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 unidentifiable given a set of observations.

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.

In this vignette, we first use numerical optimisation to perform an analysis using real data from the initial COVID-19 wave of 2020 in London. Next, we use profile likelihood methods to formally determine the 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 the identifiability 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

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

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

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

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

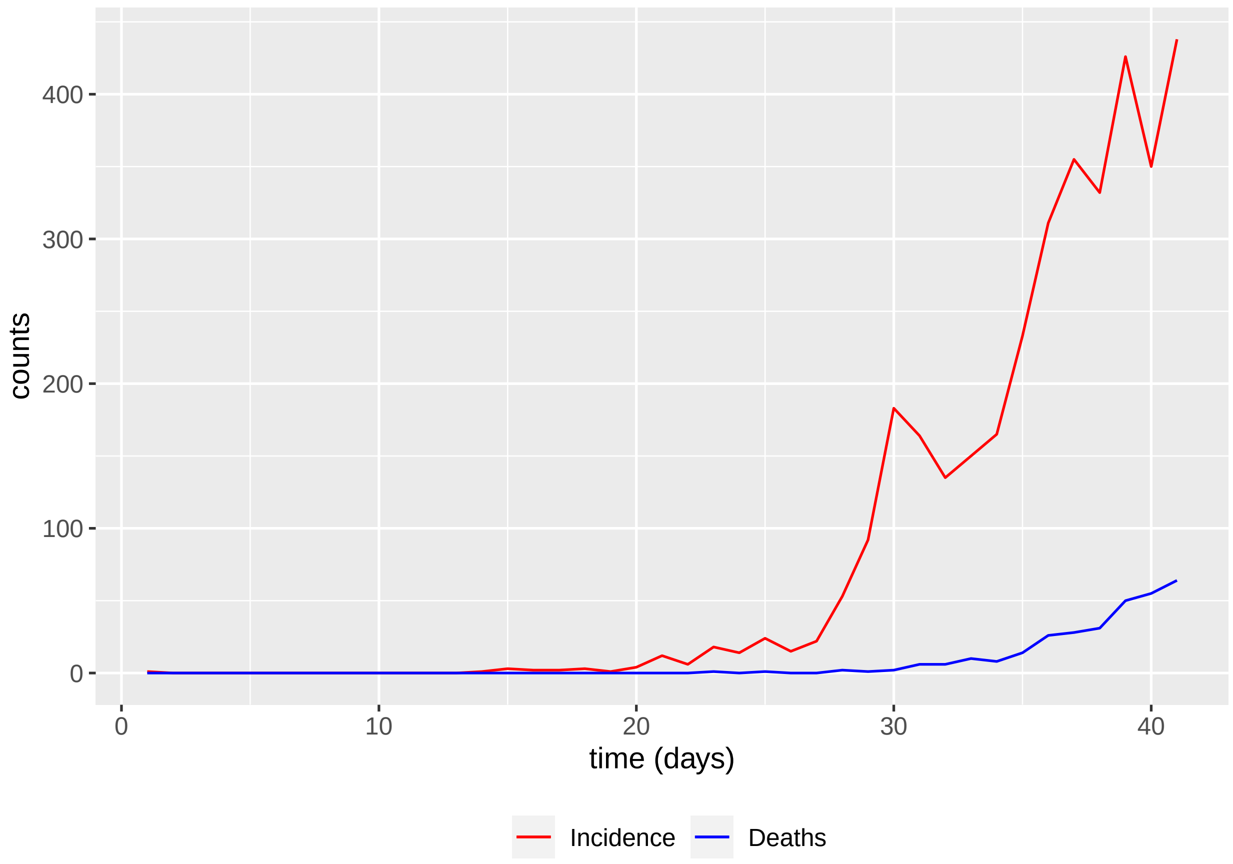
where β, κ, γ and μ are the infection rate, incubation rate, recovery rate and death rate respectively. More details can be found in the SEIRD vignette from the `comomodels` package.

