Parameter estimation

Once we know how to solve an ODE with known parameters, the next step is to infer the model directly from the data.

Setting up

Let's load the necessary libraries:

```
library(dplyr)
library(ggplot2)
library(knitr)
library(deSolve)
library(bbmle)
```

For this exercise we will use data from a flu epidemic in an English boys schoolboard (763 boys in total) from January 22nd, 1978 (day 0) to February 4th, 1978 (day 13).

The first step is to recreate the function to perform the ODE solving.

```
sir_fun <- function(beta, gamma, S0, I0 = 1 - S0, R0 = 0, times, N = 1) {

  # the differential equations:
  sir_equations <- function(time, variables, parameters) {
    with(as.list(c(variables, parameters)), {
      dS <- -beta * I * S
      dI <-  beta * I * S - gamma * I
      dR <-  gamma * I
      return(list(c(dS, dI, dR)))
    })
  }

  # the parameters values:
  parameters_values <- c(beta  = beta, gamma = gamma)

  # the initial values of variables:
  initial_values <- c(S = S0, I = I0, R = R0)

  # solving
  out <- ode(initial_values, times, sir_equations, parameters_values)

  # returning the output:
  as.data.frame(out) %>%
    mutate(across(-time, ~ .x * N)) # Scale to population
}
```

