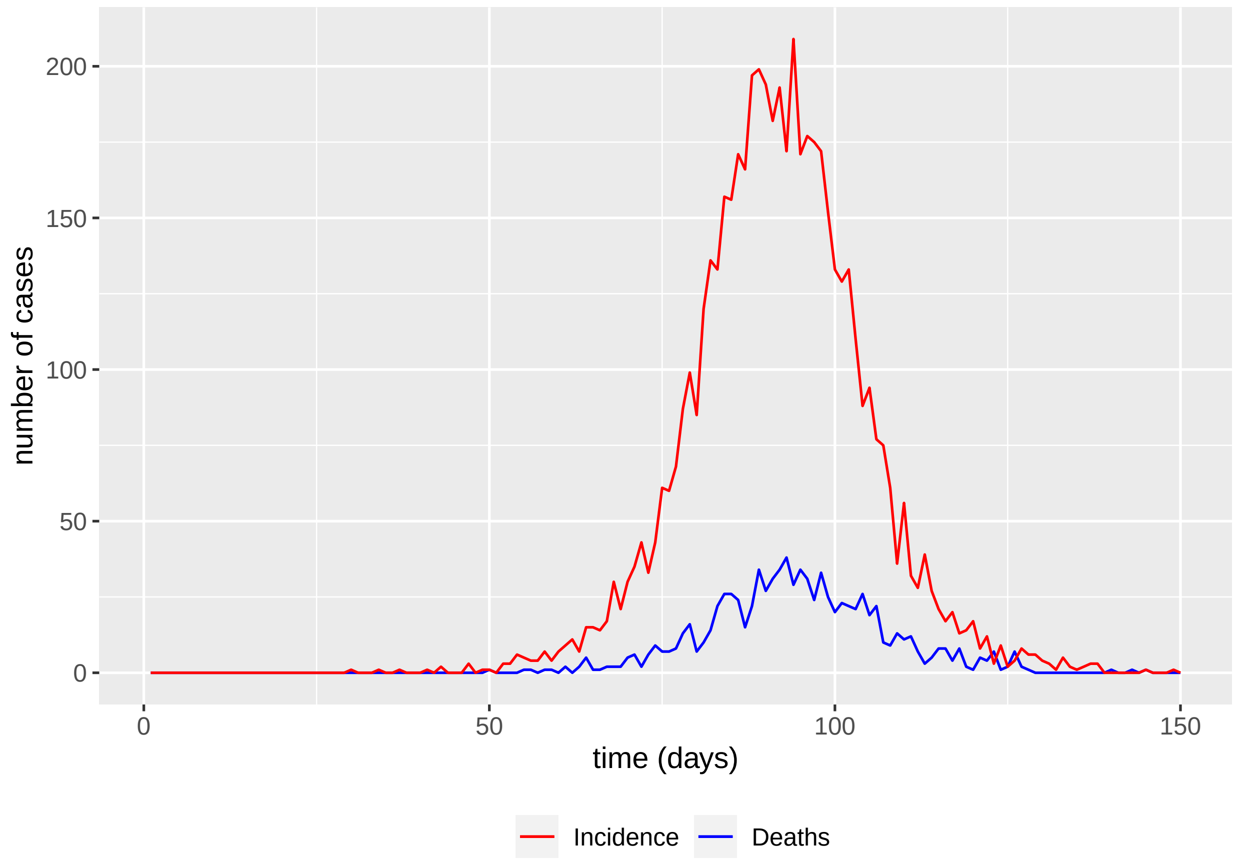
$$C^*(t) \sim \mathsf{Pois}(C(t))$$
$$D^*(t) \sim \mathsf{Pois}(D(t))$$

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.

```r
population_size <- 1e4

# 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)
```

```r
# load saved data
population_size <- 1e4
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 = "number of cases") +
  theme(text = element_text(size = 12), legend.position = "bottom",
        legend.text = element_text(size = 10))
```