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

We begin by fitting the SEIRD model to real COVID-19 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 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 = "counts") +
  theme(text = element_text(size = 12), legend.position = "bottom",
        legend.text = element_text(size = 10))
```



To estimate the parameters, we use maximum likelihood estimation. Here, we assume observed cases (C_t^*) and deaths (D_t^*) are sampled from Poisson distributions with the true cases (C_t) and deaths (D_t) as the respective means:

$$C_t^* \sim \mathrm{Pois}(C_t),$$
$$D_t^* \sim \mathrm{Pois}(D_t).$$

Note, that the above implicitly assumes that there is 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. In what follows, we stack the cases and deaths data into a vector, $\boldsymbol{x} = (C_t^*, D_t^*)$.

$$p(\boldsymbol{x}|\beta, \kappa, \gamma, \mu) \propto \left(\prod_{t=1}^{T} D_t^{D_t^*} \exp(-D_t)\right) \left(\prod_{t=1}^{T} C_t^{C_t^*} \exp(-C_t)\right),$$

where C_t and D_t are both functions of the transmission parameters. Note that, in the above, we have dropped the constant of proportionality for readability.

The initial conditions are assumed fixed at:

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

which represent the result of an optimisation across all model parameters (scripts not shown).

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

We aim to determine the maximum likelihood estimates of the four transmission parameters: β, κ, γ and μ. We use a Nelder-Mead optimiser (Nelder and Mead 1965) to find these estimates. 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 four transmission parameters. Note that, in the below, `RealData_LogLikelihoodFn` is a function defined in `SEIRD_optimisation.R` that returns the log-likelihood for the cases and deaths data.

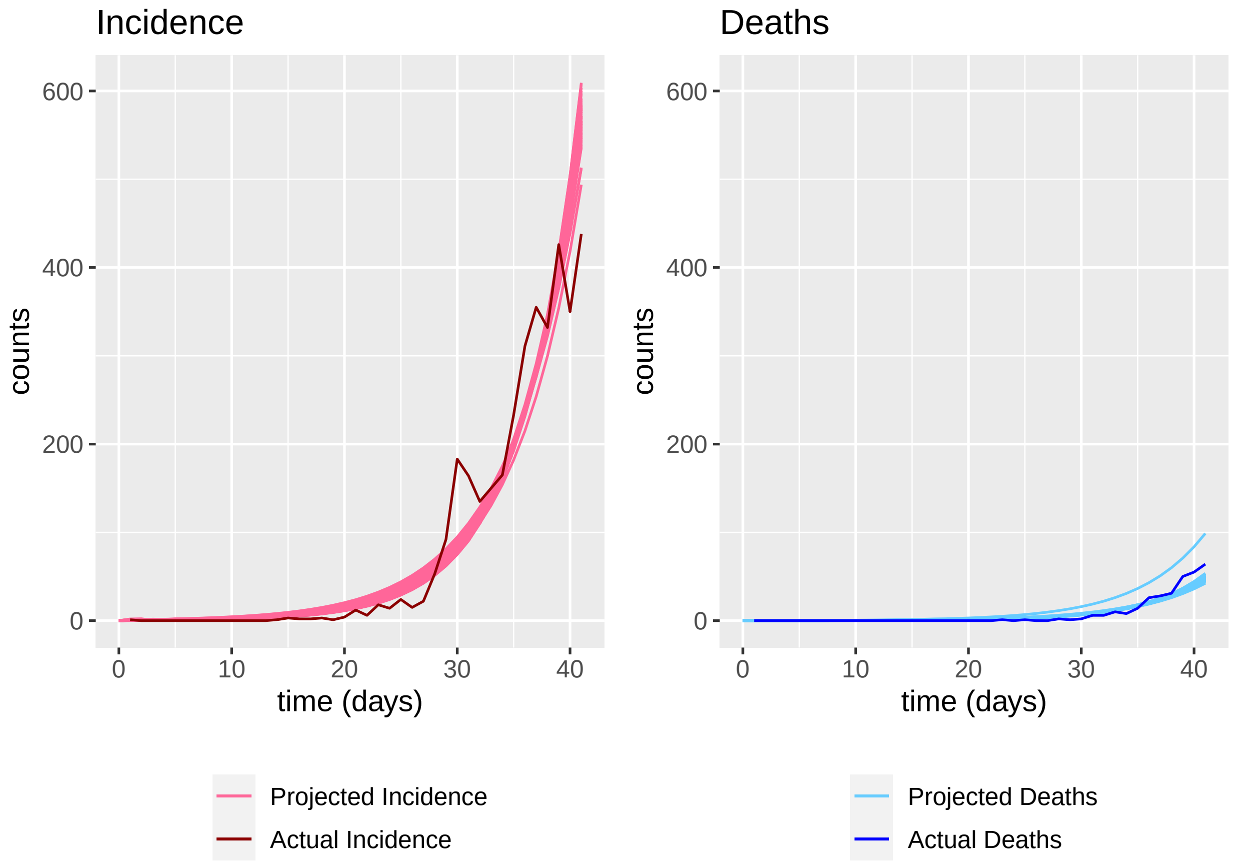
```
# Set up conditions for optimisation
constraint_ui <- rbind(diag(4))
constraint_ci <- c(rep(0, 4))
repeat_num <- 100

# 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)
```

Or, alternatively, we can load results that were produced by a previous run.

```
# 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.

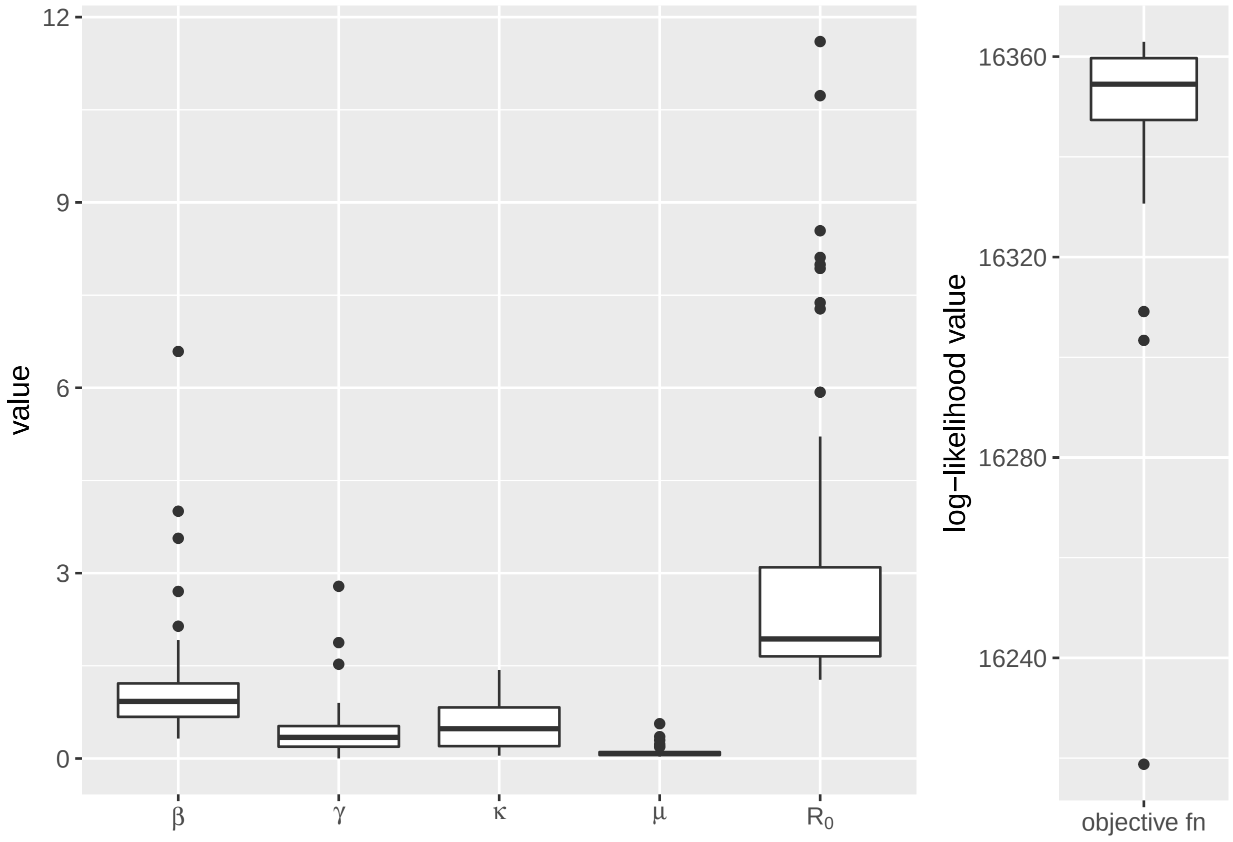
```
data_optimisation_plot(optimised_para, London)
```



