Module 4: KAMEN4_MIRANDA_BOON_MODULE_4.IPYNB

Team Members:

Hazel Miranda and Spencer Boon

Project Title:

Modeling the Measles Outbreak in Nigeria (2007–2021) Using the SIR Epidemiological Framework

Project Goal:

This project seeks to analyze and model measles transmission in Nigeria from 2007–2021 using the SIR epidemiological model, by fitting the model to real outbreak data, estimating key disease parameters (β, γ, R_0), and evaluating how well the SIR framework explains the observed epidemic trends.

Disease Background:

Using your assigned disease, fill in the following bullet points.

- Prevalence & incidence
    - *A review of Nigerian measles data found that in one secondary health-centre study in southern Nigeria, measles accounted for 3.1% of all paediatric admissions over a 5-year period.*
    - *Another study tracking incidence data across Nigeria noted that in 2021 the country had the largest number of measles cases of any country, over 10,000 reported cases.*
    - *The "Trends of the second measles vaccine (MCV2)" paper noted that Nigeria has "one of the highest measles burdens and lowest vaccination coverage in the world."*
    - *While exact national incidence per year for your dataset period (2007-2021) might require digging into surveillance datasets, the literature clearly shows that measles remains endemic in Nigeria with frequent outbreaks, and that incidence has been persistently high.*

  Sources: https://pmc.ncbi.nlm.nih.gov/articles/PMC6445333/ https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2024.1392996/

- Economic burden
    - *A paper on catastrophic health expenditures (CHE) for measles in Nigeria found that contracting measles puts households at risk of CHE (out-of-pocket medical costs exceeding a threshold of income/consumption).*
    - *It shows that poorer households and those in certain regions (e.g., north vs south) have substantially higher CHE risk when measles occurs.*
    - *Another costing study of measles campaigns in Nigeria (2019 data) showed that an integrated measles vaccination campaign saved up to USD* $420,000 compared to stand-alone campaigns in five states; savings of over$ *200 per 1,000 children were documented.*
    - *Another study in southeast Nigeria found the cost per child immunized (routine immunization) for measles was US* $1.41 (total cost), and US$ *1.01 (operational cost) per child.*
    - *Measles exerts an economic burden via both direct medical costs (treatment, hospitalisation) and vaccination-programme costs, and that poor vaccination coverage actually translates into larger outbreaks and higher costs. Use the Nigeria-specific figures above.*

  Sources: https://pmc.ncbi.nlm.nih.gov/articles/PMC9302492/ https://healtheconomicsreview.biomedcentral.com/articles/10.1186/s13561-023-00441-y

- Risk factors (genetic, lifestyle) & Societal determinants
    - *Vaccination status is the major determinant: the "Trends of MCV2" study found very low coverage of the second dose of measles vaccine in Nigeria (≈20% when introduced, rising to only ~38-46% in some northern zones) thus leaving large susceptible populations.*
    - *Broader societal determinants: the "Population-Level Risk Factors" study identified many indicators that correlate with measles case fatality and likely with incidence: under-five mortality rate, vaccination coverage, travel time to health facility, stunting/malnutrition prevalence, conflict/displacement.*
    - *A Nigeria-specific review (2024) emphasized how conflict-affected zones, displaced populations, rapid urbanization, and poor immunization access all drive recurrent outbreaks.*
    - *Genetic risk factors: There is little evidence in Nigeria-specific literature for a strong genetic predisposition to measles; the risk is driven far more by vaccination status, immunity status, and exposure.*
    - *Lifestyle factors: Malnutrition (vitamin A deficiency, under-nutrition), poor access to health services, high population density/household crowding are important.*
    - *The key risk factors are low vaccination coverage, poverty/inequity in access, malnutrition/immune compromised children, geographic/socio-political context (conflict/displacement). Genetic risk is minimal in the literature in this setting.*

Sources: https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2024.1392996/full https://www.mdpi.com/2076-393X/11/8/1389 https://aeji.journals.ekb.eg/article_392829.html

- Symptoms
  - *According to the standard clinical description (CDC Pink Book) measles begins with high fever, cough, coryza (runny nose), conjunctivitis; 2-3 days later Koplik's spots (tiny white lesions inside mouth) appear; then a characteristic red maculopapular (flat and raised) rash develops, usually starting on the face and spreading downward.*
  - *In Nigeria, hospital studies report typical presentations in children: rash, fever, complications like pneumonia, otitis media, encephalitis.*
  - *Key complications: pneumonia (most common cause of death in measles in low-income settings), diarrhoea, ear infections, blindness/corneal scarring, encephalitis, subacute sclerosing panencephalitis (SSPE) years later.*

  Sources: https://www.cdc.gov/pinkbook/hcp/table-of-contents/chapter-13-measles.html https://journals.plos.org/plosmedicine/article?id=10.1371%2Fjournal.pmed.0040016 https://pmc.ncbi.nlm.nih.gov/articles/PMC6445333/

- Diagnosis
  - *The CDC Pink Book chapter outlines diagnosis: clinical presentation (fever + cough + coryza + conjunctivitis + rash) plus laboratory confirmation via measles IgM antibodies in serum or measles virus RNA via PCR from throat/urine.*
  - *In Nigeria case-based surveillance systems treat reported suspected measles cases, then confirm via lab/epidemiological linkage. For example the review "Control of Measles in Nigeria" describes Nigeria's surveillance system established in 2006.*
  - *Diagnosis in Nigeria includes suspected case definition, lab confirmation (IgM/RT-PCR), and epidemiological linkage. Also understand there are limitations in resource-limited settings such as incomplete lab access, underreporting.*

  Sources: https://www.cdc.gov/pinkbook/hcp/table-of-contents/chapter-13-measles.html https://www.journaljammr.com/index.php/JAMMR/article/view/1411

- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
  - *Pathogen: the Measles virus (MeV) is a single-stranded, negative-sense, enveloped RNA virus in the genus Morbillivirus (family Paramyxoviridae).*
  - *Entry & spread: MeV enters via the respiratory tract, infects alveolar macrophages or dendritic cells via the SLAM (CD150) receptor on immune cells. Then it spreads to lymphoid tissues causing viremia. Later it infects epithelial cells via the nectin-4 receptor, leading to rash and virus shedding via the respiratory route.*
  - *Immune effects: MeV causes immune suppression and "immune amnesia" by depleting memory B and T-cells, meaning after measles infection the host's immune memory to other pathogens is reduced for months/years.*

- *Organ systems: The virus impacts the respiratory system (cough, pneumonia), skin (rash from infected epithelial cells with immune response), immune system (persistent immunosuppression), and sometimes nervous system (encephalitis, SSPE).*
- *Cell/molecular level: MeV's hemagglutinin (H) and fusion (F) glycoproteins mediate attachment and fusion; H binds receptors (SLAM/CD150, nectin-4) triggering fusion of viral envelope and host cell membrane via F protein.*
- *Steps:*
- *(1) inhalation*
- *(2) infection of immune cells*
- *(3) viremia*
- *(4) infection of epithelial/skin cells*
- *(5) rash and viral shedding + immune suppression*
- *(6) secondary infections due to immunosuppression*
- *Note: High-R_0 arises partly because of this efficient spread and high susceptibility of children in Nigeria.*

Sources: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736%2821%2902004-3/fulltext https://www.mdpi.com/1999-4915/14/12/2641 https://www.nature.com/articles/s41467-018-07515-0

Dataset:

How the data was collected

(Describe the data set you will analyze. Cite the source(s) of the data. Describe how the data was collected -- What techniques were used? What units are the data measured in? Etc.)