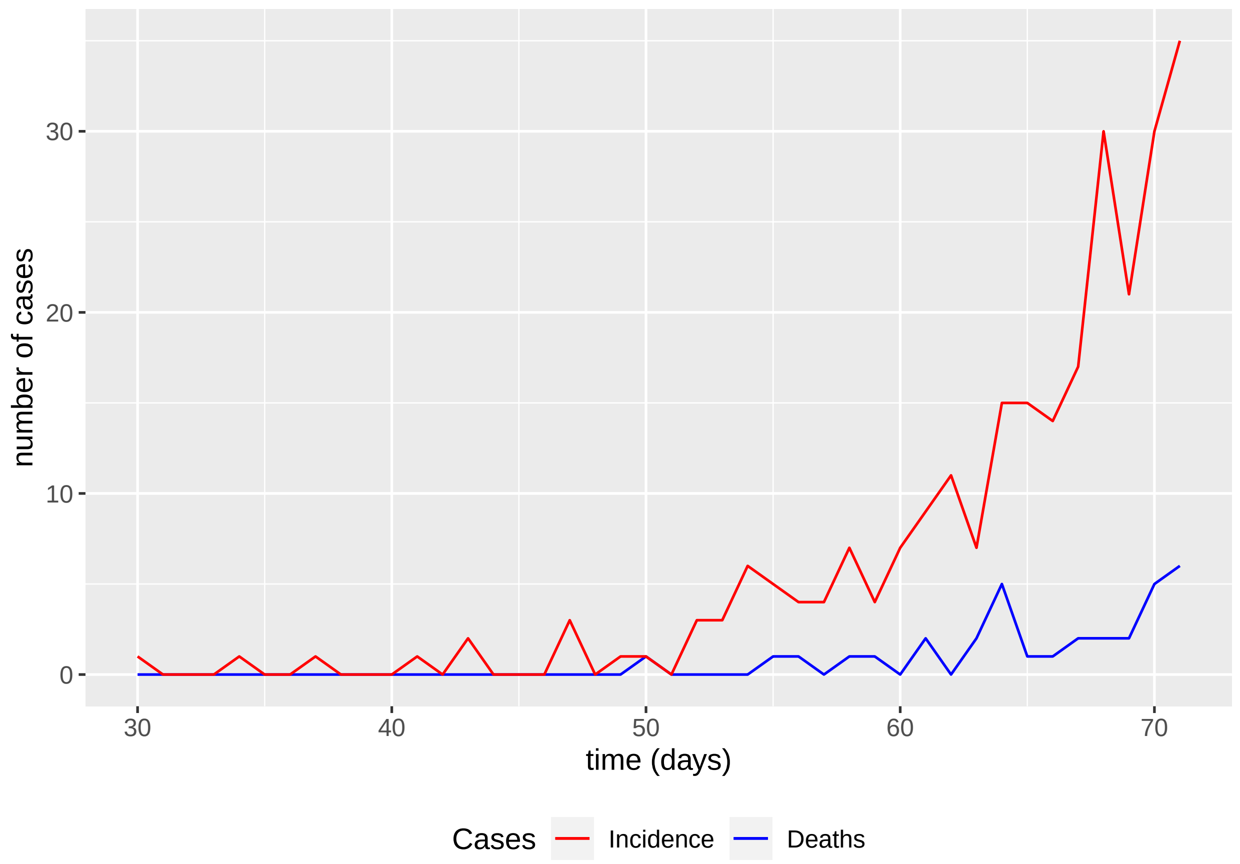


The figure above shows the synthetic data simulated for the full pandemic.

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

```r
# Extract data prior to peak
first_nonzero_ind <- which(testing_data$IncNoise > 0)[1]
testing_data_short <- testing_data[
  first_nonzero_ind:(first_nonzero_ind + 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 = "Cases",
                     values = c("Incidence" = "red", "Deaths" = "blue")) +
  labs(x = "time (days)", y = "number of cases") +
  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, which are their respective state values from the simulated data at the time points of the first incidence.

Profile likelihood of β and γ

We start by illustrating an identified model by fixing all parameters at their known values apart from β and γ.

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.