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",
```

```r
               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))
```



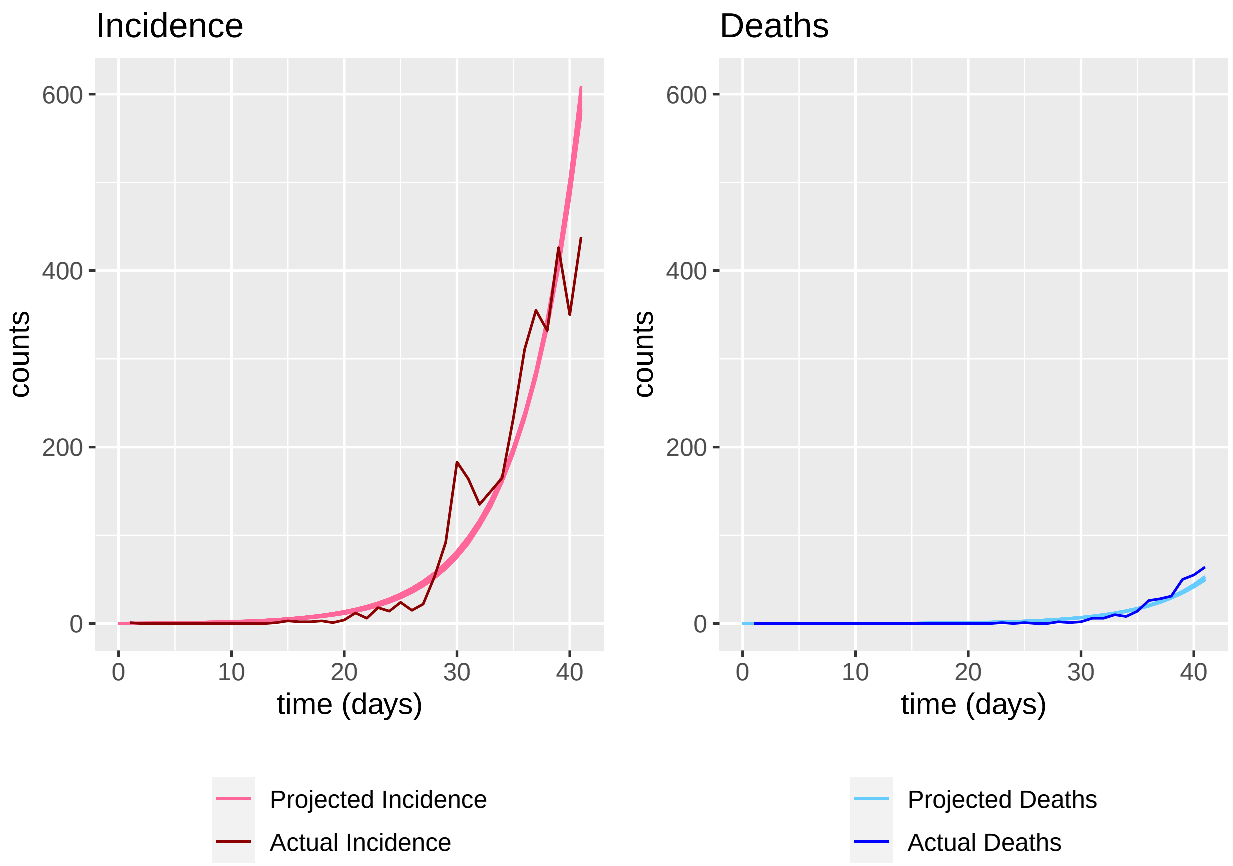
```r
# order according to log-likelihood
opt_df <- optimised_para[order(-optimised_para$obj_fn),]
```

The variation in parameter sets shown in the above plot illustrates that some of the optimisation runs clearly became stuck before reaching the true maximum, 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 variation 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, immunity in the population was likely negligible, and the behaviour of individuals was less affected than later on during the epidemic. These characteristics greatly simplify the task of estimating R_0, since the models used to fit available data 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). There are many explanations for this difference, most likely reflecting the failure of the basic SEIRD model to include key processes, such as asymptomatic or presymptomatic transmission, stratification of cases according to disease severity and age-structure.

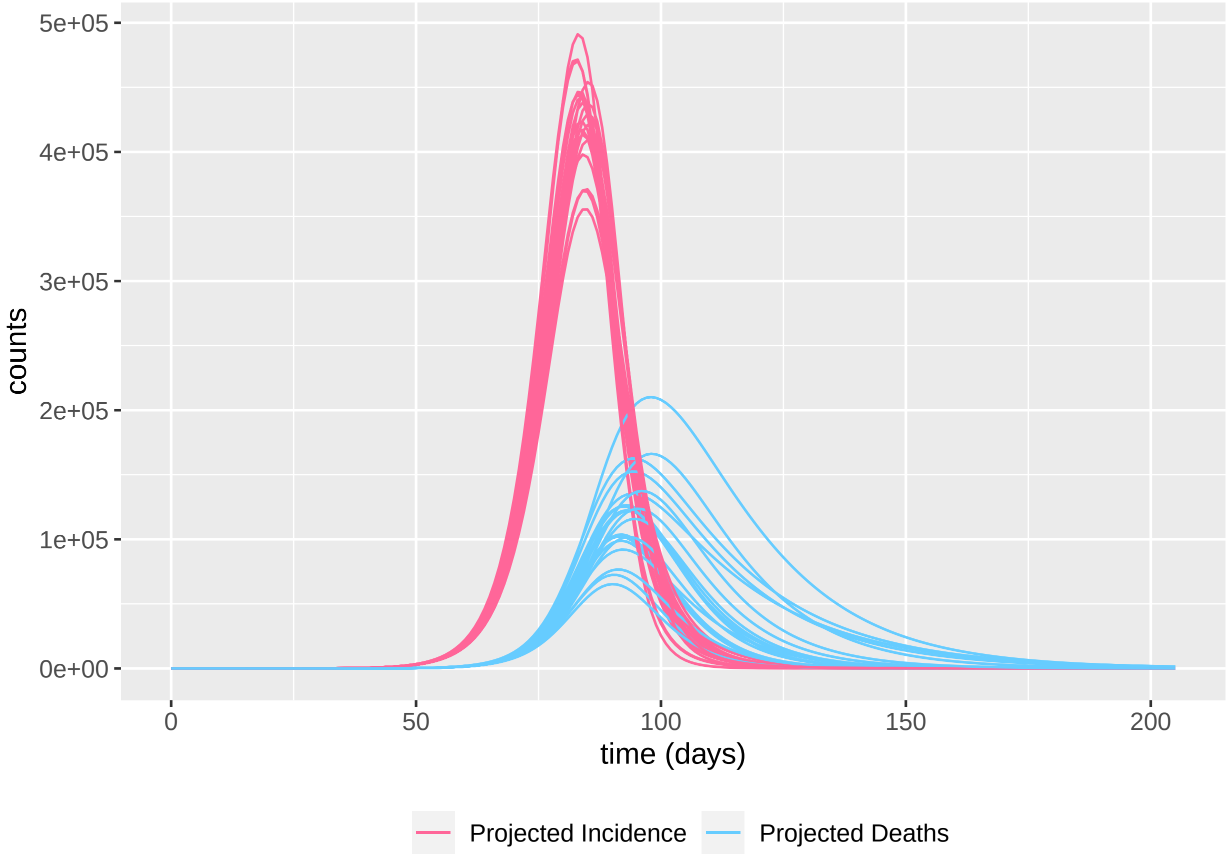
To provide a heuristic measure of uncertainty in our model fits, we chose the 20 optimisation parameter sets with the highest log-likelihood values. We recognise that a more formal uncertainty analysis like that obtained from Bayesian inference (see, for example, Gelman et al. (2013)) would yield a more theoretically grounded measure of uncertainty. The main point of our analysis, however, is to illustrate that a number of different 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
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. Here, we plot the cases and deaths series representing the mean of the Poisson distribution.

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