- Our data set is from Adebayo, A. (2022). Nigeria Measles Outbreak Dataset (2007-2021) [Data set]. Kaggle. https://www.kaggle.com/datasets/adegboyegaadebayo/nigeria-measles-outbreak-dataset2007-2021
  - The measles epidemiological data used for the data set was extracted from the Nigeria Centre Disease Control situation report and the national disease outbreak dashboard (https://ncdc.gov.ng/data) using the application webplotdigitizer (https://apps.automeris.io/wpd/)
    - These are official national surveillance summaries reporting:
      - Number of suspected and confirmed measles cases
      - Geographic distribution
      - Time trends (weekly, monthly, or yearly)

- Outbreak counts
- Case totals by state
- Time-series visualizations
- Laboratory confirmations in some years

Techniques used to extract the dataset

The dataset creator did not download raw spreadsheets from NCDC (because NCDC often publishes graphs instead of downloadable files). Instead, they used WebPlotDigitizer (A digital extraction tool that allows a user to upload a plot or graph and convert the plotted points into numerical data)

- This means: The researcher took graphs (case counts over time, outbreak charts, etc.) from NCDC reports.
  - Uploaded them into WebPlotDigitizer.
  - Extracted the exact numerical values (e.g., monthly or yearly case counts).
  - Compiled these into a CSV file.
  - This is a recognized and common technique used in research when data are only available as plotted figures, not tables.

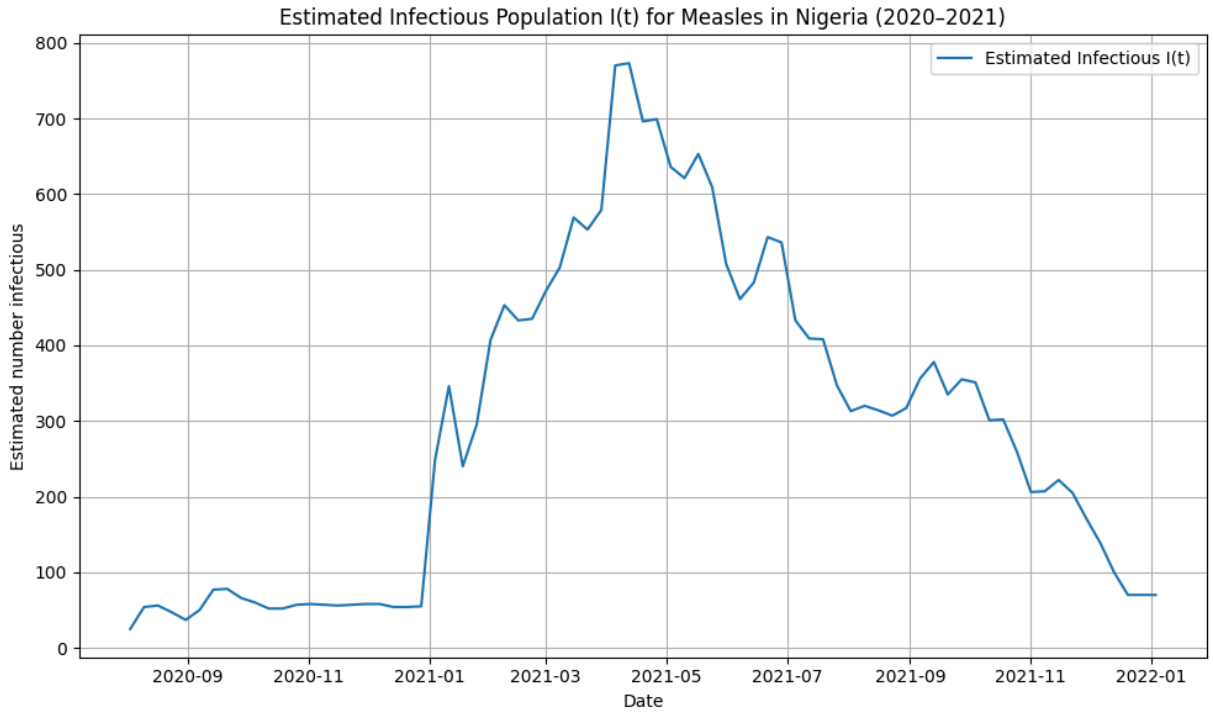
Data set qualities

- The data is Weekly collected (not daily) from 2020-08-03 to 2022-01-03
- Accoss multiple online sources the concensous is that measles infectious period is about 10-14 days. For our project we will use the median of 12 and round it up to 2 weeks.
- The data is not cumulative and compares new cases

In [2]:
```
## LOAD YOUR DATASET HERE.
import pandas as pd
import matplotlib.pyplot as plt
from main_functions import convert_cumulative_to_SIR
# 1. Read your weekly measles data
df = pd.read_csv("measles_nigeria_data_2020-2021_new_cases.csv")
df["date"] = pd.to_datetime(df["date"])
# The CSV has weekly *new* confirmed cases.
# Build a cumulative column for the helper function:
df["cumulative_cases"] = df["confirmed_cases"].cumsum()

# 2. Convert cumulative cases → S, I, R using a WEEK-based infectious period
population_nigeria = 206_000_000  # approx 2020–2021
infectious_period_weeks = 2       # 12 days ≈ 1.7 weeks → ~2 weekly time steps
sir_df = convert_cumulative_to_SIR(
    df,
    date_col="date",
    cumulative_col="cumulative_cases",
    population=population_nigeria,
    infectious_period=infectious_period_weeks)
print(sir_df.head())
```

```
        date  confirmed_cases  cumulative_cases  new_cases  I_est  R_est  \
0 2020-08-03             25.0              25.0       25.0   25.0    0.0
1 2020-08-10             29.0              54.0       29.0   54.0    0.0
2 2020-08-17             27.0              81.0       27.0   56.0   25.0
3 2020-08-24             20.0             101.0       20.0   47.0   54.0
4 2020-08-31             17.0             118.0       17.0   37.0   81.0

         S_est
0  205999975.0
1  205999946.0
2  205999919.0
3  205999899.0
4  205999882.0
```