```r
# Set interested parameters for profile likelihood

simulating_para[c("S0", "E0", "I0", "R0")] <- out_df$states[
  (out_df$states$time == 30), ]$value[c(1:4)]

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 - 0.5),
                    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))
```

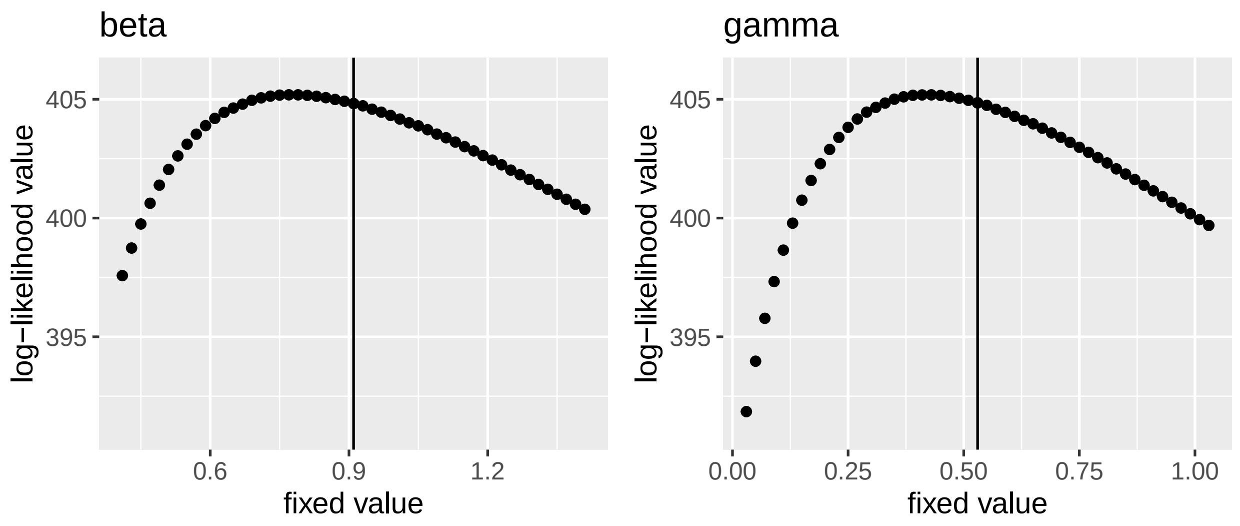
We now run the optimisations required for our profile likelihood determination. The function is defined in a separate R script for better readability.

```r
# 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.

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

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



The figure above shows the log-likelihood value obtained by fixing, in turn, each parameter of interest and optimising the remaining transmission