The figure shows that the 20 parameter sets produced a wide variety of possible epidemics, despite exhibiting very similar behaviour early on. Indeed, here, the maximum difference in peak values of all projections is more than 1×10^5 for both cases and deaths. This result indicates the importance of including measures of uncertainty when making projections over the eventual course of the epidemic.

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 contain insufficient information to estimate the true parameters. So, we now perform a simulated data study, where we generate simulated series for cases and deaths 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.

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

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

# Set up SEIRD model
```

```r
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 raw ODE model outputs do not generate the random fluctuations seen in the real data, and we use a Poisson distribution to generate simulated series for cases and deaths. To do so, we assume that the means for cases and deaths are the model outputs for each of these quantities:

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

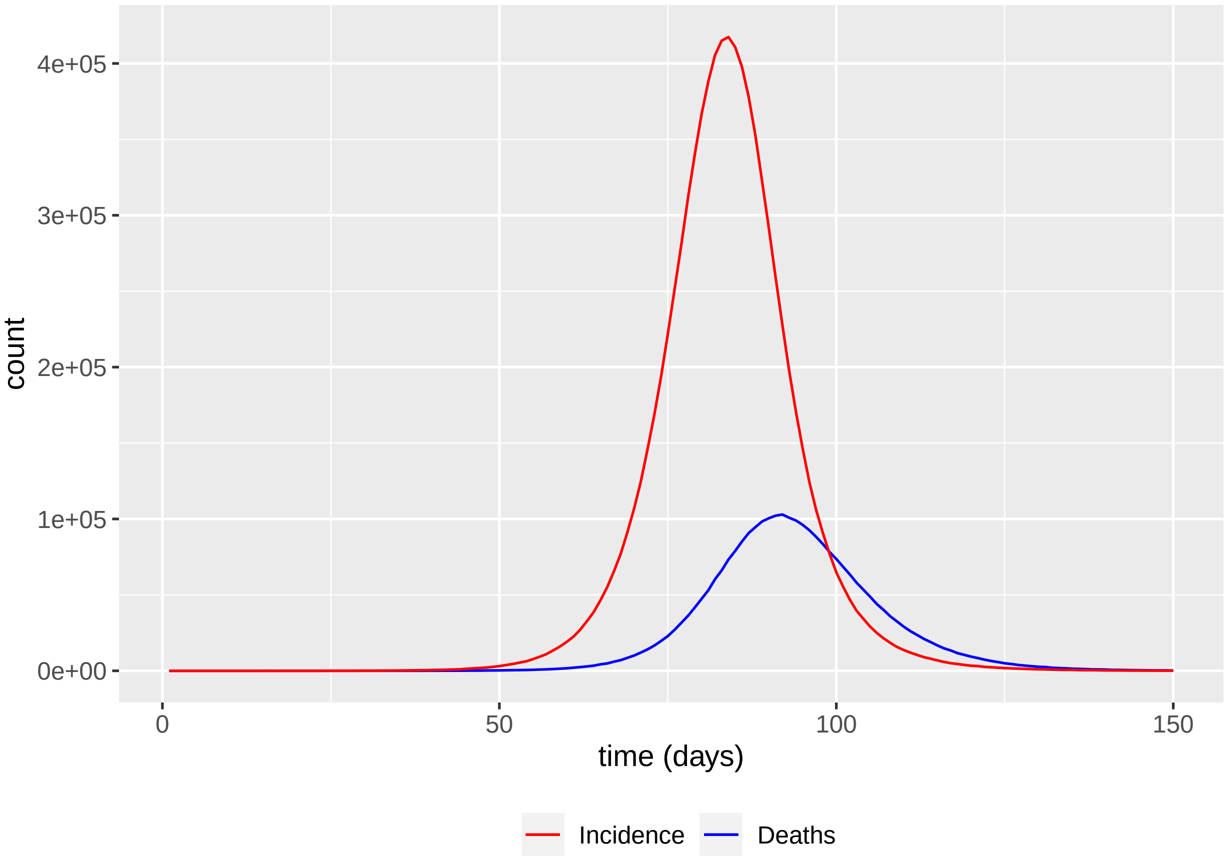
where $C(t)$ and $D(t)$ are cases and deaths from the raw ODE model outputs respectively, while $C^*(t)$ and $D^*(t)$ are noisy cases and deaths of the SEIRD model.

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

We then plot these simulated data, which shows a complete epidemic wave.

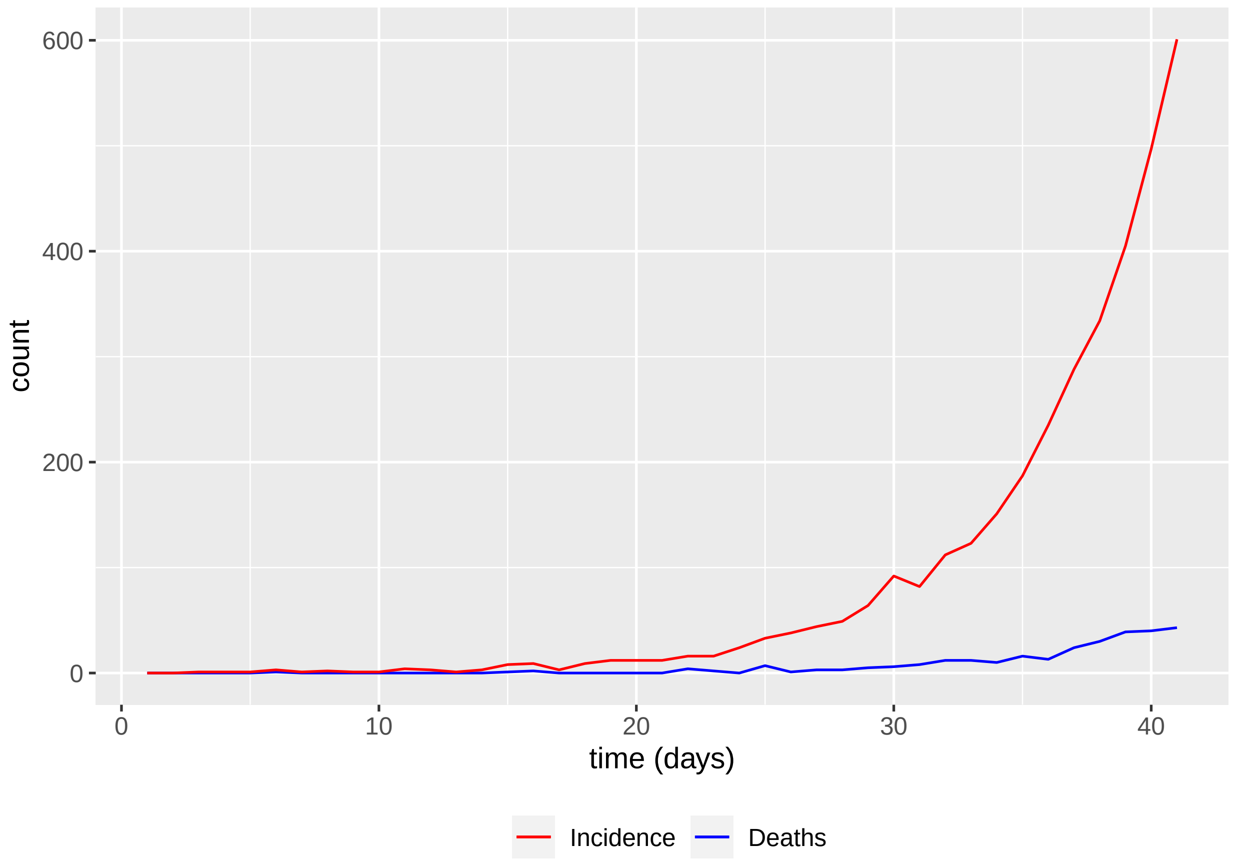
```r
ggplot() +
  geom_line(data = testing_data,
            aes(x = time, y = DeathNoise, color = "Deaths")) +
  geom_line(data = testing_data,
            aes(x = time, y = IncNoise, color = "Incidence")) +
  scale_color_manual(name = "",
                     values = c("Incidence" = "red", "Deaths"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