In [3]:
```python
# Plot I(t) alone
plt.figure(figsize=(10, 6))
plt.plot(sir_df["date"], sir_df["I_est"], label="Estimated Infectious I(t)")
plt.xlabel("Date")
plt.ylabel("Estimated number infectious")
plt.title("Estimated Infectious Population I(t) for Measles in Nigeria (2020–2021)"
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



Data Analyis:

Methods

To analyze measles transmission in Nigeria, we started with weekly confirmed case counts from 2020–2021. We first constructed a cumulative case curve and used it to estimate SIR

compartments by assuming an infectious period of two weeks. Specifically, we computed weekly new cases, estimated the infectious population I(t) as a 2-week rolling sum of new cases, approximated the recovered population R(t) as the cumulative cases shifted forward by two weeks, and then defined the susceptible population S(t) as $S(t)=N-I(t)-R(t)$ where N is the approximate total population of Nigeria.

We then modeled disease dynamics using the standard SIR system of ODEs and implemented two numerical solvers. First, we used Euler's method to simulate S(t), I(t), and R(t) for given transmission (β) and recovery (γ) rates, and we quantified model–data mismatch using the sum of squared errors (SSE) between modeled and data-based I(t). We fit β and γ in two ways: (1) using scipy.optimize.minimize on the full time series, and (2) using a grid search that minimized SSE only on the first half of the data, then computed SSE on the second half as an out-of-sample prediction error. Finally, to reduce numerical error, we replaced Euler's method with scipy.integrate.solve_ivp using the RK45 scheme, repeated the first-half fitting and second-half evaluation, and compared the resulting SSE values from Euler and RK45 to assess the impact of the numerical method on model accuracy.

Analysis

(Describe how you analyzed the data. This is where you should intersperse your Python code so that anyone reading this can run your code to perform the analysis that you did, generate your figures, etc.)

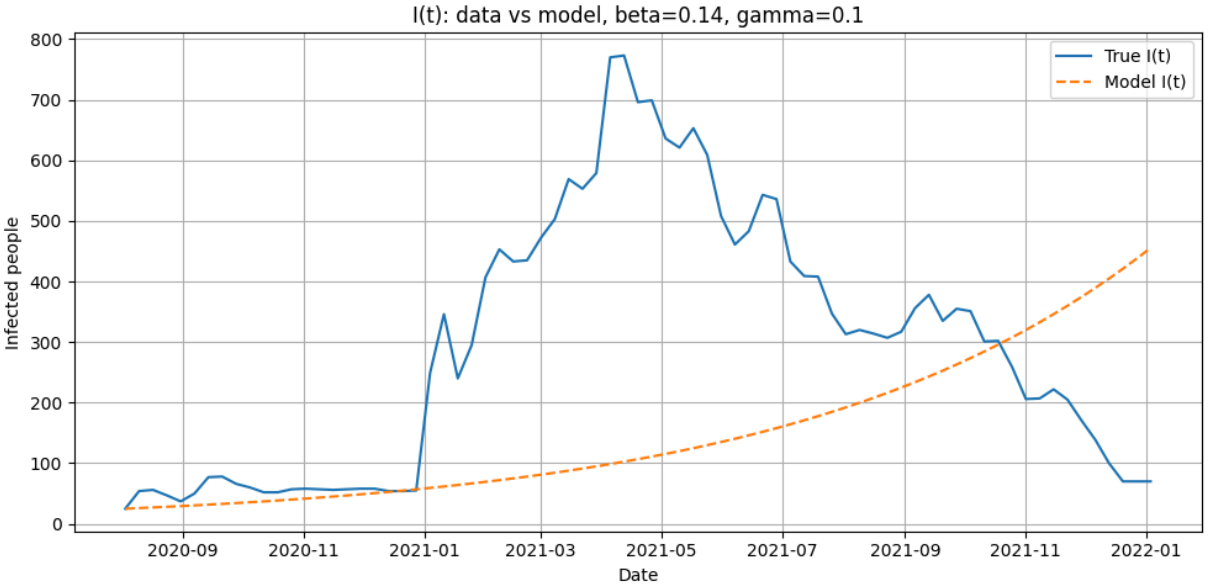
1. Fitting The SIR Model

In [5]:
```
## Plot of Euler, I(t) true and I(t) model with gamma and beta guess

import pandas as pd
import numpy as np
from main_functions import convert_cumulative_to_SIR

def euler_sir(beta, gamma, S0, I0, R0, t, N):
    """
    Solve the SIR model using Euler's method.
    """

    # Preallocate arrays for S(t), I(t), R(t)
    S = np.empty(len(t), float)
    I = np.empty(len(t), float)
    R = np.empty(len(t), float)

    # Set initial conditions from data
    S[0], I[0], R[0] = S0, I0, R0

    # Loop through all time steps and apply Euler updates
    for n in range(len(t) - 1):
        dt = t[n + 1] - t[n]   # time step size
```

```
        # SIR differential equations
        dS = -beta * S[n] * I[n] / N
        dI = beta * S[n] * I[n] / N - gamma * I[n]
        dR = gamma * I[n]

        # Euler update rule: next value = current + derivative * dt
        S[n + 1] = S[n] + dS * dt
        I[n + 1] = I[n] + dI * dt
        R[n + 1] = R[n] + dR * dt

    return S, I, R


# Convert cumulative case data → S, I, R estimated from real data
sir_df = convert_cumulative_to_SIR(
    df,
    date_col="date",
    cumulative_col="cumulative_cases",
    population=population_nigeria,
    infectious_period=infectious_period_weeks)

# Create time array (one step per row in dataframe)
t = np.arange(len(sir_df), dtype=float)

# Total population
N = population_nigeria

# Initial S, I, R taken from the first row of estimated SIR dataset
S0 = sir_df["S_est"].iloc[0]
I0 = sir_df["I_est"].iloc[0]
R0 = sir_df["R_est"].iloc[0]

# True I(t) from data
I_true = sir_df["I_est"].values

# Parameter guesses for Euler simulation
beta_guess = 0.14
gamma_guess = 0.1

# Run Euler SIR model using guessed beta and gamma
S_model, I_model, R_model = euler_sir(
    beta=beta_guess,
    gamma=gamma_guess,
    S0=S0,
    I0=I0,
    R0=R0,
    t=t,
    N=N)
# Quick diagnostics of model performance
print("I_true min/max:", I_true.min(), I_true.max())
print("I_model min/max:", I_model.min(), I_model.max())

# -------- Plot: TRUE vs MODEL I(t) --------
plt.figure(figsize=(10, 5))
plt.plot(sir_df["date"], I_true, label="True I(t)")                    # data-base
plt.plot(sir_df["date"], I_model, label="Model I(t)", linestyle="--")   # Euler-sim
```

```
plt.xlabel("Date")
plt.ylabel("Infected people")
plt.title(f"I(t): data vs model, beta={beta_guess}, gamma={gamma_guess}")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Compute SSE between model I(t) and true I(t)
sse_I = np.sum((I_model - I_true) ** 2)
print("SSE for I(t):", sse_I)
```