parameters. The true parameter values used to generate the data are indicated by the vertical lines. The profile likelihood shows a single peak for both β and γ, which implies that the parameters are identifiable when the SEIRD model has only two transmission parameters.

However, we note that, due to the noise, there is a bias when using the maximum likelihood estimate away from the true values.

Profile likelihood of all transmission parameters

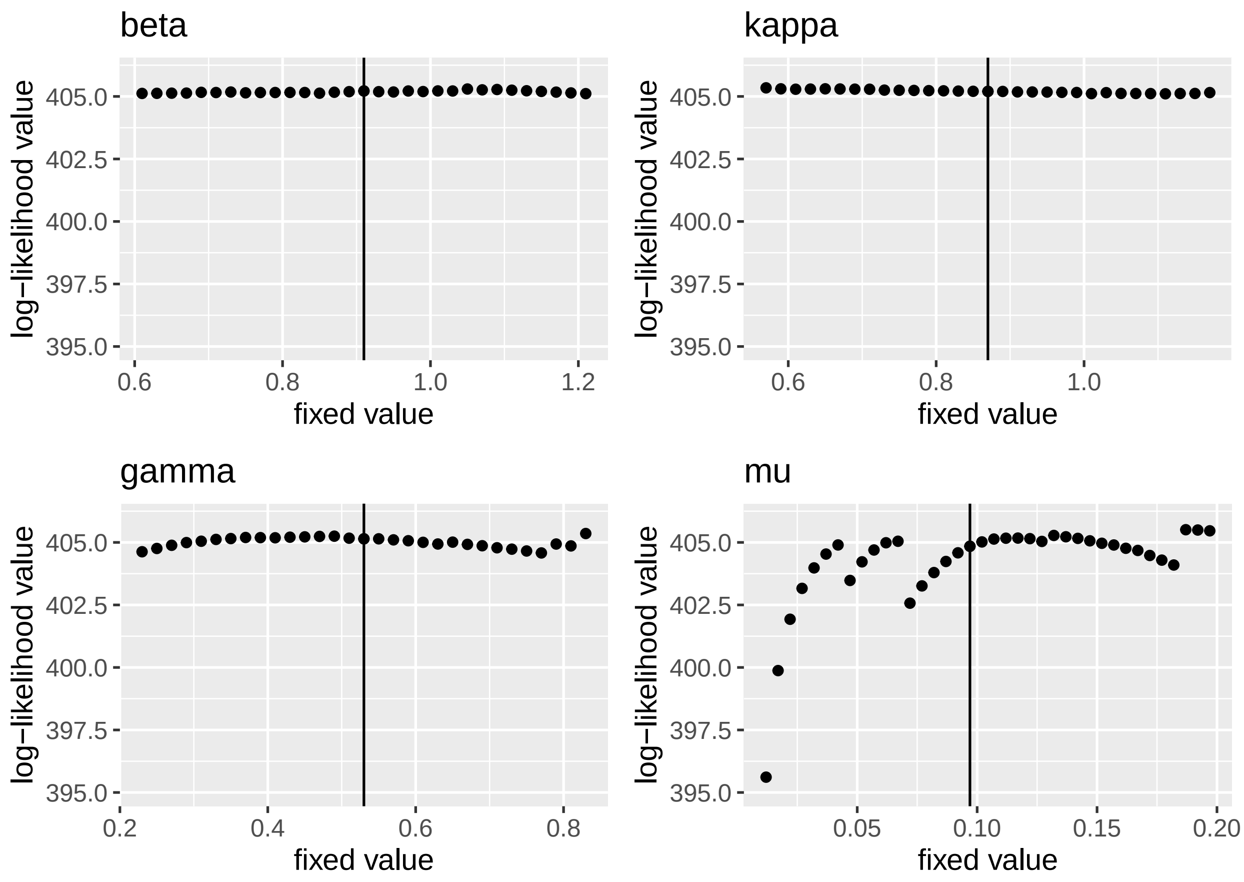
Here, we construct the profile likelihood by adding more unknown parameters into the optimisation. We will see how the number of parameters affects the identifiability of the transmission parameters of the SEIRD model. The same procedure is repeated with all the transmission parameters (instead of two out of the four): β, κ, γ and μ.