Profile likelihood of β and γ: early epidemic phase

We initially use a subset of the above data which corresponds to time series for cases and deaths which are of the same length as the real set we considered above for London. This subset of the original dataset is visualised in the figure below.

```
# 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 initially suppose that only β and γ are unknown and that all other parameters are fixed at their true values. To determine the profile likelihood for γ, we consider a grid of γ values, and, in each case, we use a Nelder-Mead optimiser (Nelder and Mead 1965) to find the value of β that maximises the profile log-likelihood:

$$g(\hat{\gamma}) = \max_{\beta} \log p(\boldsymbol{x}|\beta, \hat{\gamma}).$$

Here, $g(\beta|\hat{\gamma})$ denotes the profile likelihood function, which gives the value of a log-likelihood function that is maximised over choice of β conditional on $\gamma = \hat{\gamma}$. By considering a discrete (and finite) set of γ values, we obtain a trace that hopefully well approximates this function.

We then follow the same procedure to obtain (an approximation to) the profile likelihood for β, which is the function:

$$h(\hat{\beta}) = \max_{\gamma} \log p(\boldsymbol{x}|\hat{\beta}, \gamma).$$

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 parameter values that were used to generate the data.

```
# 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))
```

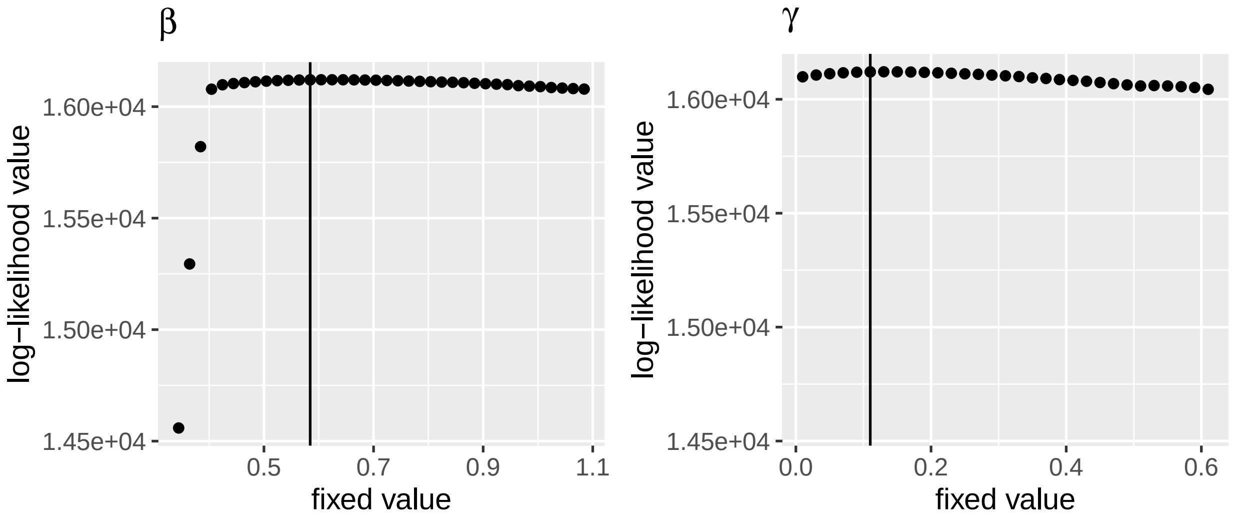
We now use optimisation to trace out the profile likelihoods for both β and γ. The function is defined in the separate `SEIRD_optimisation.R` script for readability. In performing optimisation, we do so in a loop over the grid of values for each of β and γ. For the first grid values, we use the true parameter values as the initial points for the optimiser; for latter grid values, we use the last obtained optima.