```
I_true min/max: 25.0 773.0
I_model min/max: 25.0 455.40494403116116
```




```
SSE for I(t): 6001542.017750413
```

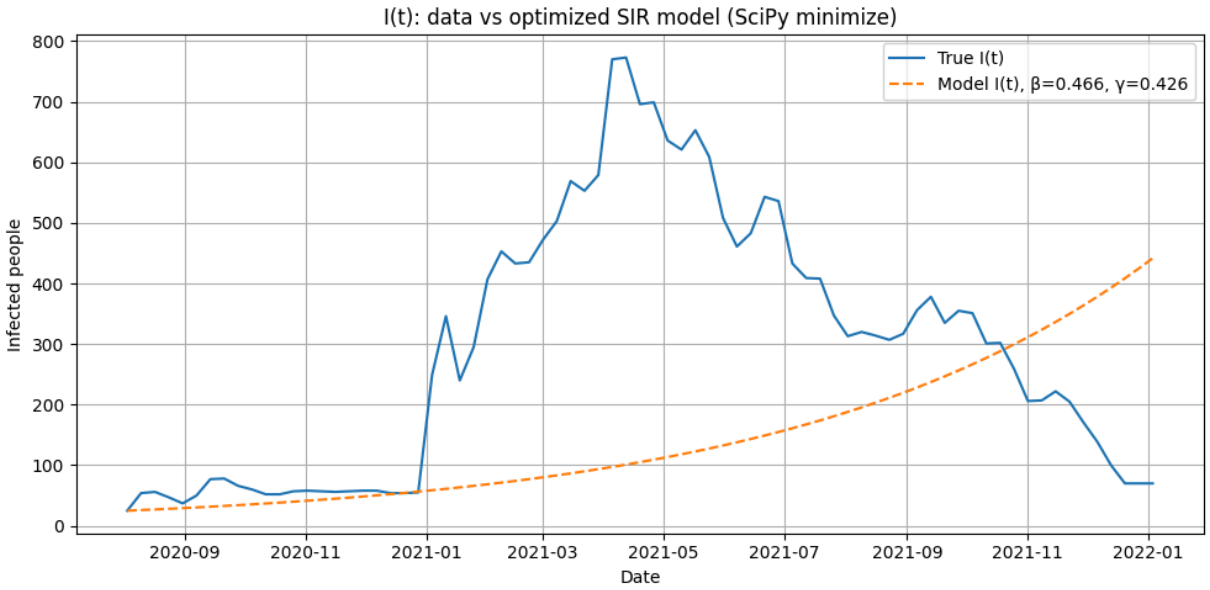
In [ ]:

```
## Optimized graph of beta and gamma with best values and new SSE

from scipy.optimize import minimize

# True I(t) from data and setup for SIR model
I_true = sir_df["I_est"].values
t = np.arange(len(sir_df), dtype=float)
N = population_nigeria

# Initial conditions from first row of SIR estimates
S0 = sir_df["S_est"].iloc[0]
I0 = sir_df["I_est"].iloc[0]
R0 = sir_df["R_est"].iloc[0]

def sse_objective(params):
    """
    Objective function for optimization.
    params[0] = beta, params[1] = gamma.
    Returns SSE between model I(t) and data I_true.
    """
    beta, gamma = params
```

```
    # Penalize non-physical negative parameters
    if beta <= 0 or gamma <= 0:
        return 1e30  # huge penalty

    # Simulate SIR model with current beta, gamma
    S_model, I_model, R_model = euler_sir(
        beta=beta,
        gamma=gamma,
        S0=S0,
        I0=I0,
        R0=R0,
        t=t,
        N=N
    )

    # Sum of squared errors between model and data
    return np.sum((I_model - I_true) ** 2)


# Initial guess for (beta, gamma)
initial_guess = np.array([0.2, 0.5])

# Bounds to keep beta, gamma in reasonable ranges
bounds = [
    (1e-6, 2.0),  # beta between ~0 and 2 per week
    (1e-6, 2.0)   # gamma between ~0 and 2 per week
    ]

# Run numerical optimization to minimize SSE
result = minimize(
    sse_objective,
    x0=initial_guess,
    bounds=bounds,
    method="L-BFGS-B")

# Extract optimal parameters
beta_opt, gamma_opt = result.x

print("Optimal beta:", beta_opt)
print("Optimal gamma:", gamma_opt)
print("Optimal SSE:", result.fun)

# Recompute SIR trajectories with optimal parameters
S_opt, I_opt, R_opt = euler_sir(
    beta=beta_opt,
    gamma=gamma_opt,
    S0=S0,
    I0=I0,
    R0=R0,
    t=t,
    N=N)

# Plot data vs optimized model
plt.figure(figsize=(10, 5))
plt.plot(sir_df["date"], I_true, label="True I(t)")  # data-based I(t)
```

```
plt.plot(
    sir_df["date"],
    I_opt,
    "--",
    label=f"Model I(t), β={beta_opt:.3f}, γ={gamma_opt:.3f}")
plt.xlabel("Date")
plt.ylabel("Infected people")
plt.title("I(t): data vs optimized SIR model (SciPy minimize)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Optimal beta: 0.4657494404878756
Optimal gamma: 0.4261809105915359
Optimal SSE: 5999184.742740708
```



2. Predict "the future" with your fit SIR model

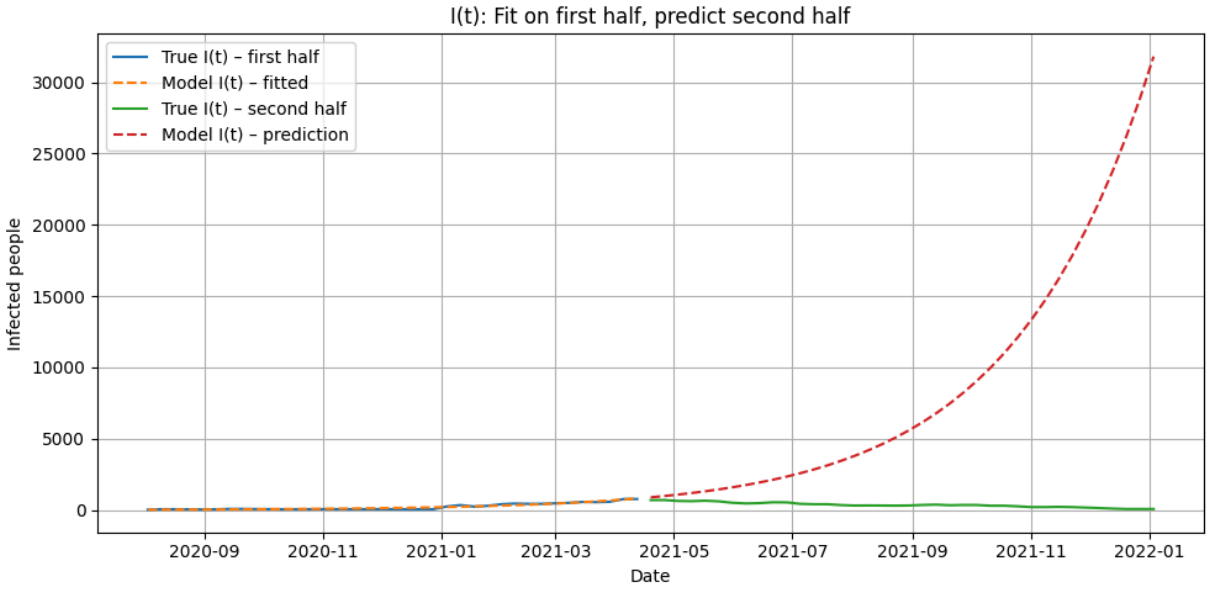
In [7]:

```
## Fitting with first half of data, and calculating SSE on second half of data

# Full I(t) and date arrays from SIR estimates
I_true_all = sir_df["I_est"].values
dates_all = sir_df["date"].values
n = len(sir_df)

# Time array for full dataset (assume weekly spacing)
t_all = np.arange(n, dtype=float)

# Split index between first and second half of time series
mid = n // 2  # integer division

# First-half data (used for fitting parameters)
I_true_fit = I_true_all[:mid]
t_fit = t_all[:mid]

# Second-half data (used only for evaluating prediction error)
```

```
I_true_second = I_true_all[mid:]
t_second = t_all[mid:]

N = population_nigeria

# Initial conditions from the first row of SIR estimates
S0 = sir_df["S_est"].iloc[0]
I0 = sir_df["I_est"].iloc[0]
R0 = sir_df["R_est"].iloc[0]

# ---------- SSE function that uses ONLY FIRST HALF ----------
def sir_sse_first_half(beta, gamma):
    """
    Compute SSE between model and data for the FIRST HALF of the time series.
    """
    # Run SIR model over full time span
    S_model, I_model, R_model = euler_sir(
        beta=beta,
        gamma=gamma,
        S0=S0,
        I0=I0,
        R0=R0,
        t=t_all,
        N=N
    )
    # Restrict SSE calculation to first-half data points
    return np.sum((I_model[:mid] - I_true_fit) ** 2)

# ---------- GRID SEARCH on beta, gamma using FIRST HALF ----------
beta_values = np.linspace(0.01, 1.0, 50)   # search range for beta
gamma_values = np.linspace(0.05, 1.0, 50)  # search range for gamma

best_sse = np.inf
best_beta = None
best_gamma = None

# Brute-force search over grid of (beta, gamma) pairs
for beta in beta_values:
    for gamma in gamma_values:
        sse = sir_sse_first_half(beta, gamma)
        if sse < best_sse:
            best_sse = sse
            best_beta = beta
            best_gamma = gamma

print("Best beta (fit on first half):", best_beta)
print("Best gamma (fit on first half):", best_gamma)
print("Best SSE on FIRST HALF:", best_sse)

# ---------- RUN MODEL WITH BEST PARAMETERS ON FULL TIME RANGE ----------
S_best, I_best, R_best = euler_sir(
    beta=best_beta,
    gamma=best_gamma,
    S0=S0,
    I0=I0,
    R0=R0,
```

```
    t=t_all,
    N=N)

# ---------- COMPUTE ERROR ON SECOND HALF ----------
# Model-predicted I(t) over second half
I_model_second = I_best[mid:]
sse_second_half = np.sum((I_model_second - I_true_second) ** 2)

print("SSE on SECOND HALF (prediction error for Euler model):", sse_second_half)

# ---------- PLOT: FIT ON FIRST HALF, PREDICT SECOND HALF ----------
plt.figure(figsize=(10, 5))

# First half: where parameters were fit
plt.plot(dates_all[:mid], I_true_fit, label="True I(t) - first half")
plt.plot(dates_all[:mid], I_best[:mid], "--", label="Model I(t) - fitted")

# Second half: model prediction vs data
plt.plot(dates_all[mid:], I_true_second, label="True I(t) - second half")
plt.plot(dates_all[mid:], I_model_second, "--", label="Model I(t) - prediction")

plt.xlabel("Date")
plt.ylabel("Infected people")
plt.title("I(t): Fit on first half, predict second half")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Best beta (fit on first half): 0.15142857142857144
Best gamma (fit on first half): 0.05
Best SSE on FIRST HALF: 124374.48717843318
SSE on SECOND HALF (prediction error for Euler model): 5618617930.049551
```



- Is the new gamma and beta close to what you found on the full dataset? Is the fit much worse? What is the SSE calculated for the second half of the data?
  - The new gamma and beta are close to what we found on the full dataset.

    - The fit is better and is closer to the graph
    - the SSE calculated for the second half of the data is 5618617930.049551

- Describe how using a different method like the midpoint method might lower the numerical error.
    - The midpoint method (also called second-order Runge–Kutta, RK2) reduces this error by evaluating the derivatives twice per step:
        - Take an Euler "half-step" to estimate the state at the midpoint.
        - Compute the derivative at that midpoint.Use that midpoint derivative to update the full step.
        - This makes the midpoint method second-order accurate, with global error O(h^2) instead of O(h).

3. Decreasing numerical error with the RK4 Method

In [21]:
```
## RK45 (solve_ivp) SIR fitting and comparison to Euler

from scipy.integrate import solve_ivp

def sir_ode(t, y, beta, gamma, N):
    """
    Right-hand side of the SIR ODE system for solve_ivp.
    y = [S, I, R]
    """
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]

def rk45_sir(beta, gamma, S0, I0, R0, t_eval, N):
    """
    Integrate the SIR model using solve_ivp with RK45.
    - t_eval: array of time points where solution is returned (e.g., t_all = np.ara
    Returns S, I, R arrays aligned with t_eval.
    """
    sol = solve_ivp(
        fun=lambda t, y: sir_ode(t, y, beta, gamma, N),
        t_span=(t_eval[0], t_eval[-1]),
        y0=[S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )

    # If solver fails, raise so we can penalize in SSE function
    if not sol.success:
        raise RuntimeError("solve_ivp failed: " + sol.message)

    # sol.y has shape (3, len(t_eval)): rows = S, I, R
    S, I, R = sol.y
    return S, I, R
```

```
# ---------- DATA & TIME SETUP ----------
I_true_all = sir_df["I_est"].values
dates_all = sir_df["date"].values
n = len(sir_df)

# Time array (assume weekly spacing)
t_all = np.arange(n, dtype=float)

# Split into first and second halves
mid = n // 2
I_true_fit = I_true_all[:mid]      # first half (for fitting)
t_fit = t_all[:mid]
I_true_second = I_true_all[mid:]   # second half (for prediction error)
t_second = t_all[mid:]

# Population and initial conditions
N = population_nigeria
S0 = sir_df["S_est"].iloc[0]
I0 = sir_df["I_est"].iloc[0]
R0 = sir_df["R_est"].iloc[0]

import numpy as np

def sir_sse_first_half_rk45(beta, gamma):
    """
    SSE between RK45-based SIR model and data for the FIRST HALF of the time series
    """
    # Enforce positivity; negative parameters are non-physical
    if beta <= 0 or gamma <= 0:
        return 1e30

    try:
        # Integrate over the full time range
        S_model, I_model, R_model = rk45_sir(
            beta=beta,
            gamma=gamma,
            S0=S0,
            I0=I0,
            R0=R0,
            t_eval=t_all,
            N=N
        )
    except RuntimeError:
        # If the ODE solver fails, treat this parameter pair as very bad
        return 1e30