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

# Set range of values for interested parameters
beta_range <- c(seq(simulating_para$beta,
                    max(0.01, simulating_para$beta - 0.3),
                    by = -0.02),
                seq(simulating_para$beta,
                    simulating_para$beta + 0.3,
                    by = 0.02))
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))
gamma_range <- c(seq(simulating_para$gamma,
                     max(0.01, simulating_para$gamma - 0.3),
                     by = -0.02),
                 seq(simulating_para$gamma,
                     simulating_para$gamma + 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))
```

```r
# Run optimisations to create profile likelihood
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data_short,
                                            reltol = 1e-6)
# Save results
write.csv(profile_likelihood,
          "data/synthetic_halftrend_4param_profilelikelihood.csv",
          row.names = FALSE)
```

```r
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_halftrend_4param_profilelikelihood.csv")
```

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



These results show that all transmission parameters are not identifiable since there is not a clear maximum to the profile likelihood in each case, especially for β and κ, the profile likelihood is flat. However, in the case of μ, its profile likelihood is discontinuous. When the parameter μ is fixed, the log-likelihood value by fitting the number of deaths is fully determined. Since β, κ and γ are free to be optimised, we assume that the log-likelihood value from number of cases is the optimum. Thus, the discontinuity in the profile likelihood of μ could come from the noise added to the death data.

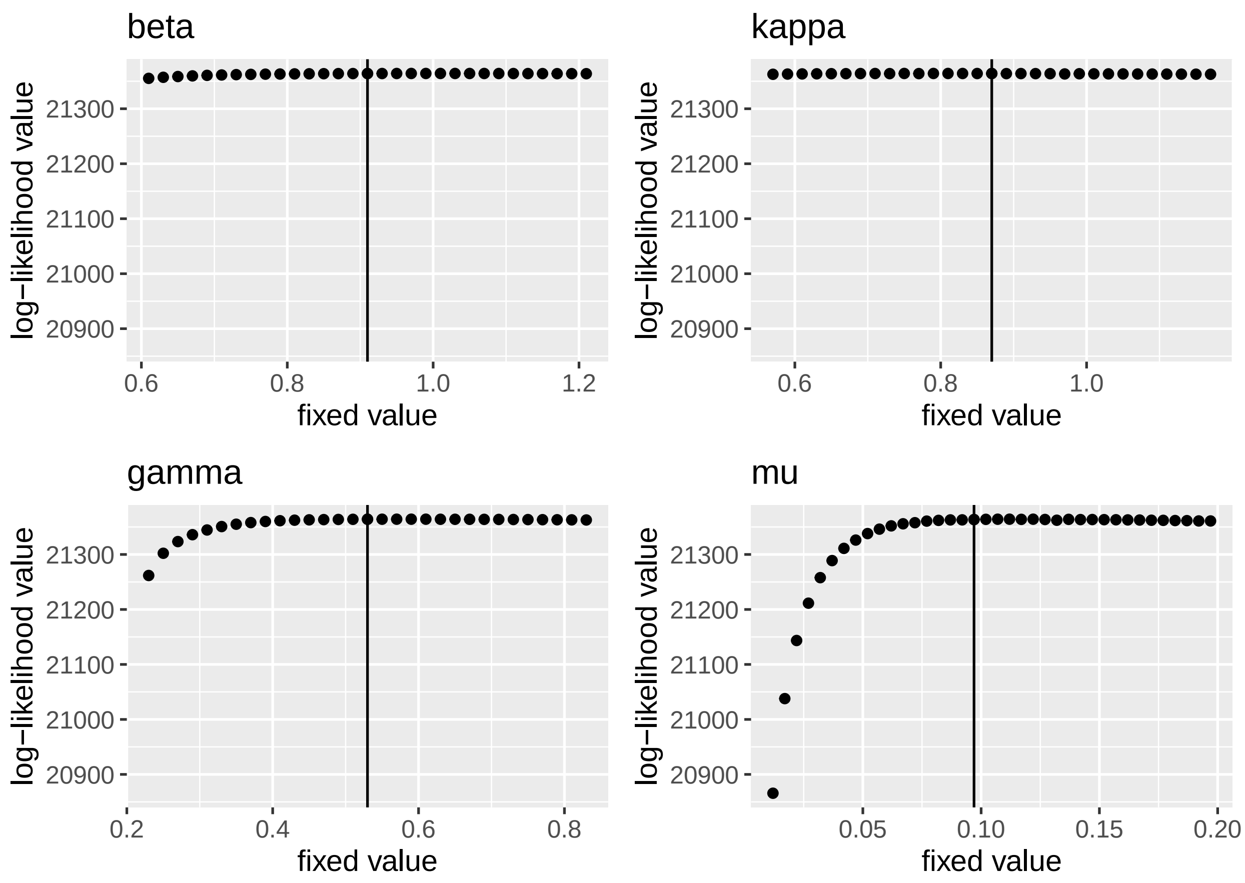
Profile likelihood with more information

When more information is provided to the optimisation, the identifiability issue could be improved. Here, we show an example by constructing the profile likelihood using synthetic data with time points that cover the first wave of the pandemic. To be exact, the time points taken start from the first incidence to day 150, an arbitrary value when the simulated data ends.

```r
testing_data_mid <- testing_data[-(1:first_nonzero_ind - 1), ,drop = FALSE]
profile_likelihood <- profilelikelihood_opt(profile_parameters,
                                            range_transmission,
                                            LogLikelihoodFn,
                                            testing_data_mid,
                                            reltol = 1e-6)