```
# 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)
```

```
# 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.

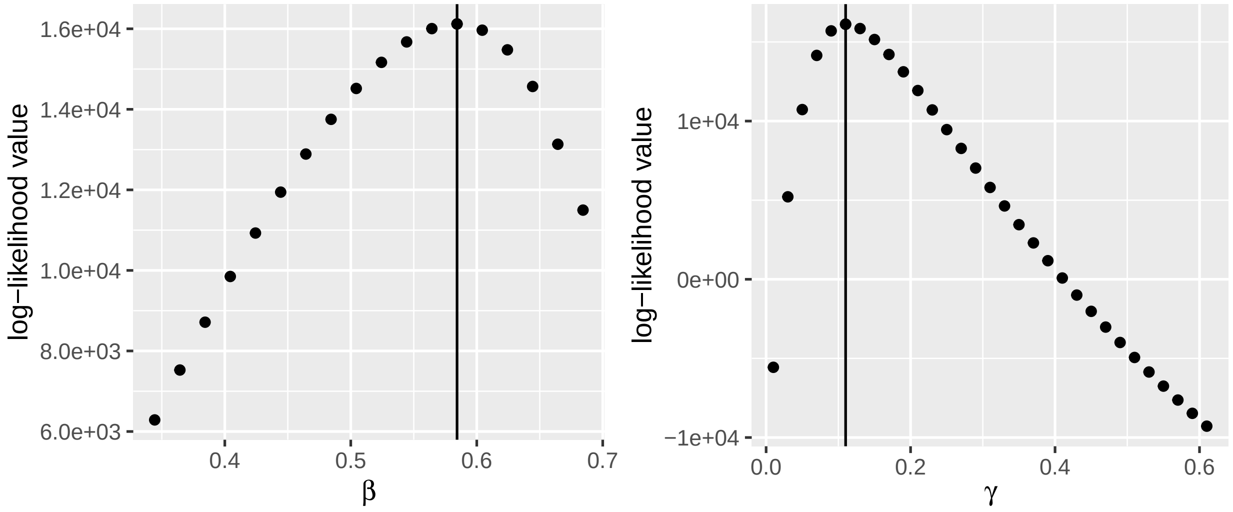
```
profilelikelihood_plot(profile_likelihood, profile_parameters)
```



We next consider an easier inference problem, when all of the parameters except the parameter under consideration are fixed at their known values. In the figure below, in the left panel, we plot 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 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 of β and γ: full epidemic wave

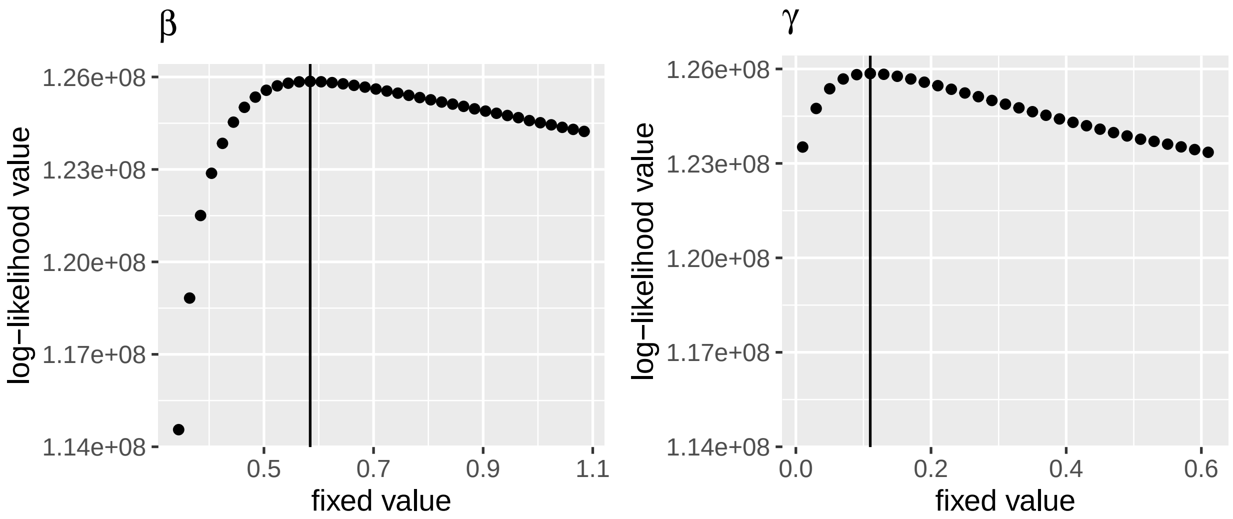
If more data are collected, it may be possible to better identify a model's parameters. Here, we suppose that, as before, β and γ have unknown values, and we have time series for cases and deaths that encompasses the entire epidemic wave.