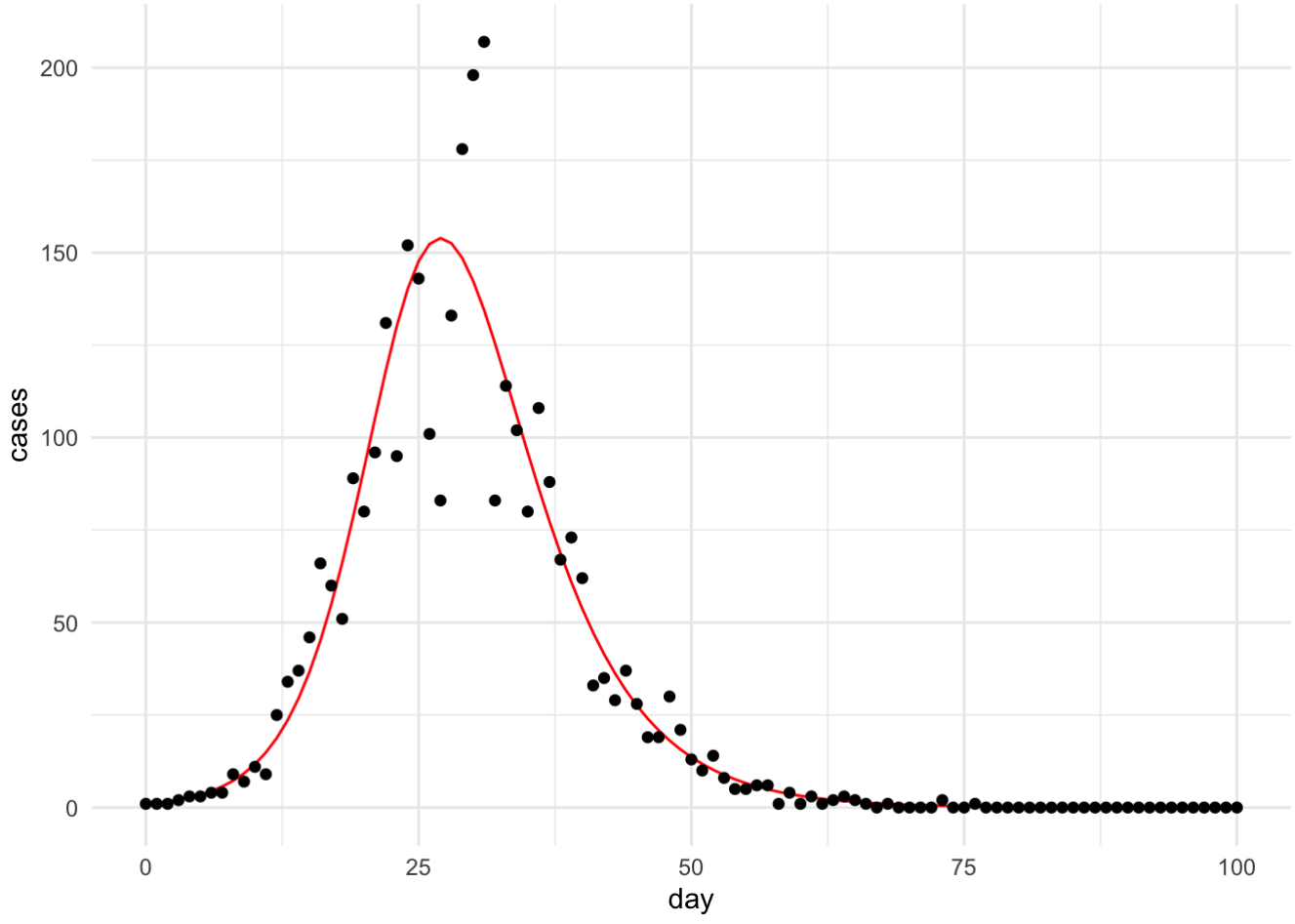
Data simulation

To simulate the data we generate an epidemic with known parameters plus some noise. The R_0 in this situation is 2.

```
N = 1000

set.seed(1270129109) # for reproducibility

# Theoretical trajectory given beta and gamma parameters
sir_data_theor <- sir_fun(0.5, .25, S0 = .999, times = seq(0, 100)) %>%
  transmute(day = time, cases = I * N)

# Observed data after adding noise
sir_data_obs <- mutate(sir_data_theor, cases = rnbinom(n(), mu = cases, size = 20))

# Forcing the introduction of 1 case in the population
sir_data_obs$cases[1] <- 1
```

Let's see how does it look:

```
ggplot() +
  geom_line(data = sir_data_theor, mapping = aes(day, cases), color = 'red') +
  geom_point(data = sir_data_obs, mapping = aes(day, cases)) +
  theme_minimal()
```



Next, we need to define a "Likelihood function" or "Score function" or "Objective function", which estimates the outbreak curve for each combination of beta and gamma parameters and evaluates how well it fits the data.

```
LLpois <- function(beta, gamma, day, cases, N) {
  # the optimizer works with values from -Inf:+Inf but SIR parameters can only
  # be positive (R+)
  beta <- exp(beta)
  gamma <- exp(gamma)

  I0 <- cases[1] # initial number of infectious
  observations <- cases[-1] # the fit is done on the other data points

  # generate one outbreak curve given the parameters
  predictions <- sir_fun(beta = beta, gamma = gamma,
                         S0 = 1 - I0/N, I0 = I0/N, R0 = 0, times = day, N)
  predictions <- predictions$I[-1] # removing the first point too

  if (any(predictions < 0)) return(NA) # removing impossible estimates

  # returning minus log-likelihood. Sum of logs is easier to manage than the
  # product of raw likelihoods
  -sum(dpois(x = observations, lambda = predictions, log = TRUE))
}
```

Using an “Optimizer” we are able to find the best set of parameters for the objective function that better describe the data:

```
starting_param_val <- list(beta = 0.3, gamma = 0.3) # some starting parameters

estimates <- mle2(
  minuslogl = LLpois, # the objective function to optimize
  start = lapply(starting_param_val, log), # starting values on log scale
  fixed = list(N = 1000), # the population size is considered fixed
  method = "Nelder-Mead", # an optimization algorithm
  data = sir_data_obs # the simulated data
)
```

Which gives us the following estimated parameters:

```
# the parameters were still on log scale.
# The third parameter is N which is fixed.
best_params <- exp(coef(estimates))[1:2]

print(best_params)
```

```
##      beta     gamma
## 0.4993768 0.2513257
```