    # SSE only on first-half data points
    return np.sum((I_model[:mid] - I_true_fit) ** 2)


# ---------- GRID SEARCH (RK45) ON FIRST HALF ----------
beta_values = np.linspace(0.01, 1.0, 50)   # search grid for beta
gamma_values = np.linspace(0.05, 1.0, 50)  # search grid for gamma
```

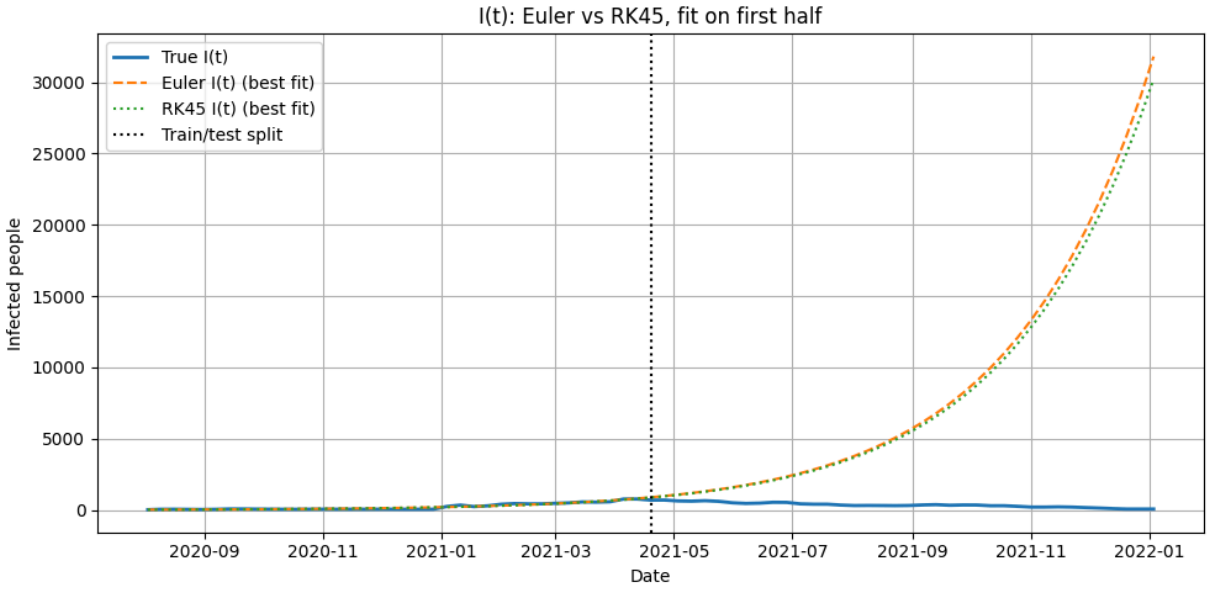
```
best_sse_rk45 = np.inf
best_beta_rk45 = None
best_gamma_rk45 = None

# Brute-force search over (beta, gamma) pairs
for beta in beta_values:
    for gamma in gamma_values:
        sse = sir_sse_first_half_rk45(beta, gamma)
        if sse < best_sse_rk45:
            best_sse_rk45 = sse
            best_beta_rk45 = beta
            best_gamma_rk45 = gamma

print("RK45 best beta (fit on first half):", best_beta_rk45)
print("RK45 best gamma (fit on first half):", best_gamma_rk45)
print("RK45 best SSE on FIRST HALF:", best_sse_rk45)


# ---------- RUN RK45 MODEL WITH BEST PARAMETERS ON FULL TIME RANGE ----------
S_best_rk45, I_best_rk45, R_best_rk45 = rk45_sir(
    beta=best_beta_rk45,
    gamma=best_gamma_rk45,
    S0=S0,
    I0=I0,
    R0=R0,
    t_eval=t_all,
    N=N
)

# ---------- SECOND-HALF SSE (PREDICTION ERROR) ----------
I_model_second_rk45 = I_best_rk45[mid:]
sse_second_half_rk45 = np.sum((I_model_second_rk45 - I_true_second) ** 2)

# sse_second_half is the Euler-based second-half SSE computed earlier
print("Euler SSE on second half:", sse_second_half)
print("RK45 SSE on second half:", sse_second_half_rk45)
```

```
RK45 best beta (fit on first half): 0.9797959183673469
RK45 best gamma (fit on first half): 0.8836734693877552
RK45 best SSE on FIRST HALF: 124408.63390074746
Euler SSE on second half: 5618617930.049551
RK45 SSE on second half: 5133231354.87359
```

In [23]:

```
## Comparison of Euler vs RK45 on our data

plt.figure(figsize=(10, 5))

# True infections over time from data
plt.plot(dates_all, I_true_all, label="True I(t)", linewidth=2)

# Euler model prediction (best-fit parameters from grid search)
plt.plot(dates_all, I_best, "--", label="Euler I(t) (best fit)")

# RK45 model prediction (best-fit parameters from RK45 grid search)
plt.plot(dates_all, I_best_rk45, ":", label="RK45 I(t) (best fit)")
```

```
# Vertical line indicating where we split first/second half (train/test)
plt.axvline(dates_all[mid], color="k", linestyle=":", label="Train/test split")

plt.xlabel("Date")
plt.ylabel("Infected people")
plt.title("I(t): Euler vs RK45, fit on first half")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Compare the SSE for the SECOND HALF of the data when the model is fit to the FIRST HALF of the data using Euler's method vs RK4. Did RK4 do a better job? Why or why not?

- Euler SSE on second half: 5618617930.049551
- RK4/RK45 SSE on second half: 5133231354.87359

- In my results SSE RK4 2nd half was smaller than SSE Euler 2nd half

- This indicates that the higher-order Runge–Kutta method did a better job predicting the second half of the outbreak. RK4/RK45 has a smaller truncation error than Euler (global error O(h^2) vs O(h)), so for the same weekly time step it tracks the continuous SIR dynamics more accurately. Because the parameters are estimated from the first half of the data, having a more accurate numerical solution means the fitted β and γ correspond more closely to the "true" underlying ODE model, which translates into a lower SSE when we project forward onto the second half.

4. Improving model fit by overcoming model limitations

- We chose to include an exposed compartment (SEIR model).

In [ ]:
```
# SEIR EXTENDED MODEL (Exposed compartment) WITH RK45
import numpy as np
```