# Save results
write.csv(profile_likelihood,
          "data/synthetic_fulltrend_4param_profilelikelihood.csv",
          row.names = FALSE)
```

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

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



The figure above shows the profile likelihood of β, κ, γ and μ, with the same notations as the previous figure. However, although the identifiability is slightly improved for γ and μ, it is insufficientto accurately infer the parameters from the incidences and deaths data.

Profile likelihood of β, γ and μ

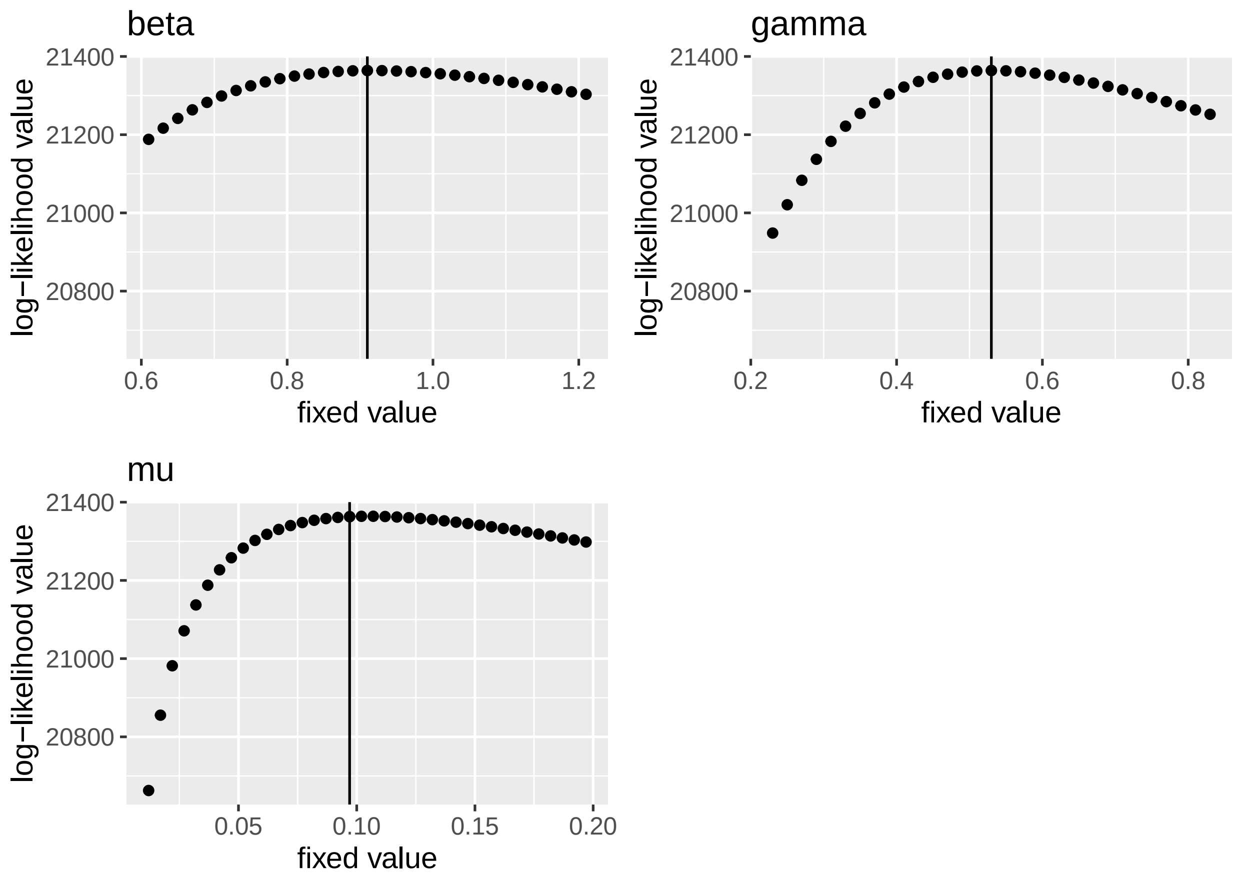
Since we are fitting the transmission parameters to incidences and deaths, there is lack of information to accurately identify κ and γ individually. To test the hypothesis, we plot profile likelihood for β, γ and μ, fixing κ to its true parameter value.