We first generate profile likelihoods for β and γ using this larger dataset.

```
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)
```

```
# 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. 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, using a dataset which encompasses the entire epidemic wave has helped us to identify the parameter values.

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 calculate then plot a profile likelihood assuming that we have data for the entire epidemic wave.

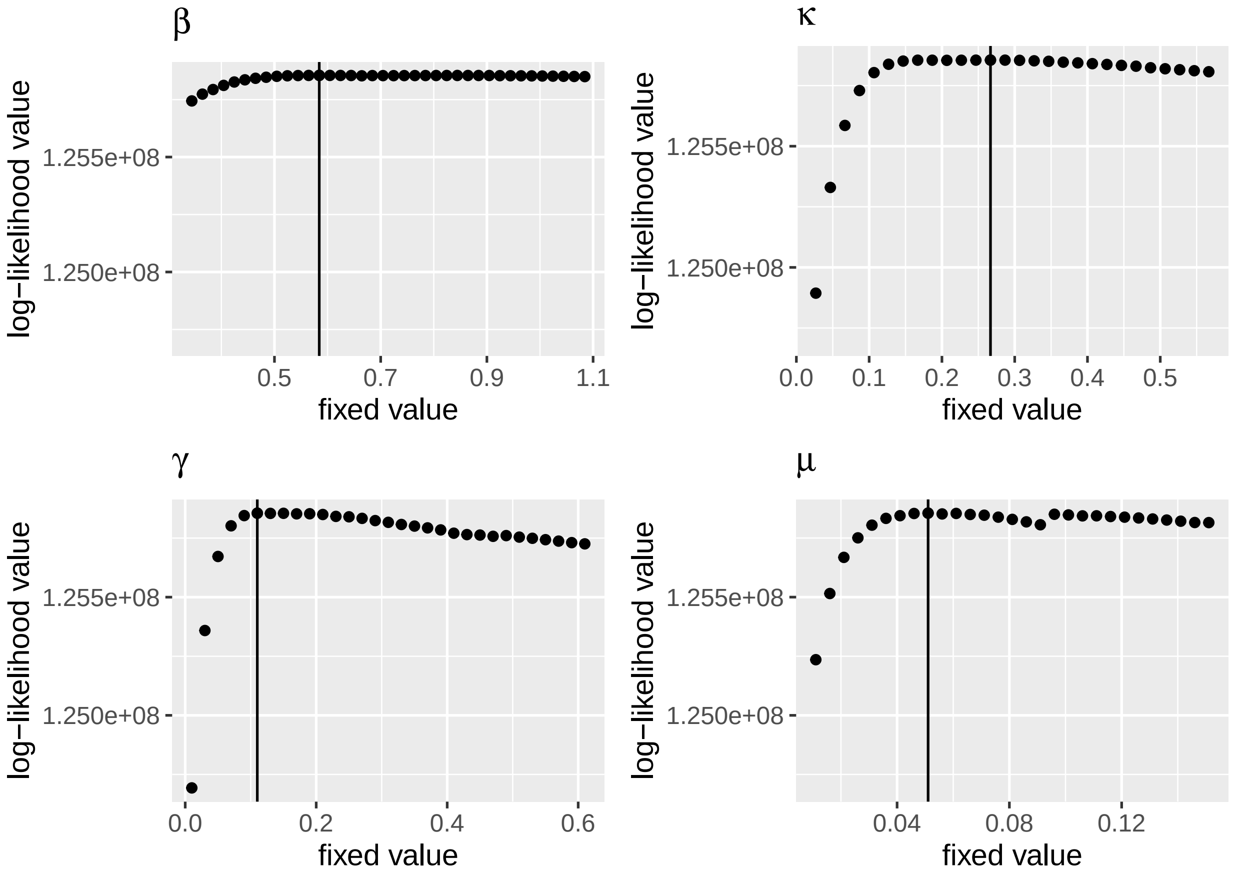
```r
# Set interested parameters for profile likelihood
profile_parameters <- c('beta', 'kappa', 'gamma', 'mu')

# 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)
```

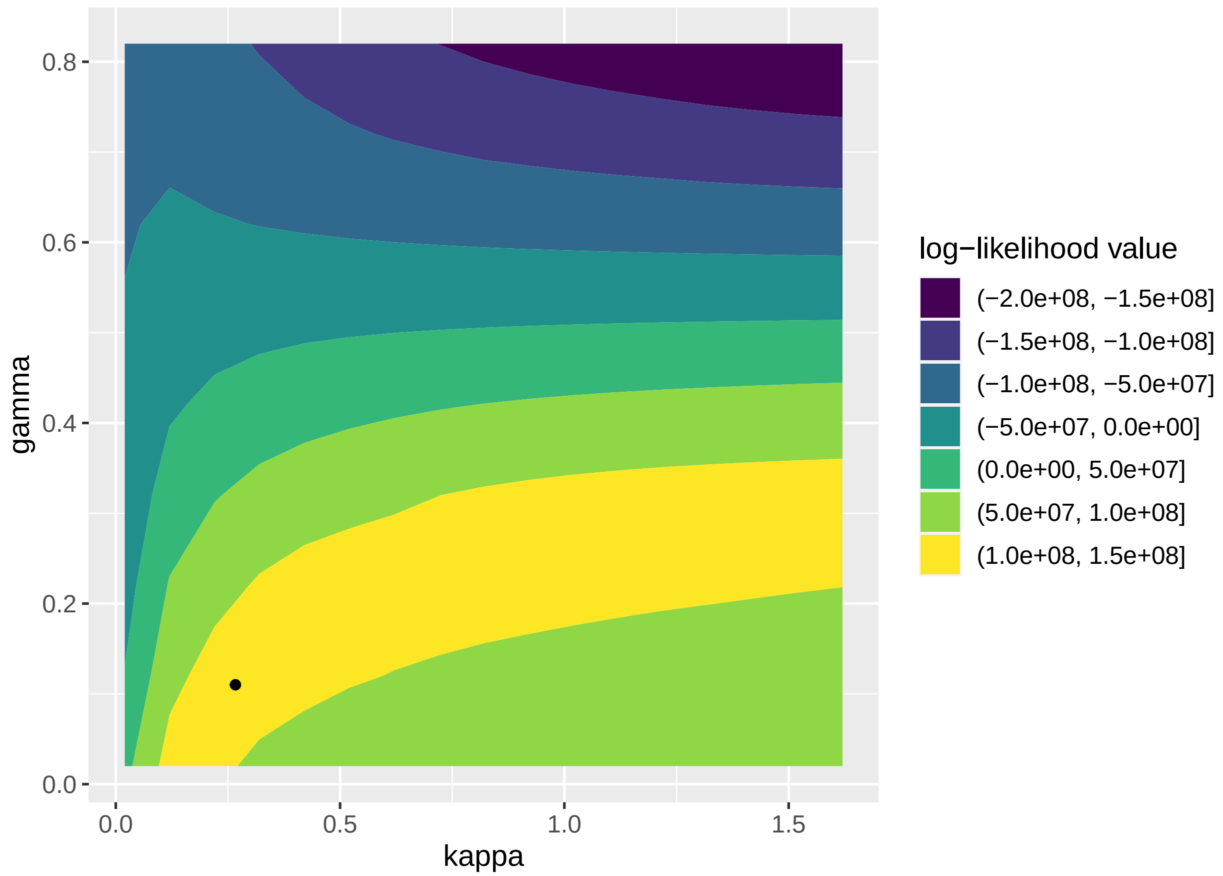


For all four parameters, the profile likelihoods are not strongly peaked near the true parameter values, indicating identification problems. This is because, in a model with more free parameters, there will generally be more combinations of parameter values that yield similar fits to the data. A result of this is that the profile likelihood curves for each of the parameters are wider.

Reducing the number of free parameters

A higher value of κ, implying a shorter latent period, means that an epidemic will appear sooner and will have a higher peak number of infectious individuals. This could be partly offset by an increased value of γ, meaning that individuals spend less time being infectious, 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 (see below), explaining why it is difficult to estimate both of these parameters from data. In the plot below, we use a black point to indicate the true parameter values.