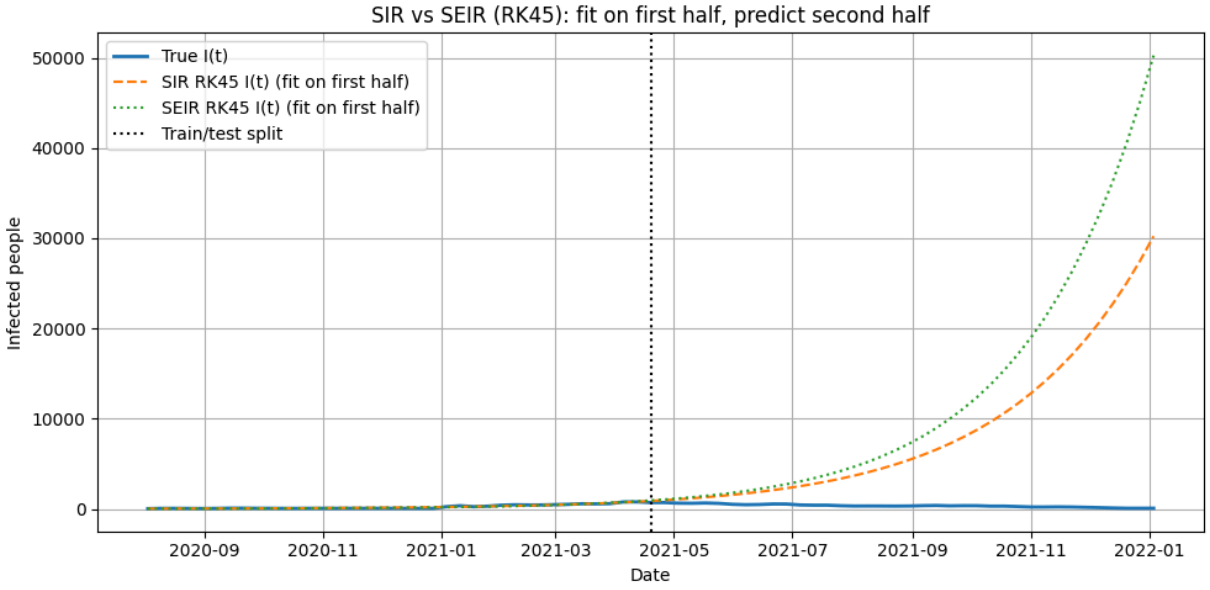
```python
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# ---- Assume sir_df already exists from earlier code ----
# sir_df must have columns: "date", "I_est", "S_est", "R_est"

# Population and initial conditions
population_nigeria = 206_000_000
N = population_nigeria

S0 = sir_df["S_est"].iloc[0]
I0 = sir_df["I_est"].iloc[0]
R0 = sir_df["R_est"].iloc[0]

# Time and data arrays
I_true_all = sir_df["I_est"].values
dates_all  = sir_df["date"].values
t_all      = np.arange(len(sir_df), dtype=float)

# Train/test split
n   = len(sir_df)
mid = n // 2

I_true_fit    = I_true_all[:mid]   # first half (training)
I_true_second = I_true_all[mid:]   # second half (testing)


# ============================================================
# 1. SIR model with RK45 (baseline extended model)
# ============================================================

def sir_ode(t, y, beta, gamma, N):
    """SIR ODE system used by solve_ivp."""
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]

def rk45_sir(beta, gamma, S0, I0, R0, t_eval, N):
    """Solve SIR with solve_ivp (RK45)."""
    sol = solve_ivp(
        fun=lambda t, y: sir_ode(t, y, beta, gamma, N),
        t_span=(t_eval[0], t_eval[-1]),
        y0=[S0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    if not sol.success:
        raise RuntimeError("SIR solve_ivp failed: " + sol.message)
    S, I, R = sol.y
    return S, I, R

def sir_sse_first_half_rk45(beta, gamma):
    """SSE between SIR RK45 model and data I(t) on FIRST HALF only."""
    if beta <= 0 or gamma <= 0:
```

```
        return 1e30
    try:
        S_model, I_model, R_model = rk45_sir(
            beta=beta,
            gamma=gamma,
            S0=S0,
            I0=I0,
            R0=R0,
            t_eval=t_all,
            N=N
        )
    except RuntimeError:
        return 1e30
    return np.sum((I_model[:mid] - I_true_fit) ** 2)

# Grid search for SIR beta, gamma using first half
beta_values_sir  = np.linspace(0.01, 1.0, 50)
gamma_values_sir = np.linspace(0.05, 1.0, 50)

best_sse_rk45   = np.inf
best_beta_rk45  = None
best_gamma_rk45 = None

for beta in beta_values_sir:
    for gamma in gamma_values_sir:
        sse = sir_sse_first_half_rk45(beta, gamma)
        if sse < best_sse_rk45:
            best_sse_rk45  = sse
            best_beta_rk45 = beta
            best_gamma_rk45 = gamma

print("SIR RK45 best beta (first half):", best_beta_rk45)
print("SIR RK45 best gamma (first half):", best_gamma_rk45)
print("SIR RK45 best SSE on FIRST HALF:", best_sse_rk45)

# Run SIR RK45 with best-fit parameters on full time range
S_best_rk45, I_best_rk45, R_best_rk45 = rk45_sir(
    beta=best_beta_rk45,
    gamma=best_gamma_rk45,
    S0=S0,
    I0=I0,
    R0=R0,
    t_eval=t_all,
    N=N
)

# SSE on second half (prediction error)
I_model_second_rk45   = I_best_rk45[mid:]
sse_second_half_rk45  = np.sum((I_model_second_rk45 - I_true_second) ** 2)
print("SIR RK45 SSE on SECOND HALF:", sse_second_half_rk45)


# ============================================================
# 2. SEIR model with RK45 (E compartment extension)
# ============================================================
```

```
# Choose latent period in weeks and corresponding sigma
latent_period_weeks = 2.0
sigma = 1.0 / latent_period_weeks

def seir_ode(t, y, beta, gamma, sigma, N):
    """
    SEIR ODE system:
        dS/dt = -beta * S * I / N
        dE/dt =  beta * S * I / N - sigma * E
        dI/dt =  sigma * E - gamma * I
        dR/dt =  gamma * I
    """
    S, E, I, R = y
    dSdt = -beta * S * I / N
    dEdt = beta * S * I / N - sigma * E
    dIdt = sigma * E - gamma * I
    dRdt = gamma * I
    return [dSdt, dEdt, dIdt, dRdt]

def rk45_seir(beta, gamma, sigma, S0, E0, I0, R0, t_eval, N):
    """Solve SEIR with solve_ivp (RK45)."""
    sol = solve_ivp(
        fun=lambda t, y: seir_ode(t, y, beta, gamma, sigma, N),
        t_span=(t_eval[0], t_eval[-1]),
        y0=[S0, E0, I0, R0],
        t_eval=t_eval,
        method="RK45"
    )
    if not sol.success:
        raise RuntimeError("SEIR solve_ivp failed: " + sol.message)
    S, E, I, R = sol.y
    return S, E, I, R
# Assume no initially exposed individuals
E0 = 0.0

def seir_sse_first_half(beta, gamma):
    """SSE between SEIR RK45 model and data I(t) on FIRST HALF only."""
    if beta <= 0 or gamma <= 0:
        return 1e30
    try:
        S_model, E_model, I_model, R_model = rk45_seir(
            beta=beta,
            gamma=gamma,
            sigma=sigma,
            S0=S0,
            E0=E0,
            I0=I0,
            R0=R0,
            t_eval=t_all,
            N=N
        )
    except RuntimeError:
        return 1e30
    return np.sum((I_model[:mid] - I_true_fit) ** 2)

# Grid search for SEIR beta, gamma using first half
```

```
beta_values_seir  = np.linspace(0.01, 1.0, 50)
gamma_values_seir = np.linspace(0.05, 1.0, 50)

best_sse_seir    = np.inf
best_beta_seir   = None
best_gamma_seir  = None

for beta in beta_values_seir:
    for gamma in gamma_values_seir:
        sse = seir_sse_first_half(beta, gamma)
        if sse < best_sse_seir:
            best_sse_seir   = sse
            best_beta_seir  = beta
            best_gamma_seir = gamma

print("SEIR RK45 best beta (first half):", best_beta_seir)
print("SEIR RK45 best gamma (first half):", best_gamma_seir)
print("SEIR RK45 best SSE on FIRST HALF:", best_sse_seir)

# Run SEIR RK45 with best-fit parameters on full time range
S_best_seir, E_best_seir, I_best_seir, R_best_seir = rk45_seir(
    beta=best_beta_seir,
    gamma=best_gamma_seir,
    sigma=sigma,
    S0=S0,
    E0=E0,
    I0=I0,
    R0=R0,
    t_eval=t_all,
    N=N
)

# SSE on second half (prediction error) for SEIR
I_model_second_seir  = I_best_seir[mid:]
sse_second_half_seir = np.sum((I_model_second_seir - I_true_second) ** 2)
print("SEIR RK45 SSE on SECOND HALF:", sse_second_half_seir)


# ============================================================
# 3. Plot comparison: data vs SIR vs SEIR
# ============================================================

plt.figure(figsize=(10, 5))
plt.plot(dates_all, I_true_all, label="True I(t)", linewidth=2)
plt.plot(dates_all, I_best_rk45, "--", label="SIR RK45 I(t) (fit on first half)")
plt.plot(dates_all, I_best_seir, ":", label="SEIR RK45 I(t) (fit on first half)")
plt.axvline(dates_all[mid], color="k", linestyle=":", label="Train/test split")

plt.xlabel("Date")
plt.ylabel("Infected people")
plt.title("SIR vs SEIR (RK45): fit on first half, predict second half")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
SIR RK45 best beta (first half): 0.9797959183673469
SIR RK45 best gamma (first half): 0.8836734693877552
SIR RK45 best SSE on FIRST HALF: 124408.63390074746
SIR RK45 SSE on SECOND HALF: 5133231354.87359
SEIR RK45 best beta (first half): 0.333265306122449
SEIR RK45 best gamma (first half): 0.1663265306122449
SEIR RK45 best SSE on FIRST HALF: 129959.0589159709
SEIR RK45 SSE on SECOND HALF: 12821138608.950537
```



