Introduction to Modeling Infectious Disease:
Basic Data Fitting

Lauren M. Childs
Associate Professor, Mathematics
Virginia Tech

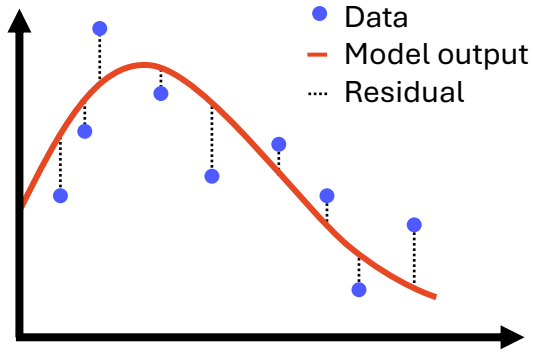
Omar Saucedo
Assistant Professor, Mathematics
Virginia Tech

Goal for data fitting

Generate a model that matches the data.

Given a data set and a model that represents the data,

how can we find the “best” parameters that illustrate the data set?



Find the parameters that give the smallest residuals.

Basic data fitting approach

A basic approach for finding the optimal parameters is the least-squares method.

Given a data set with n data points given by

$$\{(t_1, y_1), \ldots, (t_n, y_n)\},$$

and n model outputs measured at the same time points given by

$$\{(t_1, \hat{y}(t_1)), \ldots, (t_n, \hat{y}(t_n))\}.$$

The sum of squared error is

$$\text{SSE} = \sum_{j=1}^{n} \left(y_j - \hat{y}(t_j)\right)^2$$

To obtain the “best” parameters, minimize the SSE.

Basic data fitting approach

1. Formulate the mathematical model to describe the data.
2. Choose an initial set of parameters.
3. Simulate the model across some time period.
    a. Build and use a function to produce model output.
4. Record the output of the model at the time points of the data.
5. Compare the model output to the data by determining SSE.
    a. Build and use a function to compare and return error.
6. Change the parameters and repeat until a minimum SSE is found.
    a. Use a built-in function (optim) to automate this.
7. Record the best parameters.

Data fitting of SIR model in R

Code file: 6_*Intro_ID_Fit_SIR.R*

Set up similar to the SIR code

- Requires data as well as model set-up.

```
# Initial conditions
N <- 763
I_init <- 3
R_init <- 0
S_init <- N - I_init - R_init

# Initial guesses
alpha <- 0.1
beta <- 0.001

# Initial conditions for state variables
init_cond <- c(S_init, I_init, R_init)

# Data from the table in Example 5.1
Idata <- c(3, 6, 25, 75, 227, 296, 258, 236, 192, 126, 71, 28, 11, 7)

# Time span
tspan <- seq(1, 14, by = 0.01)

# Vector with the initial guesses
par <- c(alpha, beta)
```

Functions needed for optimization

```r
# SIR model function
SIR_model <- function(t_vec, y_init, par) {
  dt <- t_vec[2] - t_vec[1]
  alpha <- par[1]
  beta <- par[2]

  n <- length(t_vec)
  S <- numeric(n)
  I <- numeric(n)
  R <- numeric(n)

  S[1] <- y_init[1]
  I[1] <- y_init[2]
  R[1] <- y_init[3]

  for (t in 1:(n - 1)) {
    S[t + 1] <- S[t] - (beta * S[t] * I[t]) * dt
    I[t + 1] <- I[t] + (beta * S[t] * I[t] - alpha * I[t]) * dt
    R[t + 1] <- R[t] + alpha * I[t] * dt
  }
  cbind(S, I, R)
}
```

```r
# Error function
err_in_dataSIR <- function(par, Idata, init_cond, tspan) {
  y <- SIR_model(tspan, init_cond, par)
  indices <- round(seq(1, length(tspan), length.out = length(Idata)))
  Infected <- y[indices,2]
  error_in_data <- sum((Infected - Idata)^2)
  is.na(error_in_data)
  if (is.na(error_in_data)){
    error_in_data <- 1e99
  }
  error_in_data
}
```

Run optimization and output

```
# Optimization to estimate parameters
opt <- optim(par, fn = err_in_dataSIR, Idata = Idata, init_cond = init_cond, tspan = tspan)
par_est <- opt$par

# Solve system with estimated parameters
y_est <- SIR_model(tspan, init_cond, par_est)
```

Plot data and model output

```r
# Plot solution along with data
# data frame for model output
df_plot <- data.frame(
  Time = tspan,
  Infected = y_est[, 2]
)
# data frame for data
df_data <- data.frame(
  Time = 1:14,
  Infected = Idata
)

ggplot() +
  geom_line(data = df_plot, aes(x = Time, y = Infected), color = "blue", size = 1) +
  geom_point(data = df_data, aes(x = Time, y = Infected), color = "red", size = 3) +
  labs(x = "Time in days", y = "Number of infected individuals") +
  theme_minimal() +
  theme(text = element_text(size = 14)) +
  ggtitle("Fitted Model and Data") +
  scale_x_continuous(breaks = 1:14) +
  theme(legend.position = "none")
```