```r
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))
```



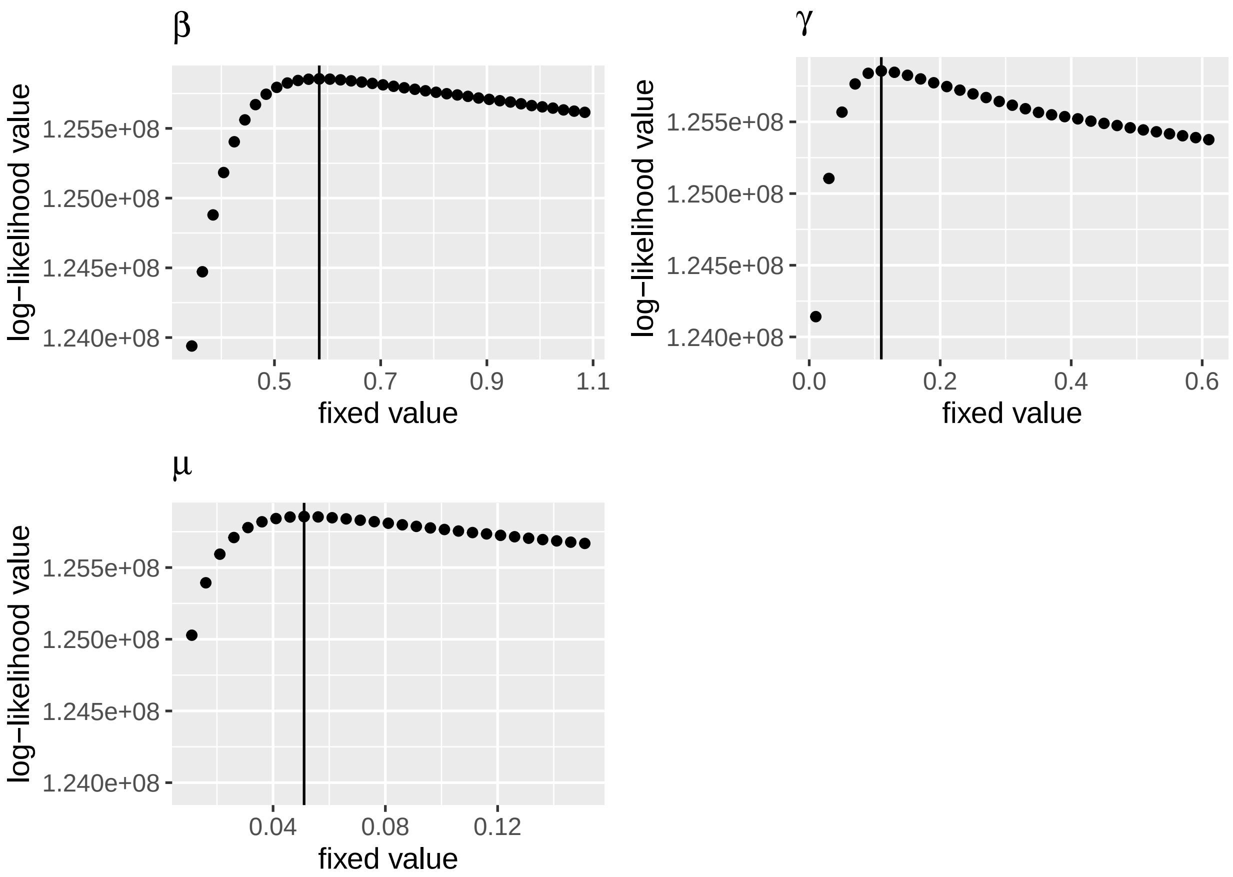
Because κ and γ cannot uniquely be identified by the cases and deaths data, we now fix κ at its true value and repeat the profile likelihood analysis for the remaining free parameters: β, γ and μ.

```
# 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,
                                            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)
```



With only β, γ and μ unknown, the profile likelihoods are more peaked nearer the true parameter values, indicating that it may be possible to estimate each of the parameters. This example shows that, by reducing the number of parameters being estimated, we may be better able to recover the remaining unknown parameters.

Conclusion

Using real data on disease cases and deaths from before the peak of the 1^{st} epidemic wave of COVID-19 in London, we showed that many parameter sets from the SEIRD model were compatible with the data. Next, using data simulated from the SEIRD model, we showed that the model was not practically identifiable given early epidemic data. Further, even if data on a complete (and unmitigated) epidemic were available, the model remains unidentifiable. This identifiability arises 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 – for instance, studies of patients admitted to hospital with COVID-19. These estimates can then be incorporated into the inference process in one of two ways: either these parameters are assumed known and are fixed at the point estimates obtained in other studies; or, if using a Bayesian approach to inference, strongly informative priors can be used that encompass this outside information (see, for example, Gelman et al. (2013)).

Overall, our results show that performing inference for transsmission dynamics models can be plagued with identifiability issues. We exemplified this using the relatively simple SEIRD model, which is the simplest in our package and has substantially fewer parameters than the complex models that have often been used to inform public health policy. Thus, it is generally not possible to infer all the parameters of these larger models, since many combinations of the parameter values can yield similar fits to the data. As a result, these models rely heavily on external studies to inform their parameter values. Sensitivity analyses can be used to assess whether the conclusions of modelling depend strongly on the chosen set of parameter values. Thus, sensitivity analyses are crucial to perform and to present when using models to inform policy.

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.”

Gelman, A, JB Carlin, HS Stern, DB Dunson, A Vehtari, and DB Rubin. 2013. *Bayesian Data Analysis*. CRC press.

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/.