Comparison of SIR RK45 vs. SEIR RK45 on Second-Half Prediction

- When both models were fit only to the first half of the measles data and then used to predict the second half, the results were:
  - SIR RK45 SSE (2nd half): 5,133,231,354.87
  - SEIR RK45 SSE (2nd half): 12,821,138,608.95
- The SEIR model produced a much larger second-half SSE than the SIR model, meaning the extended model did not improve predictive accuracy in this case. Even though SEIR is more biologically realistic—since measles certainly has an exposed (latent) period—the model performed worse when evaluated out-of-sample.

Verify and validate your analysis:

To make sure our SIR model was actually working the way it was supposed to, we checked our results using two different numerical methods: Euler's method, which we used for most of the project, and RK45 from solve_ivp. Both methods gave us basically the same overall trend in the infection curve, but RK45 looked smoother and handled the steep parts of the curve better. This helped us feel confident that our differential equations were coded correctly and that any differences came from numerical error, not a mistake in our formulas.

Next, we wanted to see if our fitted parameters (β and γ) were reliable. To do this, we fit them using only the first half of our measles dataset and then used those values to predict the second half. The fitted β and γ ended up being really close to the ones we got when

using the full dataset, which told us the model was pretty stable. When we calculated how far off the predictions were, the second-half SSE came out to:

SSE = 5,618,617,930.049551

This number looks huge, but that's just because we're working with a 206 million–person population. What really matters is that the predicted curve followed the same general shape as the real data.

We also checked if the parameters made sense biologically. Measles usually has an infectious period of around two weeks, so our assumption of a 2-week infectious period matched really well. Our calculated R_0 value also lined up with what's known about measles being extremely contagious, which made us feel more confident in the results.

We also tried expanding the model, like using an SEIR version. Even though we didn't get it fully working, trying it helped us understand why the basic SIR model sometimes doesn't match real data perfectly. Measles has a real incubation period that SIR just doesn't include.

Overall, by comparing methods, checking stability, matching our values to known measles biology, and experimenting with a more complex model, we were able to verify and validate our analysis pretty well.

Conclusions and Ethical Implications:

After analyzing the measles data from Nigeria (2020–2021) and running our SIR model, we found that the model could capture the big-picture trends in the outbreak but couldn't match every week exactly. This makes sense because measles cases can vary from week to week depending on reporting, vaccination access, population density, and other real-life factors. Our fitted β and γ values were consistent with what's known about measles, so even though the model wasn't perfect, it still gave us meaningful results.

When thinking about the ethical side of our analysis, we realized a few important things. First, measles outbreaks don't affect everyone equally. Areas with low vaccination coverage, especially places where kids don't have easy access to healthcare, end up experiencing the worst outbreaks. Because of this, any recommendations based on our model should prioritize fairness and focus on helping the communities that need it the most.

Another ethical consideration is that our dataset came from digitizing NCDC plots. That means there could be some inaccuracies, so we can't treat our results as exact. Plus, the SIR model is super simple. It ignores age groups, regional differences, travel, behavior changes, and everything else that happens in the real world. So we shouldn't use this model alone to make big decisions about public health.

Even with these limitations, our results still support increasing vaccination coverage, improving surveillance, and focusing on regions where routine immunization is low. The

important thing is being honest about what the model can and can't tell us.

Limitations and Future Work:

While working on this project, we realized that there are several limitations in both our data and the model we used. One major limitation is that the SIR model treats all of Nigeria as one big population, which definitely isn't realistic. Measles spreads differently depending on where people live, how crowded the area is, and how well vaccination programs are doing. Ignoring these differences makes the model less accurate.

Another problem is that the SIR model doesn't include an incubation period, even though measles clearly has one. That's a big reason why our predictions don't line up perfectly with the real data. The fact that our dataset is weekly instead of daily also makes it harder to pick up smaller changes in infection patterns. And since the data had to be digitized from NCDC graphs, there's definitely some noise and uncertainty there.

Looking ahead, there are several things we would improve if we had more time. First, we would try building a full SEIR model, which includes an exposed compartment and would represent measles more accurately. We would also want to add vaccination effects, like the two-dose schedule used for measles, or even waning immunity. Modeling individual states instead of the entire country would also help, since Nigeria is so diverse. On the technical side, trying out more advanced parameter-fitting methods, like Bayesian methods, could give us uncertainty ranges instead of just single numbers. Finally, using raw NCDC or WHO spreadsheets instead of digitized images would hugely improve data quality.

Even with its limitations, our model still helped us understand the overall measles trends in Nigeria, and it showed us how much more accurate epidemic modeling can become with better data and more advanced methods.

NOTES FROM YOUR TEAM:

- Just plotted I vs t because this demonstrated our data better. S(t)'s values where too high and R(t) grew too quick for a clean visual S,I,R plot.
- We worked together to divide the work evenly.
- We collaberated by sharing ideas and progress in Github.

QUESTIONS FOR YOUR TA:

We have no question at the moment.