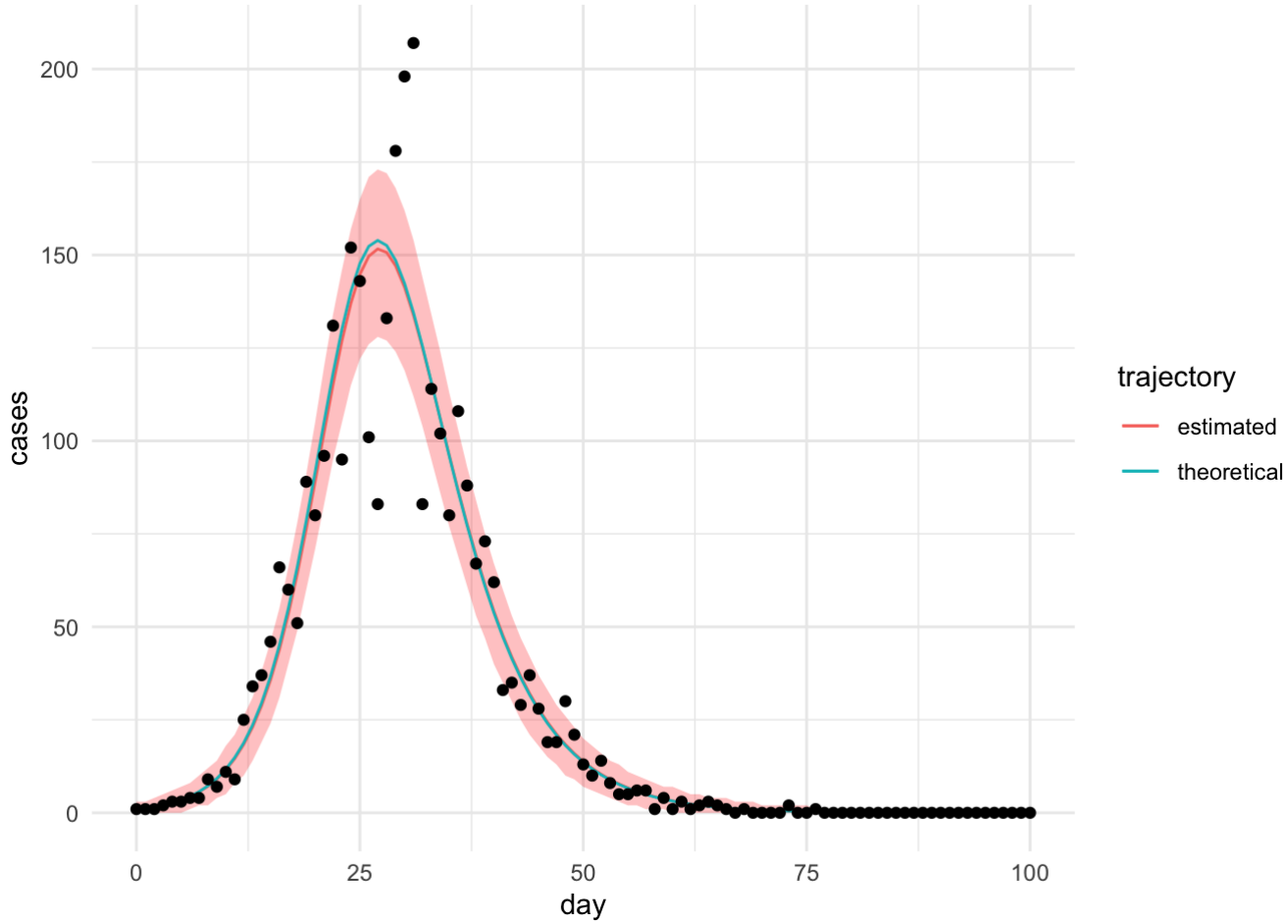
with an R_0 of:

```
print(as.vector(best_params['beta'] / best_params['gamma']))
```

```
## [1] 1.986971
```

Let's see now how these parameters compare with the original data and the theorerical outbreak curve:

```
# Generate the outbreak trajectory given the estimated parameters
best_predictions <- sir_fun(best_params["beta"],
                            best_params["gamma"],
                            S0 = 1 - sir_data_obs$cases[1]/N,
                            times = with(sir_data_obs, min(day):max(day)),
                            N = N)

ggplot(best_predictions, aes(x = time)) +
  # use a Poisson distribution to estimate the expected range of observed cases
  # at 95% probability
  geom_ribbon(aes(
    ymin = qpois(0.025, I),
    ymax = qpois(0.955, I)), fill = 'red', alpha = .25) +
  # plot the estimated trajectory
  geom_line(aes(y = I, color = 'estimated')) +

  # plot the theoretical trajectory
  geom_line(
    data = sir_data_theor,
    mapping = aes(day, cases, color = 'theoretical')) +

  # plot the observed data
  geom_point(data = sir_data_obs, mapping = aes(day, cases)) +
  theme_minimal() +
  labs(x = 'day', y = 'cases', color = 'trajectory')
```


200
150
100
50
0
cases
0
25
50
75
100
day
trajectory
estimated
theoretical