```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))
```

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

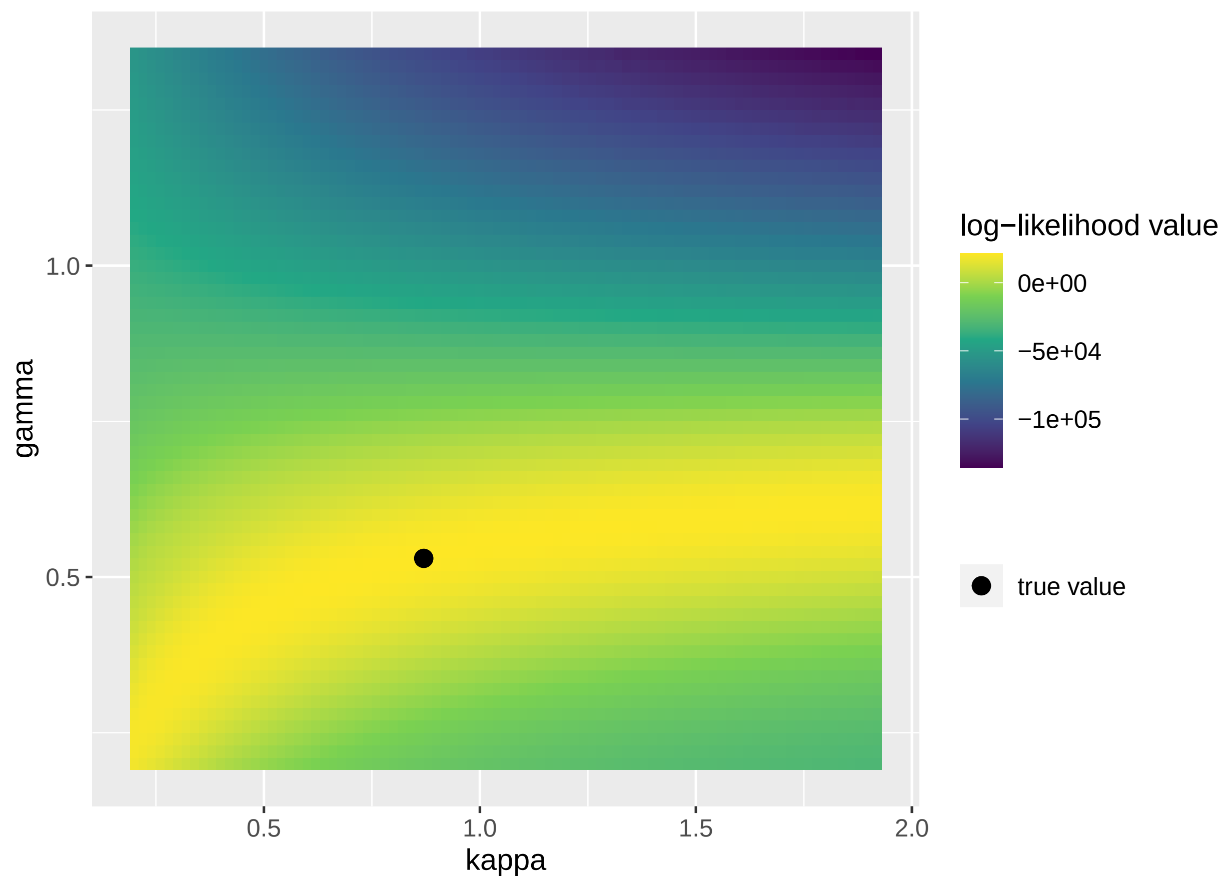
```
# Load results
profile_likelihood <- read.csv(
  "data/synthetic_fulltrend_3param_profilelikelihood.csv")
```

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



As shown in the profile likelihood graph, the parameters are identifiable.

```r
profile_parameters <- c('kappa', 'gamma')
kappa_range <- seq(0.2, 1.92, by = 0.02)
gamma_range <- seq(0.2, 1.34, by = 0.02)
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_range) {
  for (gamma in gamma_range) {
    parameters <- list(kappa = kappa, gamma = gamma)
    ll_value <- LogLikelihoodFn(parameters, model = SEIRD(),
                                inc_numbers = testing_data_mid$IncNoise,
                                death_numbers = testing_data_mid$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_tile(data = loglikelihood_values,
            aes(x = kappa, y = gamma, fill = likelihood_value)) +
  scale_fill_viridis_c() +
  geom_point(data = true_parameters,
             aes(x = kappa, y = gamma, size = "true value")) +
  labs(fill = "log-likelihood value", size = "") +
  scale_size_manual(values = c(3)) +
  theme(text = element_text(size = 12),
        legend.text = element_text(size = 10))
```



The graph shows the log-likelihood values for different combinations of κ and γ, with the black point indicating the true values of the parameters. As observed from the graph, κ and γ are strongly correlated. Region neighbouring the true value shows large area where combinations of these two parameters gives equally high log-likelihood values. Thus, when optimisating for all transmission parameters, the parameters are unidentifiable.

Conclusion

We have shown that there are identifiability issues with the parameters in the SEIRD model by showing that multiple sets of parameters can fit the SEIRD model to the pre-lockdown London COVID-19 data. This lack of model identification led to a diversity of forecasted scenarios for cases and deaths and illustrates the importance of quantifying modelled uncertainty when using transmission dynamics models to inform policy.

Additionally, we have shown how simulated data and the profile likelihood method can be used to assess the identifiability of models in the absence of real data. Since such analyses can highlight the insufficiency of data and the existence of non-inferable parameters for estimating model parameters, we argue these approaches should be pursued as precursors to performing inference on real data.

References

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

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, John A, and Roger 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/.